



It's Your Life

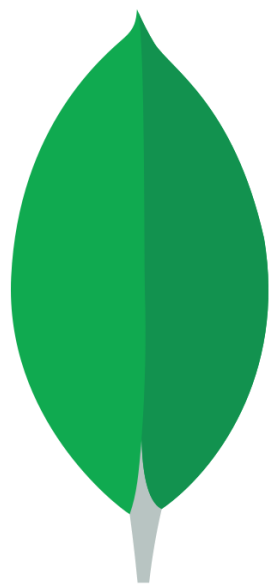
with



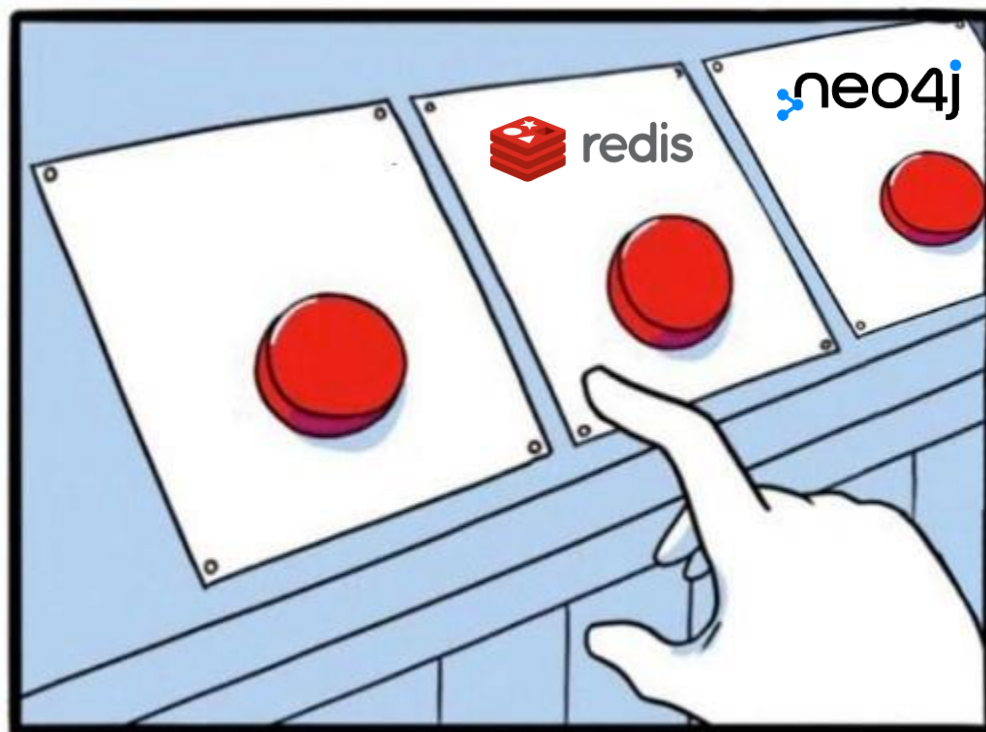


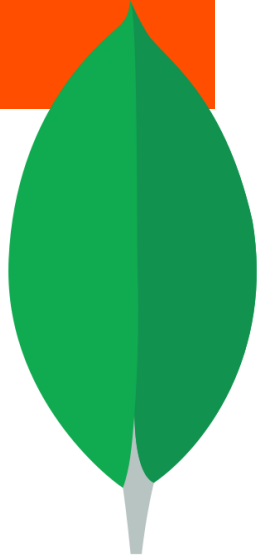
NoSQL





mongoDB®





mongoDB®

Humongous DB



user document

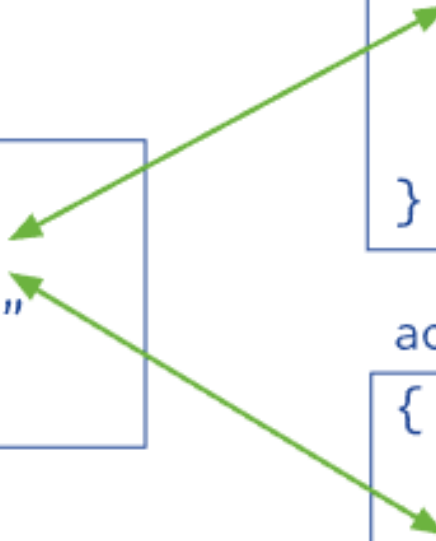
```
{
  _id: <ObjectId1>,
  username: "123xyz"
}
```

contact document

```
{
  _id: <ObjectId2>,
  user_id: <ObjectId1>,
  phone: "123-456-7890",
  email: "xyz@example.com"
}
```

access document

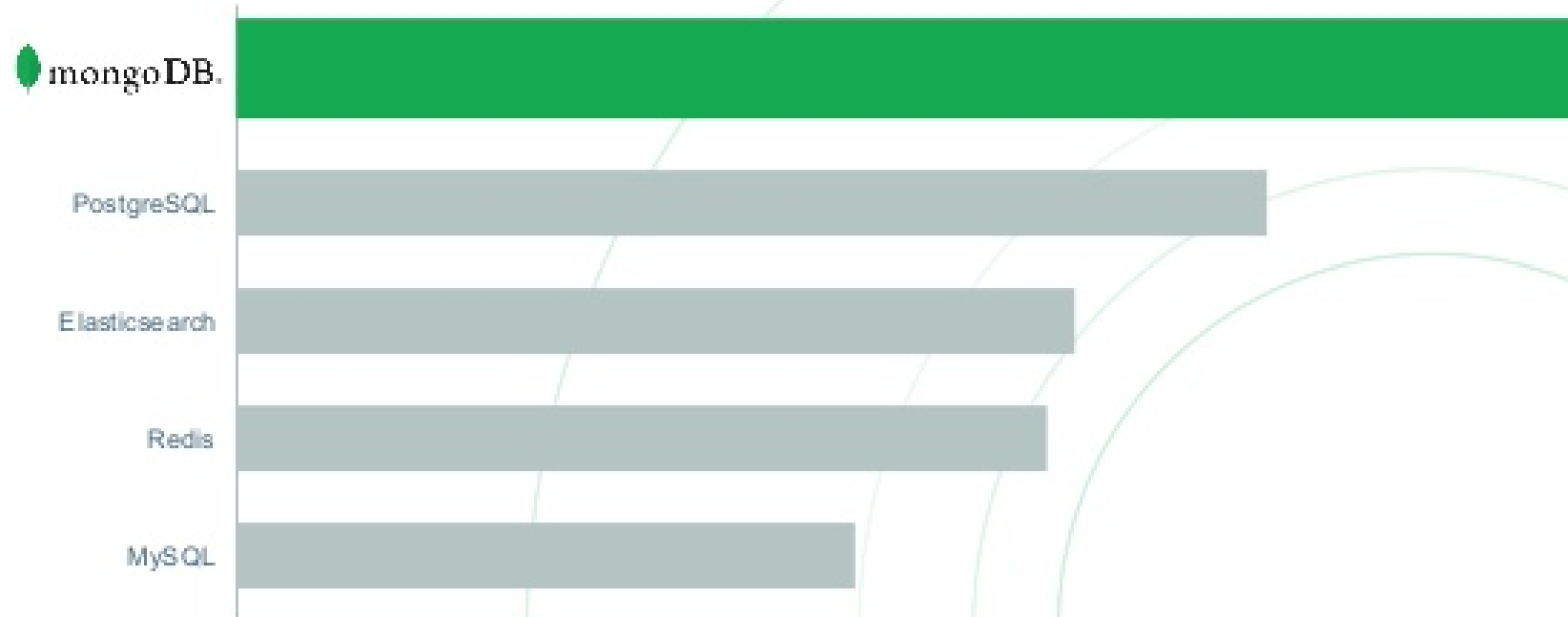
```
{
  _id: <ObjectId3>,
  user_id: <ObjectId1>,
  level: 5,
  group: "dev"
}
```

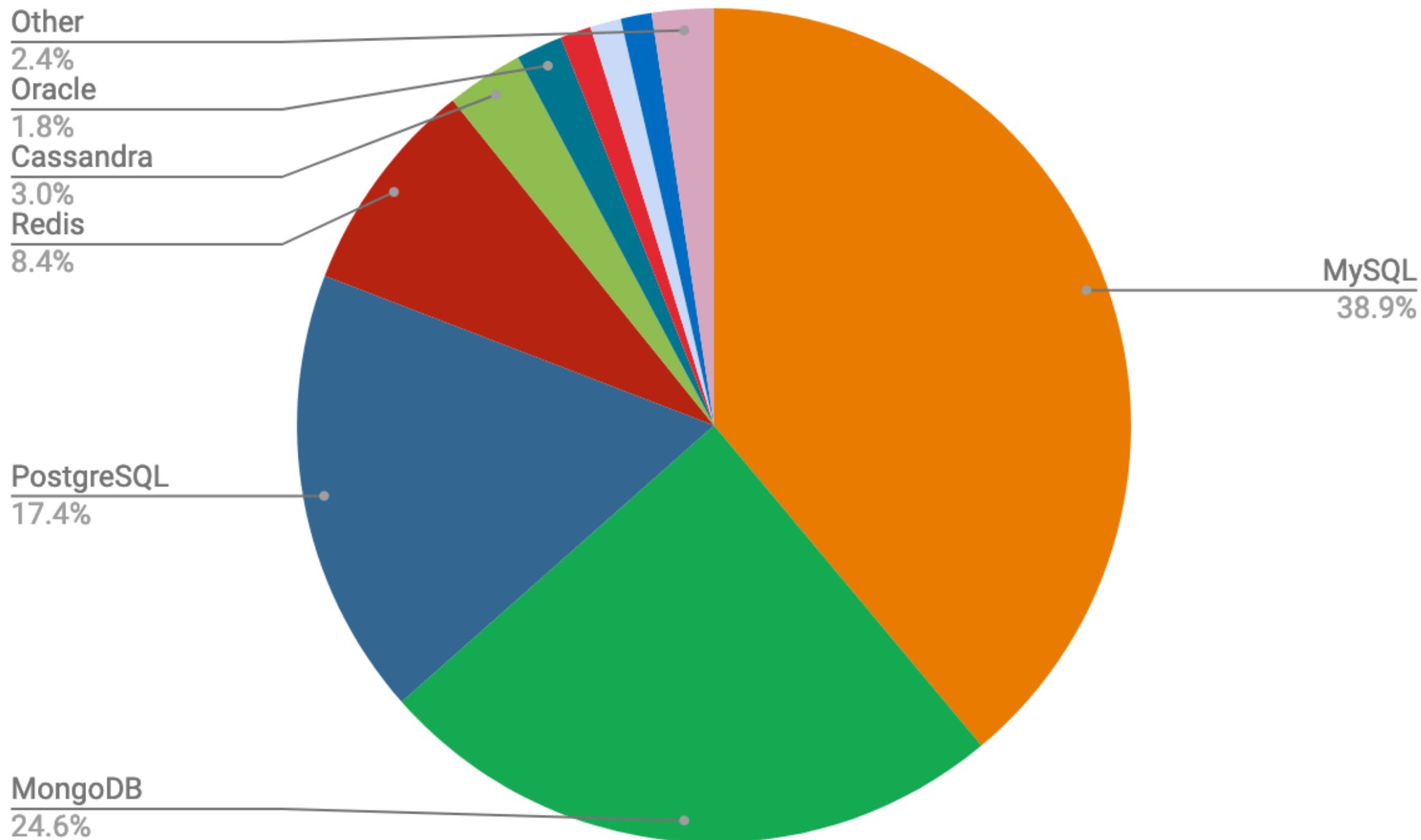






2019년 “최고 인기” 데이터베이스



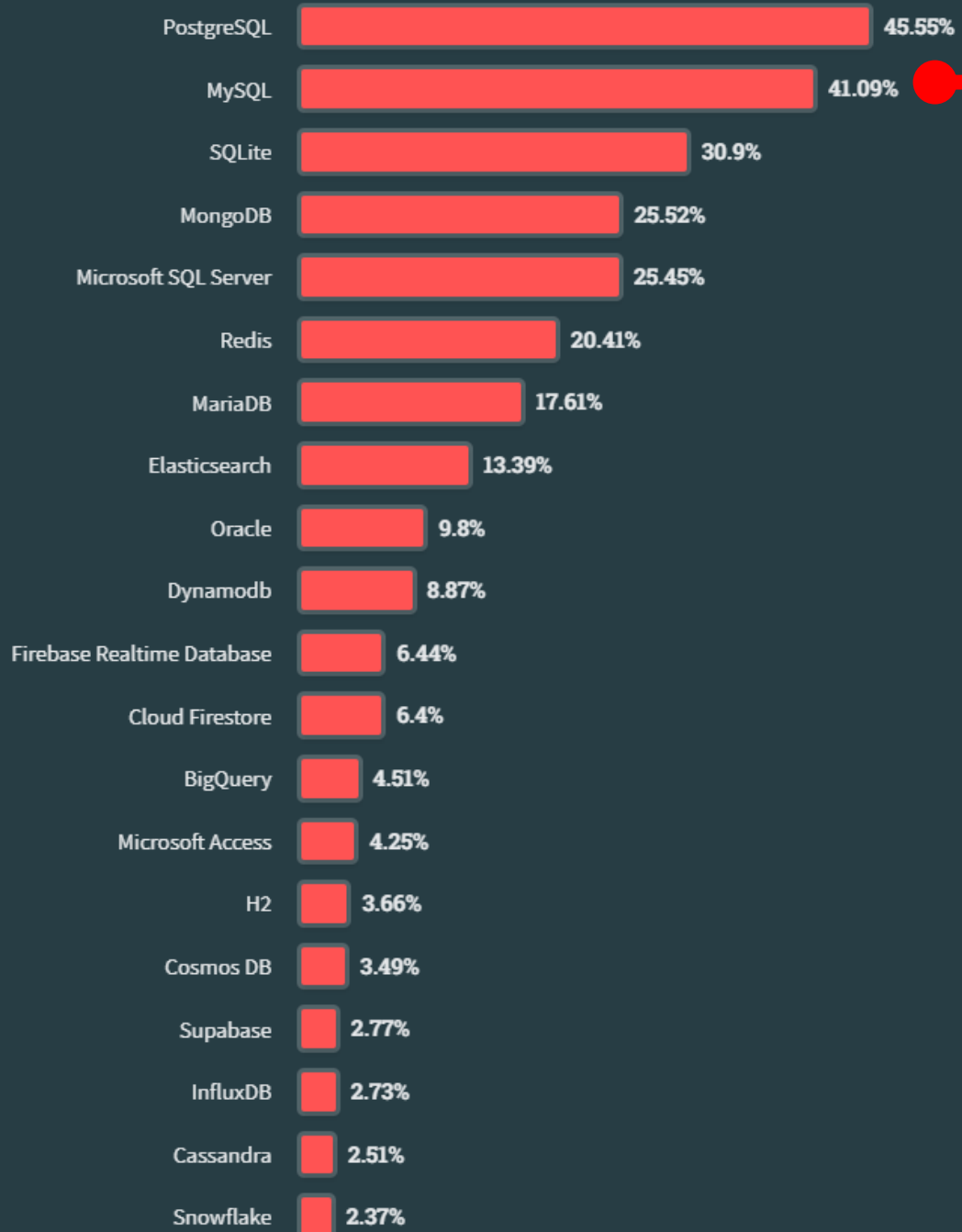




2023 Developer Survey

In May 2023 over 90,000 developers responded to our annual survey about how they learn and level up, which tools they're using, and which ones they want.

[Read the overview →](#)[Methodology →](#)



현 시점
전세계 2등 DB!!





MongoDB 의 구조



MongoDB

Collection1

Document

Document

Document

Document

Document

Document

Collection2

Document

Document

Document

Document

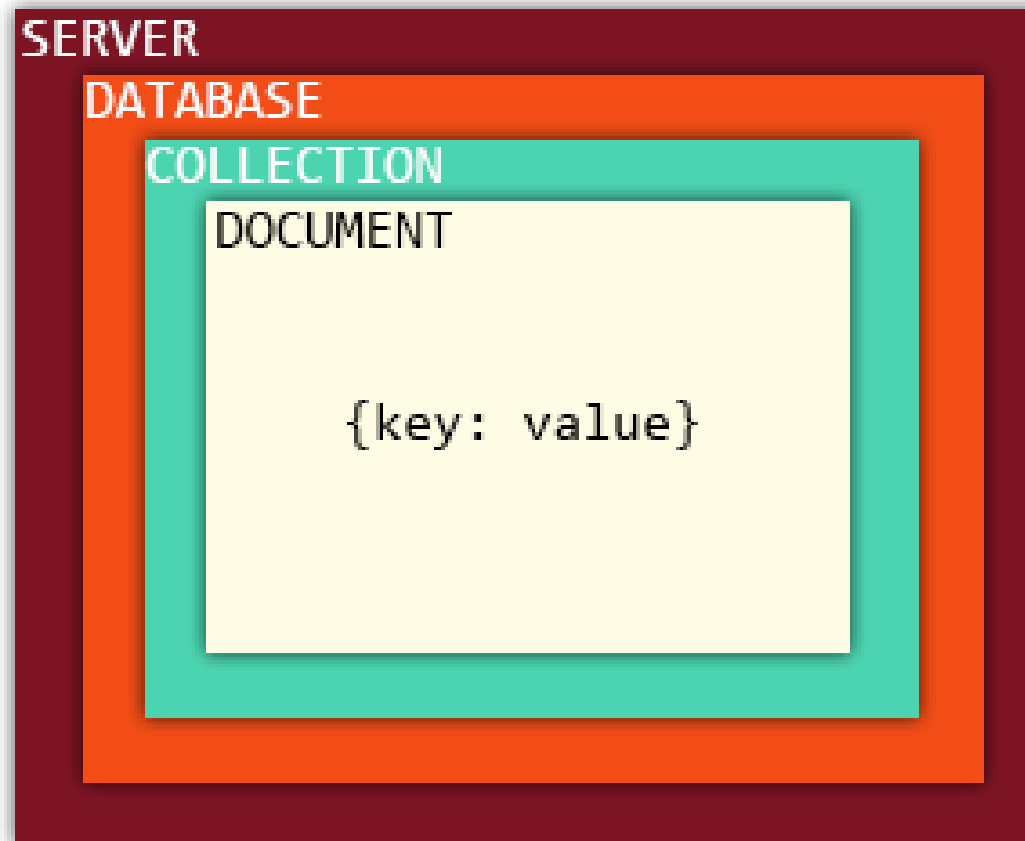
Document



```
{  
  na  
  ag  
  st  
  gr  
}  
  
{  
  na  
  ag  
  st  
  gr  
}  
  
{  
  name: "al",  
  age: 18,  
  status: "D",  
  groups: [ "politics", "news" ]  
}
```

Collection

[click to enlarge](#)



```
{  
  "student": {  
    "name": "John",  
    "class": "Intermediate",  
    "address": {  
      "street": "2293 Example Street",  
      "City": "Chicago",  
      "State": "IL"  
    }  
  }  
}
```

student 객체 안에
address 객체가 또 존재!

즉 중첩이 가능합니다!!

이걸 SQL 로 만들면 어떻게
만들어야 하나요!?



제2 정규형 만족을 위해
address 를 테이블로 빼야 합니다!

```
76  -- students 테이블 생성
77 • CREATE TABLE students (
78     name VARCHAR(50) PRIMARY KEY,
79     class VARCHAR(50)
80 );
81
82  -- addresses 테이블 생성
83 • CREATE TABLE addresses (
84     name VARCHAR(50),
85     street VARCHAR(100),
86     city VARCHAR(50),
87     state VARCHAR(2),
88     FOREIGN KEY (name) REFERENCES students(name)
89 );
```

95 • **SELECT** * **FROM** students;

96 • **SELECT** * **FROM** addresses;

Result Grid		Filter
	name	class
▶	Jhon	Intermediate
*	NULL	NULL

Result Grid					Filter Rows:
	name	street	city	state	
▶	Jhon	2293 Example Street	Chicago	IL	



Result Grid

Filter Rows:

Export:

Wrap Cell C

	name	class	name	street	city	state
▶	Jhon	Intermediate	Jhon	2293 Example Street	Chicago	IL

만약 정보를 한번에 보고 싶다면!?

```

98 • SELECT *
99 FROM students
.00 JOIN addresses
.01 ON students.name = addresses.name;
--

```

JOIN 을 사용해서 두 테이블을
합쳐야만 합니다!



그렇다면 MongoDB 의 장단점은 무엇이 있을까요!?
생각해 봅시다!!





- 장점

- 제약이 없음 → 높은 수평 확장성, 스키마 설계의 유연성
- DB 구조의 변경이 용이
- Data 를 익숙한 JSON 형태로 처리 → 빠르게 객체로 전환 가능

- 단점

- 표준이 없어요(= 제약이 없음)!
- 데이터가 구조화 되어 있지 않음
- 데이터의 일관성 및 안정성을 보장이 불가능
- DB가 아닌 APP 레벨에서 관리해줘야 함 → 버그 발생 확률 높음



시험 당일 새벽 4시
(공부 하나도 안함)



MongoDB DB 생성



및 도큐먼트 추가

Compass



New connection +



Saved connections



Recents



localhost:27017

2024년 7월 22일 오전 9:15

New Connection

Connect to a MongoDB deployment



FAVORITE

URI ⓘ

Edit Connection String ☒

mongodb://localhost:27017/

> Advanced Connection Options

Save

Save & Connect

Connect





localhost:27017 ...

My Queries

Performance

Databases

Search

admin

config

local

startup_log x +

localhost:27017 > local > startup_log

Documents 1 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#) +

+ ADD DATA EXPORT DATA UPDATE DELETE

+ 버튼으로
새로운 DB 생성!

buildinfo : Object



Create Database

x

Database Name

test

Collection Name

users

☐ Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

➤ Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Cancel

Create Database

DATABASE 이름은 test 로!

SQL의 테이블 역할을 하는
Collection 은 users 로!

localhost:27017

My Queries

Performance

Databases

Search

admin

config

local

test

users

users

localhost:27017 > test > users

Documents 0 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

ADD DATA EXPORT DATA UPDATE DELETE

Import JSON or CSV file

Insert document

ADD DATA → Insert document 를 눌러서 데이터 추가!





Insert Document

To collection test.users

VIEW  

```
1  /**
2  * Paste one or more documents here
3  */
4  {
5    "_id": {
6      "$oid": "669db377e834535fc674bcad"
7    },
8    "name": "이효석",
9    "age": 40
10 }
```

Cancel

Insert

MongoDB 의 Document 는
JSON 형태로 관리 되므로
JSON 형태를 유지하면서 데이터를 추가

+ ADD DATA ▾

EXPORT DATA ▾

UPDATE

DELETE



_id: ObjectId('669db377e834535fc674bcad')

name : "이효석"

age : 40

데이터가 추가 된 것을 확인!

_id 는 Primary key 역할을 하며
MongoDB 가 자동으로 생성 합니다!
MongoDB 전체 데이터에서
고유한 값을 가집니다!



Insert Document

To collection test.users

```
1  _id: ObjectId('669db42fe834535fc674bcae')
```

VIEW {}  ObjectId

햄버거 아이콘을 눌러서
새로운 방식으로 데이터를 추가



Insert Document

To collection test.users

```
+ _id: ObjectId('669db42fe834535fc674bcae')
```

+ Add field after _id

+ 버튼 클릭 → Add filed 선택

Insert Document

To collection test.users

```
1  _id: ObjectId('669db42fe834535fc674bcae')
2  name : "김시왕"
3  age : 30
```

VIEW  

ObjectId
String
Int32

데이터 타입 지정이
필요합니다!

Cancel

Insert

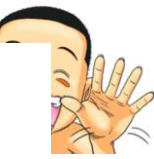
데이터 삽입!



```
_id: ObjectId('669db377e834535fc674bcad')  
name : "이호석"  
age : 40
```

```
_id: ObjectId('669db42fe834535fc674bcae')  
name : "김시완"  
age : 30
```

굿!





다른 타입의 데이터 추가

Insert Document

To collection test.users

VIEW



```
1  /**
2  * Paste one or more documents here
3  */
4  {
5  ▾  "_id": {
6  ▾    "$oid": "669db557e834535fc674bcaf"
7  },
8    "name": "na",
9    "position": "RM",
10   "age": 28
11 }
```

Cancel

Insert

기존 데이터와는 다른 구조의
도큐먼트 삽입!

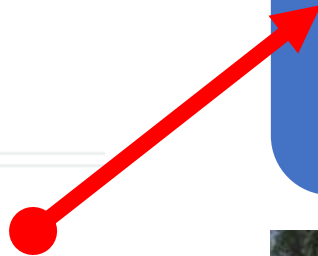




```
_id: ObjectId('669db377e834535fc674bcad')  
name : "이효석"  
age : 40
```

```
_id: ObjectId('669db42fe834535fc674bcae')  
name : "김시완"  
age : 30
```

```
_id: ObjectId('669db557e834535fc674bcaf')  
name : "na"  
position : "RM"  
age : 28
```



구조가 다르지만 문제 없이 삽입!
→ 따라서, 비관계형 DB 라 부릅니다!



초간단 실습, 같은 분단 사람 추가하기!



- 같은 분단에 있는 사람을 users 컬렉션에 전부 추가해 주세요!



데이터

검색 기능 추가



검색 기능을 쓰려면 이렇게
{ 찾을 필드 : "찾고 싶은 값" } 을 작성 후
find 버튼을 누르시면 됩니다!

users x +

localhost:27017 > test > users

Documents 3 Aggregations Schema Indexes 1 Validation

🕒 {name: "이효석"}

💡 Generate query ⚡

Explain

Reset

Find

</>

C

+ ADD DATA ▾

📄 EXPORT DATA ▾

✎ UPDATE

🗑 DELETE

1 - 1 of 1 ↺ < > ☰

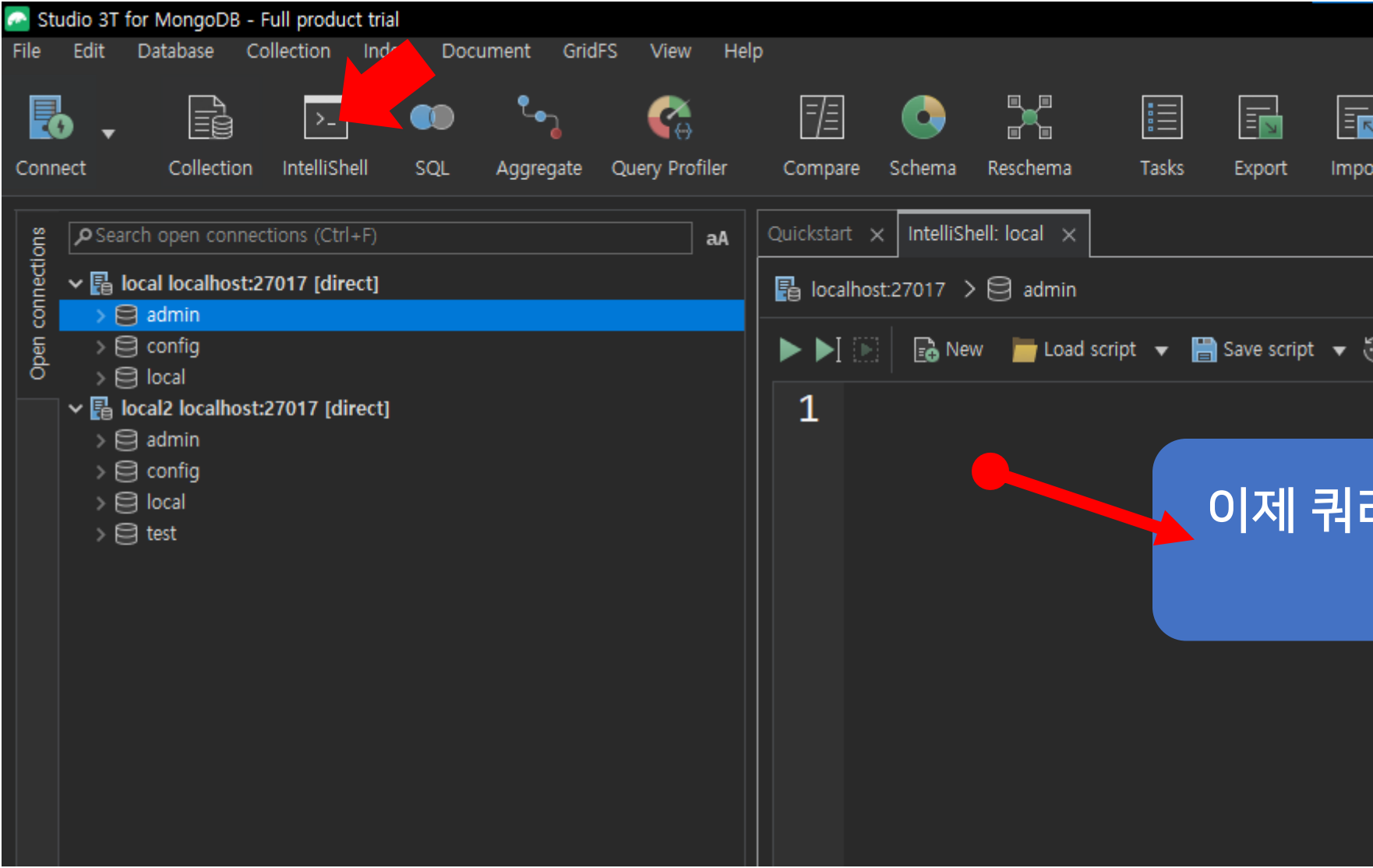
```
_id: ObjectId('669db377e834535fc674bcad')
name : "이효석"
age : 40
```



MongoDB

Query 배우기







Read(= 검색)



```
Quickstart x IntelliShell: local* x
localhost:27017 > test

1 use test; // 테스트 데이터베이스 사용
2
3 // 선택된 테스트 데이터베이스를 db 로 부릅니다!
4 // users 는 컬렉션 명
5 // find() 는 전체 검색
6 db.users.find();
7
```

아무 조건을 안주면
전체 검색이 됩니다~!

Raw shell output Find Query (line 6) x

50 Documents 1 to 3

users > name

_id	name	age	position
id 669db377e834...	이효석	40	
id 669db42fe834...	김시완	30	
id 669db557e834...	na	28	RM

검색 결과가 마치 관계형 DB 처럼
됩니다!!



Create

insertOne



- 하나의 도큐먼트를 삽입합니다

```
8 db.users.insertOne({ name: "문준일", age: 30 });
```

Result > insertedId

acknowledged

insertedId

T/F true

id 669dba612f96...

삽입이 성공하면 아래의 화면에
아래와 같은 화면이 뜹니다!

Result > insertedId	
acknowledged	insertedId
T/F true	id 669dba612f96...

요 화면은 Studio3T 가 객체 데이터를 테이블로 가공 해준 것입니다!

그럼 실제 리턴 값은 어떻게 들어올까요!?

```
{  
  acknowledged : "true",  
  insertedId : ObjectId('669dba')  
}
```



insertMany

- 여러 도큐먼트를 한번에 삽입 합니다
- 삽입할 도큐먼트는 배열에 담긴 객체 형태로 전달 되어야 합니다

```
10 db.users.insertMany([{ name: "최현수", age: 27 }, { name: "이태웅", age: 27 }]);
```

Raw shell output Shell Output (Documents) ✖

← ← → → | 50 | Documents 1 to

Result

acknowledged	insertedIds
✓ true	{ 2 fields }

Raw shell output Find Query (line 6) ✖

↺ ↻ ↹ ↹ | 50 | Documents 1 to 6 | 🔒 📄 🔄 📝 ✖

users > name

_id	name	age	position
[id] 669db377e834535fc674bcad		[i32] 40	
[id] 669db42fe834...	김시완	[i32] 30	
[id] 669db557e834...	na	[i32] 28	RM
[id] 669dba612f96...	문준일	[i32] 30	
[id] 669dbb832f96...	최현수	[i32] 27	
[id] 669dbb832f96...	이태웅	[i32] 27	

초간단 실습2



- insertOne 메서드를 이용해서 각각 이름이 뽀로로, 나이가 9 살인 데이터와 이름이 루피, 나이가 8 살인 데이터를 추가해주세요!
- insertMany 메서드를 이용해서 우리반에 있는 모든 교육생을 users 컬렉션에 추가해주세요!



Update



\$set



\$set: {}

- MongoDB 의 도큐먼트를 수정할 때 사용합니다.
- 수정 Query 에서 도큐먼트를 수정 할 때 \$set: { 수정할 내용 } 으로 수정을 해야 합니다.

```
db.users.updateOne(  
  { name: "na" },  
  { $set: { name: "나건우" } }  
);
```

조건은 name 필드의 값이 'na' 인 도큐먼트

해당 도큐먼트의 name 필드를 전달 한 값으로 수정



updateOne

- 조건을 만족하는 가정 처음의 documento 하나를 수정합니다

```
20 db.users.updateOne({ name: "na" }, { $set: { name: "나건우" } });
```

Raw shell output				
Shell Output (Documents) ✕				
< < > > 50 Documents 1 to 1 🔍				
Result > insertedId				
acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
✓ true	■ null	123 1.0	123 1.0	123 0.0

_id	name	age	position
<input type="text" value="id"/> 669db377e834...	<input type="text" value=""/> 이효석	<input type="text" value="i32"/> 40	
<input type="text" value="id"/> 669db42fe834...	<input type="text" value=""/> 김시완	<input type="text" value="i32"/> 30	
<input type="text" value="id"/> 669db557e834...	<input type="text" value=""/> 나건우	<input type="text" value="i32"/> 28	<input type="text" value=""/> RM
<input type="text" value="id"/> 669dba612f96...	<input type="text" value=""/> 문준일	<input type="text" value="i32"/> 30	
<input type="text" value="id"/> 669dbb832f96...	<input type="text" value=""/> 최현수	<input type="text" value="i32"/> 27	
<input type="text" value="id"/> 669dbb832f96...	<input type="text" value=""/> 이태웅	<input type="text" value="i32"/> 27	
<input type="text" value="id"/> 669dc2ce2f964...	<input type="text" value=""/> 뽀로로	<input type="text" value="i32"/> 9	

이름 값이 변경 된 것 확인 가능!



비교식



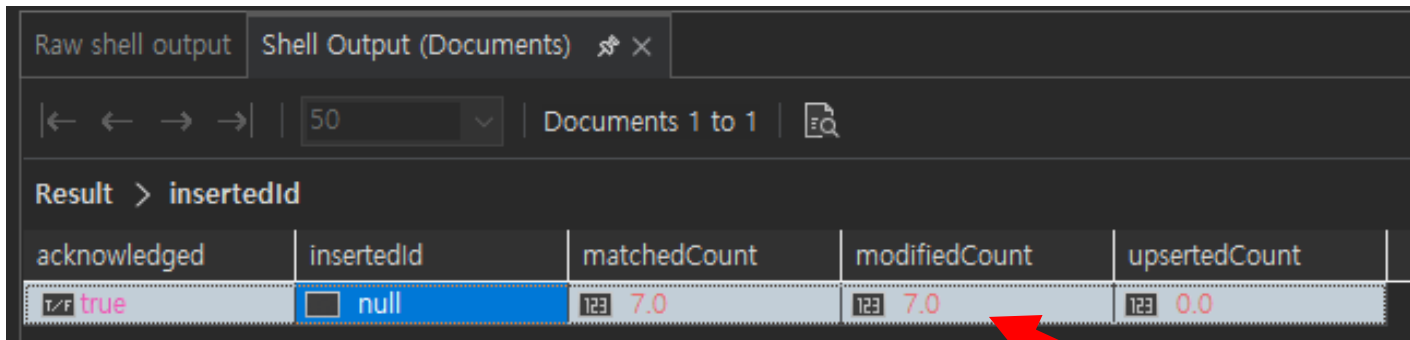
쿼리	설명
\$eq	일치하는 값을 찾는다.
\$gt	지정된 값보다 큰 값을 찾는다.
\$gte	크거나 같은 값을 찾는다.
\$lt	지정된 값보다 작은 값을 찾는다.
\$lte	작거나 같은 값을 찾는다.
\$ne	일치하지 않는 모든 값을 찾는다.(\$eq의 부정)
\$in	배열에 지정된 값 중 하나와 일치한 값을 찾는다.
\$nin	배열에 지정된 값과 일치하지 않는 값을 찾는다.

updateMany

MongoDB의 연산자는 \$를 사용합니다!
해당 구문은 이름이 건우님이 아닌 조건!

- 조건을 만족하는 모든 문서를 수정합니다

```
24 db.users.updateMany({ name: { $ne : "나건우" } }, { $set: { position: "RM 아님" } });
```



acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
T/F true	null	1 7.0	1 7.0	0 0.0

조건에 맞는 문서 개수와
변경이 일어난 문서 개수를 리턴

Raw shell output Find Query (line 17) ✕

50 Documents 1 to 8

users > name

_id	name	age	position
id 669db377e834...	이효석	40	RM 아님
id 669db42fe834...	김시완	30	RM 아님
id 669db557e834...	나건우	28	RM
id 669dba612f96...	문준일	30	RM 아님
id 669dbb832f96...	최현수	27	RM 아님
id 669dbb832f96...	이태웅	27	RM 아님
id 669dc2ce2f964...	뽀로로	9	RM 아님
id 669dc2cf2f964...	루피	8	RM 아님

updateMany 의 반영 결과 확인



Update 와 MongoDB 의 특징

- MongoDB 는 구조가 존재하지 않기 때문에 필드(= SQL 의 컬럼)를 마음대로 추가 가능 합니다!

현재의 컬렉션!

users > name			
_id	name	age	position
[id] 669db377e834...	이효석	[i32] 40	RM 아님
[id] 669db42fe834...	김시완	[i32] 30	RM 아님
[id] 669db557e834...	나건우	[i32] 28	RM
[id] 669dba612f96...	문준일	[i32] 30	RM 아님
[id] 669dbb832f96...	최현수	[i32] 27	RM 아님
[id] 669dbb832f96...	이태웅	[i32] 27	RM 아님
[id] 669dc2ce2f964...	보로로	[i32] 9	RM 아님
[id] 669dc2cf2f964...	루피	[i32] 8	RM 아님

```
37 db.users.updateMany(  
38   { age: { $lte: 30 } },  
39   { $set: { status: "파릇파릇함" } }  
40 );
```

나이가 30살 이하이면!



기존에 없던 필드인 status 에 값을
추가하는 쿼리!



users > name				
_id	name	age	position	status
[id] 669db377e834...	["_"] 이효석	[i32] 40	["_"] RM 아님	
[id] 669db42fe834...	["_"] 김시완	[i32] 30	["_"] RM 아님	["_"] 파릇파릇함
[id] 669db557e834...	["_"] 나건우	[i32] 28	["_"] RM	["_"] 파릇파릇함
[id] 669dba612f96...	["_"] 문준일	[i32] 30	["_"] RM 아님	["_"] 파릇파릇함
[id] 669dbb832f96...	["_"] 최현수	[i32] 27	["_"] RM 아님	["_"] 파릇파릇함
[id] 669dbb832f96...	["_"] 이태웅	[i32] 27	["_"] RM 아님	["_"] 파릇파릇함
[id] 669dc2ce2f964...	["_"] 뽀로로	[i32] 9	["_"] RM 아님	["_"] 파릇파릇함
[id] 669dc2cf2f964...	["_"] 루피	[i32] 8	["_"] RM 아님	["_"] 파릇파릇함



Delete

deleteOne



- 조건을 만족하는 가장 처음의 도큐먼트 하나를 삭제합니다

```
15 db.users.deleteOne({ name: "뽀로로" });
```

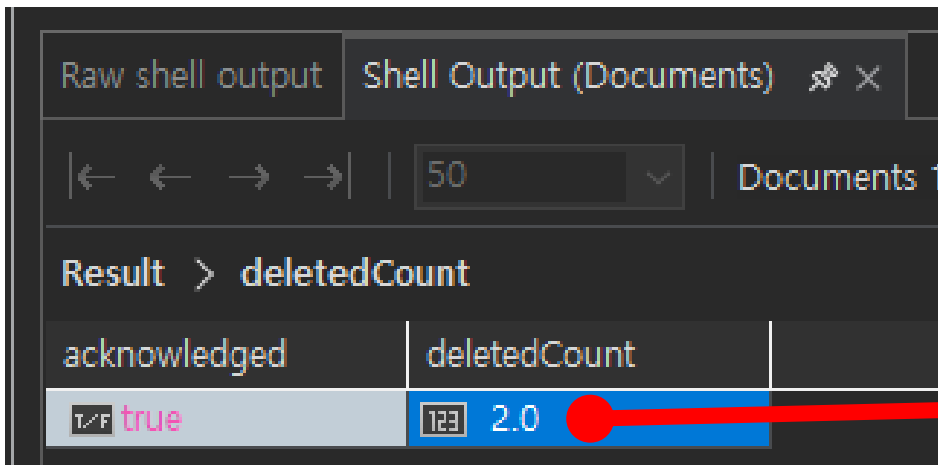
Raw shell output		Shell Output (Documents) ✖	
← ← → →		50	Documents
Result > deletedCount			
acknowledged	deletedCount		
T/F true	123 1.0		

deleteMany

- 조건을 만족하는 모든 문서를 삭제

MongoDB의 연산자는 \$를 사용합니다!
해당 구문은 age 필드가 10살 이하 조건!

```
16 db.users.deleteMany({ age: { $lte: 10 } });
```



The screenshot shows the MongoDB Shell interface. At the top, there are tabs for 'Raw shell output' and 'Shell Output (Documents)'. Below the tabs, there are navigation arrows and a dropdown menu showing '50' and 'Documents 1'. The main area displays the result of the operation: 'Result > deletedCount'. Below this, there is a table with two columns: 'acknowledged' and 'deletedCount'. The first row shows 'true' for 'acknowledged' and '2.0' for 'deletedCount'. A red dot is placed on the '2.0' value, with a red arrow pointing to a callout box.

acknowledged	deletedCount
true	2.0

해당 쿼리로 몇 개의 데이터가 삭제되었는지
정보를 리턴!



Read(= 검색)

findOne



- 검색 조건을 만족하는 최초의 도큐먼트를 1개를 찾아 줍니다

```
32 db.users.findOne({ name: "이효석" });
```

Raw shell output		Find Query (line 32)	
↺ ↻ ↻ ↻ ↻		50	Documents 1 to 1
users > name			
_id	name	age	position
id 669db377e834...	이효석	40	RM 아님

find

나이가 30살 미만인 교육생 검색

- 검색 조건을 만족하는 모든 도큐먼트를 찾아 줍니다!

```
34 db.users.find({ age: { $lt : 30 } });
```

users > name				
_id	name	position	age	status
[id] 669db557e834...	[name] 나건우	[position] RM	[age] 28	[status] 파릇파릇함
[id] 669dbb832f96...	[name] 최현수	[position] RM 아님	[age] 27	[status] 파릇파릇함
[id] 669dbb832f96...	[name] 이태웅	[position] RM 아님	[age] 27	[status] 파릇파릇함
[id] 669dc2ce2f964...	[name] 뽀로로	[position] RM 아님	[age] 9	[status] 파릇파릇함
[id] 669dc2cf2f964...	[name] 루피	[position] RM 아님	[age] 8	[status] 파릇파릇함



논리식


```
db.컬렉션명.find({쿼리: [{조건1}, {조건2}, ...]}))
```



쿼리	설명
\$or	조건들 중 하나라도 true면 반환 (true: 조건과 일치, false: 조건과 불일치)
\$and	조건들이 모두 true일 때 반환
\$not	조건이 false일 때 반환
\$nor	조건들이 모두 false일 때 반환

```
44 db.users.find({
45   $and: [
46     { position: "RM" },
47     { age: { $gt: 20 } }
48   ]
49 });
```

\$and 논리는 조건이 2개 이상이라는
의미이므로 배열을 통해서 전달!

position 의 값이 "RM" 이면서
age 의 값이 20 이상인 데이터 찾기!

users > name				
_id	name	position	age	status
id 669db557e834...	나건우	RM	28	파릇파릇함

```

51 db.users.find({
52     $or: [
53         { name: { $ne: "이효석" } },
54         { age: { $lt: 25 } }
55     ]
56 });

```

\$or 논리는 조건이 2개 이상이라는 의미이므로 배열을 통해서 전달!

이름이 "이효석"이 아니면서
나이가 25 미만인 교육생!

users > name

_id	name	age	position	status
[id] 669db42fe834...	[name] 김시완	[age] 30	[position] RM 아님	[status] 파릇파릇함
[id] 669db557e834...	[name] 나건우	[age] 28	[position] RM	[status] 파릇파릇함
[id] 669dba612f96...	[name] 문준일	[age] 30	[position] RM 아님	[status] 파릇파릇함
[id] 669dbb832f96...	[name] 최현수	[age] 27	[position] RM 아님	[status] 파릇파릇함
[id] 669dbb832f96...	[name] 이태웅	[age] 27	[position] RM 아님	[status] 파릇파릇함
[id] 669dc2ce2f964...	[name] 뽀로로	[age] 9	[position] RM 아님	[status] 파릇파릇함
[id] 669dc2cf2f964...	[name] 루피	[age] 8	[position] RM 아님	[status] 파릇파릇함

실습, 데이터 삽입 - 수정 - 삭제 - 검색 하기



- test2 데이터 베이스, users 컬렉션을 만들어 주세요
- 자신 짝궁의 이름과 나이 정보를 insertOne 쿼리를 이용하여 컬렉션에 추가해 주세요
- 자신과 같은 분단의 교육생 이름과 나이 정보를 insertMany 쿼리를 이용하여 컬렉션에 추가해 주세요
- 강사의 정보를 insertOne 쿼리를 이용하여 추가해 주세요
- 나이가 37세 이상인 도큐먼트에 status 필드를 추가하고 "늙음" 이라는 값을 넣어 주세요

실습, 데이터 삽입 - 수정 - 삭제 - 검색 하기



- 나이가 37세 이상이면서 이름이 "이효석"인 데이터를 찾아서 삭제해 주세요
- 나이가 25세 이상인 사람들만 출력해 주세요