MongoDB

고려대학교 KOREA UNIVERSITY 정보대학 컴퓨터학과

NoSQL

■ 기존 RDBMS의 초점 : 트랜잭션을 통한 안정적인 데이터 관리에 초점

○ 단일 하드웨어에서의 동작을 가정

■ 웹 2.0 환경 및 빅데이터 등장

○ 데이터와 트래픽의 양이 기하급수적으로 증가

○ 수직적 확장 구조로 데이터를 처리하는 데 필요한 비용의 증가

○ 기존의 RDBMS로는 빅데이터를 처리하는데 한계에 도달

■ 고정된 스키마(Fixed Schema)

○ 데이터를 사용하기 전에 스키마(Table, Index 등)를 정의해야 함

○ 데이터 속성이 고정되지 않거나, 데이터의 속성의 변화가 잦은 환경에서 단점으로 작용

○ 스키마 수정의 어려움: 컬럼의 추가/수정/삭제는 row에 lock을 걸고 index의 수정은 테이블에 lock을 걸기 때문

■ JOIN

○ 데이터가 많을 때 JOIN은 많은 양의 데이터에 복잡한 연산을 수행해야 하기 때문에 비용이 많이 들며 파티션을 넘어서는 동작하지 않음

■ ACID 트랜잭션

○ Atomicity(원자성): 트랜잭션과 관련된 작업들이 부분적으로 실행되다가 중단되지 않는 것을 보장

○ Consistency(일관성): 트랜잭션이 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 유지

○ Isolation(격리성): 트랜잭션을 수행 시 다른 트랜잭션의 연산 작업이 끼어들지 못하도록 보장

○ Durability(지속성): 성공적으로 수행된 트랜잭션은 영원히 반영

■ Example : A 계좌에서 B 계좌로 50만원을 송금하는 상황

○ 1. read(A)

○ 2. A := A – 500,000

○ 3. write(A)

○ 4. read(B)

○ 5. B := B + 500,000

○ 6. write(B)

■ 만약 3번까지 진행된 상태에서 시스템에 오류가 생겨서 시스템이 정지된다면?

○ A 계좌에서는 50만원이 인출되었지만, B 계좌에는 아무런 돈이 입금되지 않음

■ Atomicity(원자성)

○ Transaction이 수행되다 말고 도중에 정지될 경우, 이를 원상태로 복구(roll back)해야 함

○ All or Nothing

■ Example : A 계좌에서 B 계좌로 50만원을 송금하는 상황

○1. read(A)

○2. A := A – 500,000

○3. write(A)

○4. read(B)

○5. B := B + 500,000

○6. write(B)

■ 만약 3번까지 진행된 상태에서 시스템에 오류가 생겨서 시스템이 정지된다면?

○ A 계좌에서는 50만원이 인출되었지만, B 계좌에는 아무런 돈이 입금되지 않음

■ Consistency(일관성)

○ A계좌와 B계좌의 금액의 총합은 일정해야 함.

○ Consistency를 만족하지 못할 경우, transaction을 roll back 해야함

■ Example : A 계좌에서 B 계좌로 송금하는 도중에 C에게도 40만원을 송금

○1. A = 700,000

○2. read(A)

○3. A := A – 500,000

○4. write(A)

○5. read(B)

○6. B := B + 500,000

○7. write(B)

1. read(A)
2. A := A – 400,000
3. write(A)
4. read(C)
5. C := C + 400,000
6. write(C)

■ A가 출금한 후 변경 금액을 갱신하기 전에, 다른 작업에서 A의 잔액을 읽어서 처리할 경우

○ A는 30만원의 잔액이 남을 것

■ Isolation(격리성)

○ A계좌에서 읽고 쓰기 작업 중이라면, 다른 작업이 끼어들어서는 안됨.

■ Example : A 계좌에서 B 계좌로 50만원을 송금하는 상황

○ 1. read(A)

○ 2. A := A – 500,000

○ 3. write(A)

○ 4. read(B)

○ 5. B := B + 500,000

○ 6. write(B)

■ 만약 6번까지 모든 작업이 완료되었다면

○ 아무런 문제가 없음

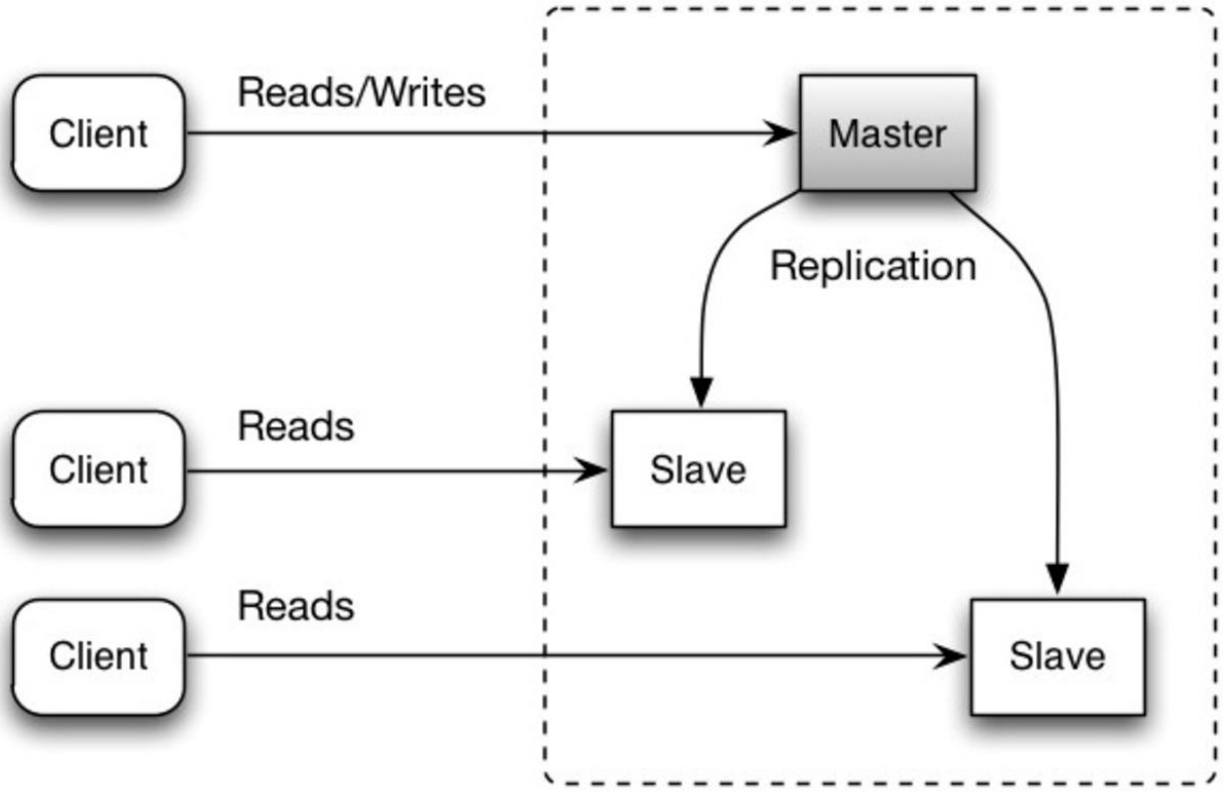
■ Durability(지속성)

○ 성공적으로 수행된 트랜잭션은 영원히 반영해야 함

○ 시스템에 오류가 생겨도, 성공 이전으로 돌아가서는 안됨

■ Master-Slave Replication

○ 데이터를 안정적으로 보존하기 위해 slave에 데이터 복제



■ Replication - 복제에 의한 확장

○ 결과를 slave의 개수만큼 복제함.

○ 용량 확장이 아닌 시스템의 안정성 부여

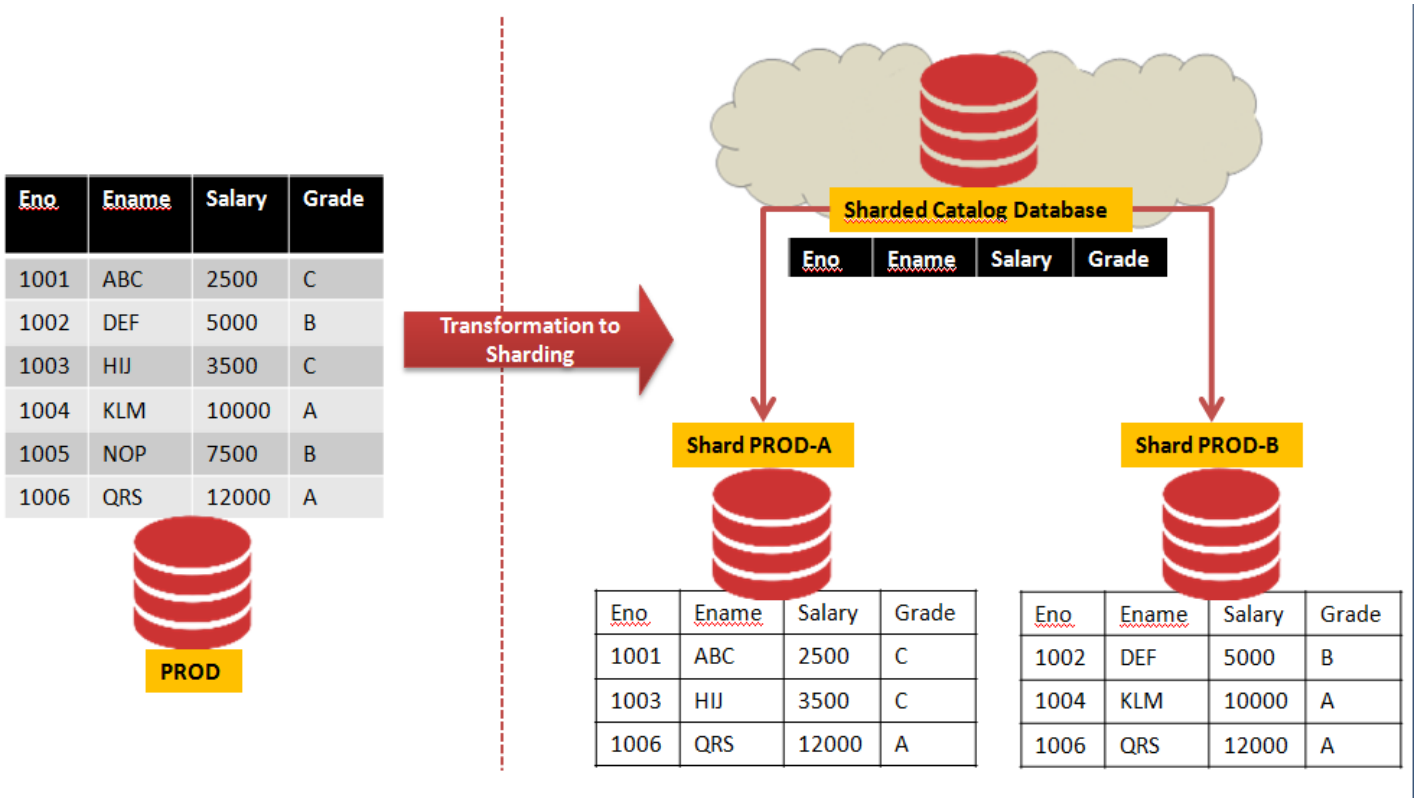
○ 읽기는 빠르지만 쓰기는 하나의 노드에 대해서만 일어나기 때문에 병목현상 발생

○ Master에서 slave로 퍼지는데 시간이 소요되기 때문에 중요한 읽기 작업은 여전히 Master에서 읽어야 하고, 이것은 어플리케이션 개발에 고려가 필요함

○ 데이터 규모가 큰 경우에는 N번 복제를 해야 하기 때문에 문제가 발생할 소지가 있음. 이것은 Master-Slave 방식으로 확장성에 대한 제한을 가지게 됨.

■ Sharding(Partitioning)

○ 데이터를 각각의 DB에 나눠서 분할 관리



■ Sharding(Partitioning) - 분할에 의한 확장

○ Read만큼 Write도 확장할 수 있지만 애플리케이션에서 파티션된 것을 인지하고 있어야 함

○ RDBMS의 가치는 관계에 있다고 할 수 있는데 파티션을 하면 이 관계가 깨져버리고 각 파티션 된 조각 간에 조인을 할 수 없기 때문에 관계에 대한 부분은 애플리케이션에서 처리

○ RDBMS에서는 수동 Sharding 작업이 필요하나 이 과정이 복잡

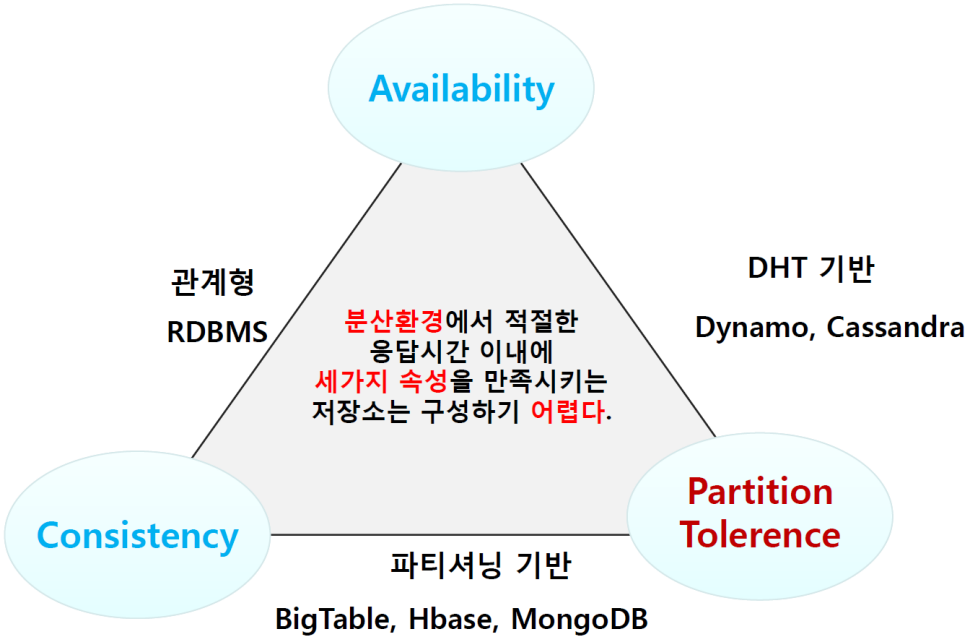
■ 분산 시스템이 보장해야 할 3가지 특성

○ Consistency: 각각의 사용자가 항상 동일한 데이터를 조회해야함.

○ Availability: 모든 사용자가 항상 읽고 쓸 수 있음

○ Partition tolerance: 물리적 네트워크 분산 환경에서 시스템이 잘 동작

■ 분산 시스템에서는 적절한 타이밍에 2가지 특성만 만족할 수 있다.



■ **1998년: SQL interface를 지원하지 않은 오픈소스 관계형DB를 NoSQL이라 정의**

○시스템 구조의 단순화

○시스템을 이기종 장비 에도 이식시킬 수 있게 함

○표준 상용 제품보다 기능을 줄이면서 가격도 저렴하게 함

○**데이터 필드 크기, 컬럼 등의 제약 없앰**

■ **2009년: NoSQL 개념 재등장. 기존 관계형 DBMS와 다른 특징으로 규정**

○**Non-Relational**

○**No-Schema**

○**Distributed**

○**Less Restricted ACID**

■ **2011년: UnQL(Unstructurred Query Language) 활동 시작**

No SQL

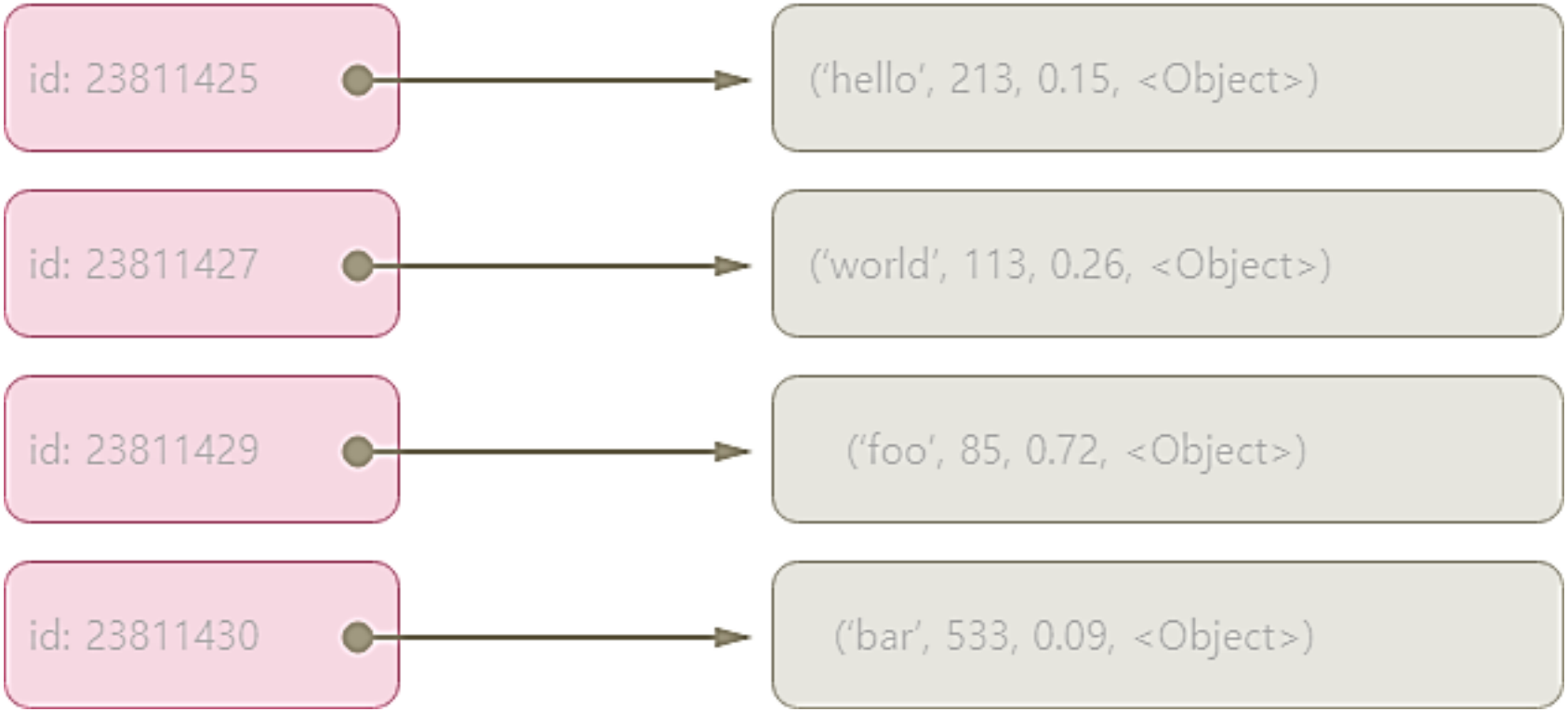
Not only SQL

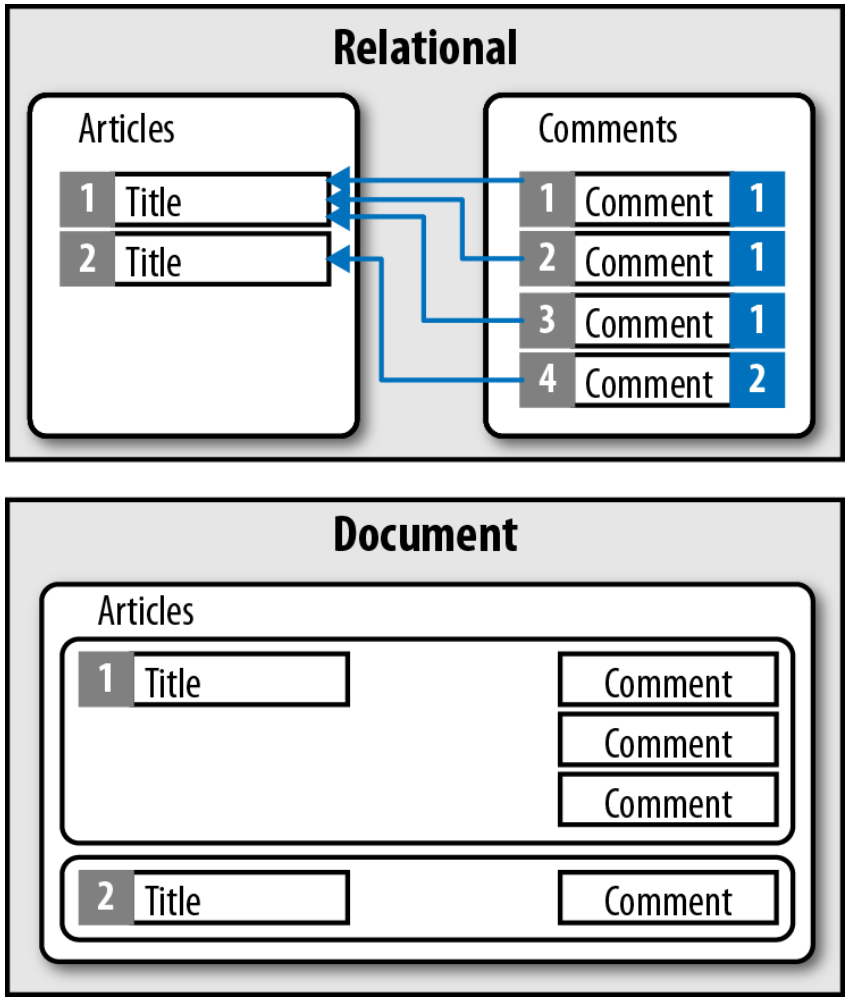
NOn-relational operation database SQL

집합 지향 모델 VS 그래프 모델

구분		특징	종류
집합 지향 모델	키-값 모델	● 데이터를 키와 값의 쌍으로 저장 ● 키: 값에 접근하기 위한 index ● 값: 모든 형태의 데이터	Voldemort Riak Redis … …
	문서 모델	● 키와 문서의 쌍으로 데이터 저장 ● 데이터를 계층적 문서 형태로 저장 ● 문서의 구조를 정하고 이 구조를 기초로 하여 데이터에 접근	**MongoDB** RavenDB couchDB … …
	칼럼-패밀리 모델	● 기존 관계형 모델과 다르게 행(row)가 아닌 열(column),즉 데이터의 필드를 기본 단위로 데이터 관리	Cassandra Hbase HyperTable …. …
그래프 모델		● 관계형 모델과 유사 ● 실제 세계의 데이타를 관계와 함께 표현하기 위해 디자인된 모델 ● 연속적인 노드, 관계, 특성의 형태로 저장	InfiniteGraph Neo4j OrientDB … …

키-값 모델(Key-Value Model)




Relational
Articles
1 Title
2 Title
Comments
1 Comment 1
2 Comment 1
3 Comment 1
4 Comment 2
Document
Articles
1 Title
Comment
Comment
Comment
2 Title
Comment

row-store

+ easy to add/modify a record

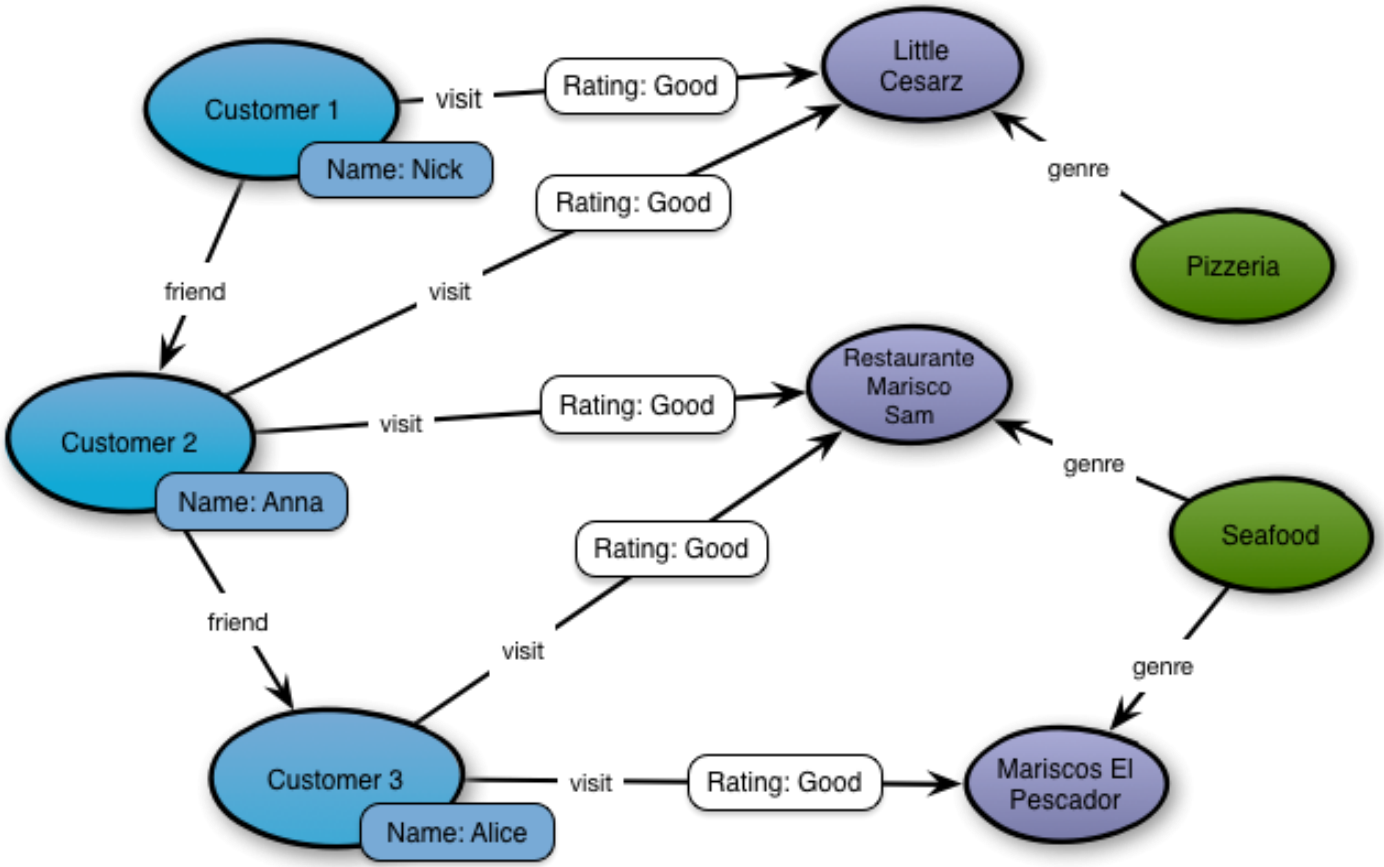
- might read in unnecessary data

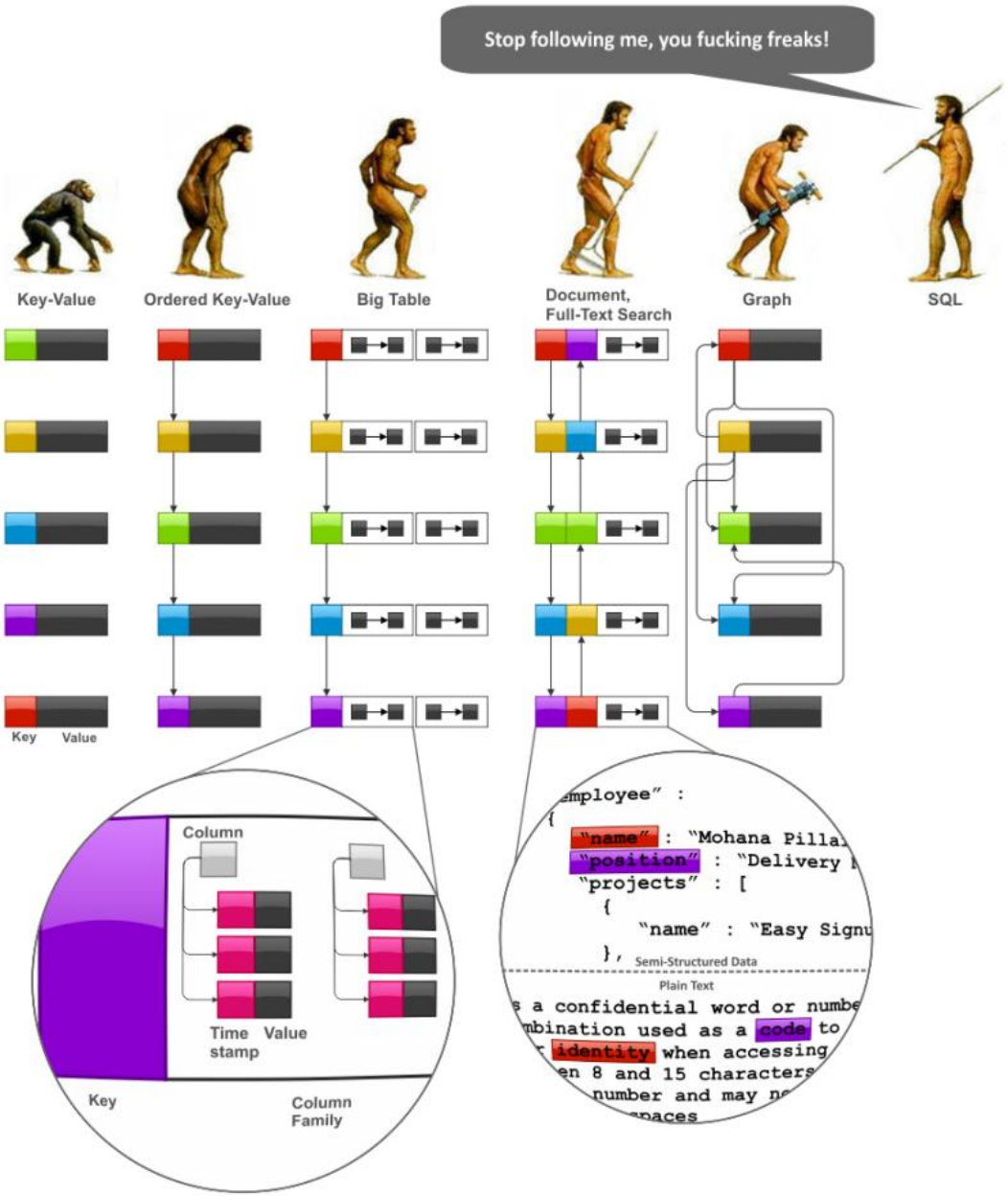
column-store

+ only need to read in relevant data

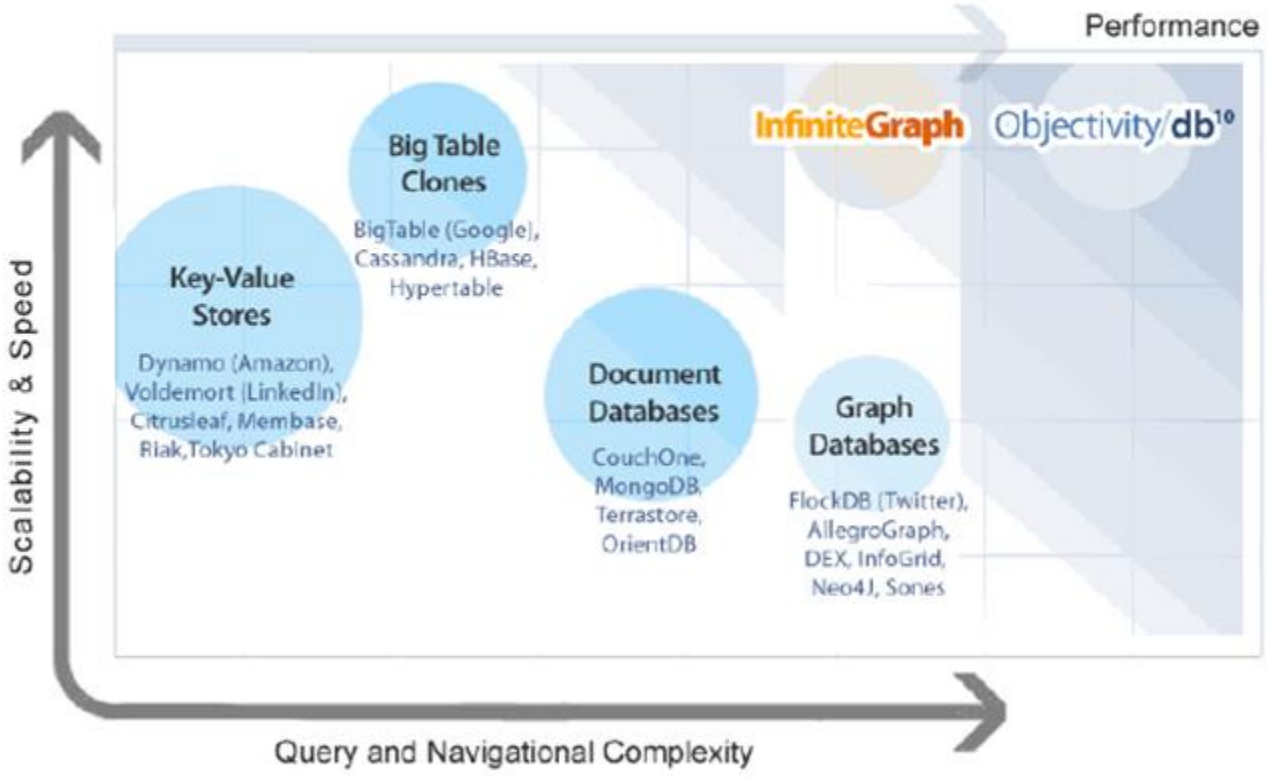
- tuple writes require multiple accesses

=>suitable for read-mostly, read-intensive, large data repositories


Customer 1
Name: Nick
visit
Rating: Good
Little Cesarz
genre
Pizzeria
Rating: Good
visit
friend
Customer 2
Name: Anna
visit
Rating: Good
Restaurante Marisco Sam
genre
Seafood
Rating: Good
friend
visit
genre
Customer 3
Name: Alice
visit
Rating: Good
Mariscos El Pescador

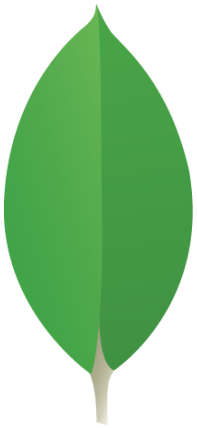

Stop following me, you fucking freaks!
Key-Value
Ordered Key-Value
Big Table
Document, Full-Text Search
Graph
SQL
Key
Value
Column
Time stamp
Value
Key
Column Family
"name" : "Mohana Pilla
"position" : "Delivery
"projects" : [
"name" : "Easy Signu
Semi-Structured Data
Plain Text
a confidential word or numbe
mbination used as a code to
identity when accessing
en 8 and 15 characters
number and may n
paces

The "NOSQL" Technology Landscape



구분	RDBMS	NoSQL
데이터셋 특성	대량, 정형데이터	초대용량, 비정형데이터
적합한 연산	랜덤 액세스, 복잡 연산	순차 액세스, 단순 연산
데이터 모델	중복제거, 정규화	No Join, 비정규화
분산시스템 특성	일관성, 가용성	가용성, 분산처리성
확장성	고성능 서버, SQL 최적화 필요	저가 서버 추가, 시스템이 확장 지원
적합한 서비스	정확성/일관성 중시, 지속적인 update, 정형 데이터	증가량이 큰 대용량 기반, 비정형 데이터 위주 웹서비스

MongoDB



mongoDB®

- Cross-platform 지원
- 문서모델 기반
- high performance, high availability, and easy scalability.
- JSON-like(BSON) documents with schemas

■ 범용 데이터베이스

○ 특정 기능에 초점을 둔 다른 NoSQL과는 다르게, MongoDB는 다양한 목적의 application에 사용 가능한 범용성을 지니고 있음

■ 유연한 Schema 디자인

○ Attribute 속성을 정의하지 않기 때문에, 자유자재로 data를 수정할 수 있음.

■ 풍부한 기능

○ MapReduce, aggregation framework, TTL/capped collections, secondary indexing등의 다양한 기능을 지원하기 때문에 여러 application에 적합

■ 확장성 및 load balancing

○ 자동 sharding 기능을 통한 수평적인 확장으로부터 read, write 확장성을 제공하며, data balancing을 자동적으로 맞추기 때문에 개발에 용이

■ Aggregation 프레임워크

○ Having an extract transform load framework built in the database means that a developer can perform most of the ETL logic before the data leaves the database, eliminating in many cases the need for complex data pipelines.

■ Native replication

○ Data will get replicated across a replica set without complicated setup.

■ 보안 기능

○ Both authentication and authorization are taken into account so that an architect can secure her MongoDB instances.

■ JSON (BSON) 구조 사용

○ JSON is widely used across the web for frontend and API communication and as such it's easier when the database is using the same protocol.

■ Schema-less

○Schema-less로 인해 데이터의 수정 및 확장이 용이

○실제로는 schema를 DBMS 레벨에서 application 레벨로 이동한 것.

○application에서 제대로 된 관리를 하지 못할 경우 DB 처리가 오히려 복잡해짐.

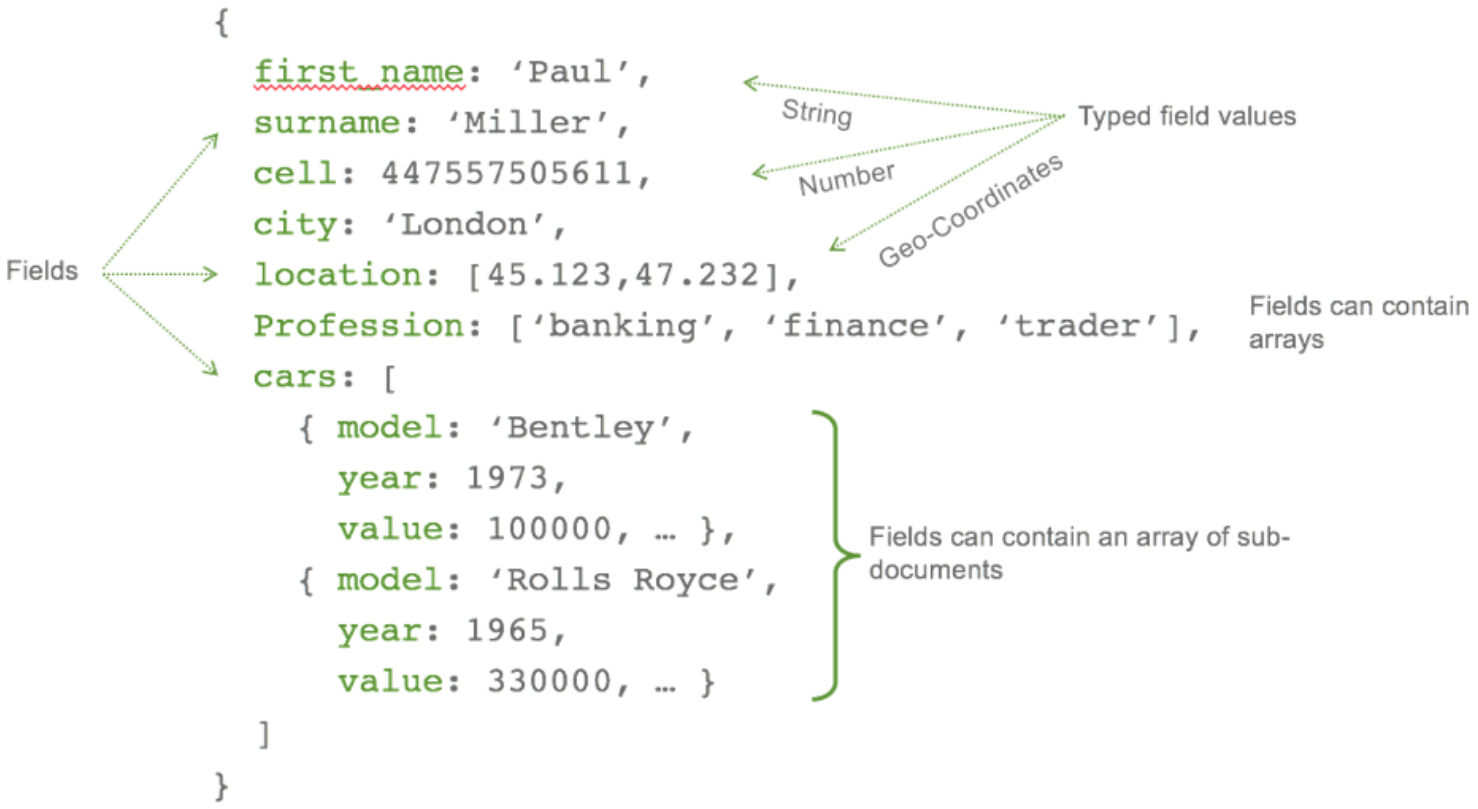
■ ACID 미지원

○ACID를 일부 포기함으로서 수평적 확장성 제공

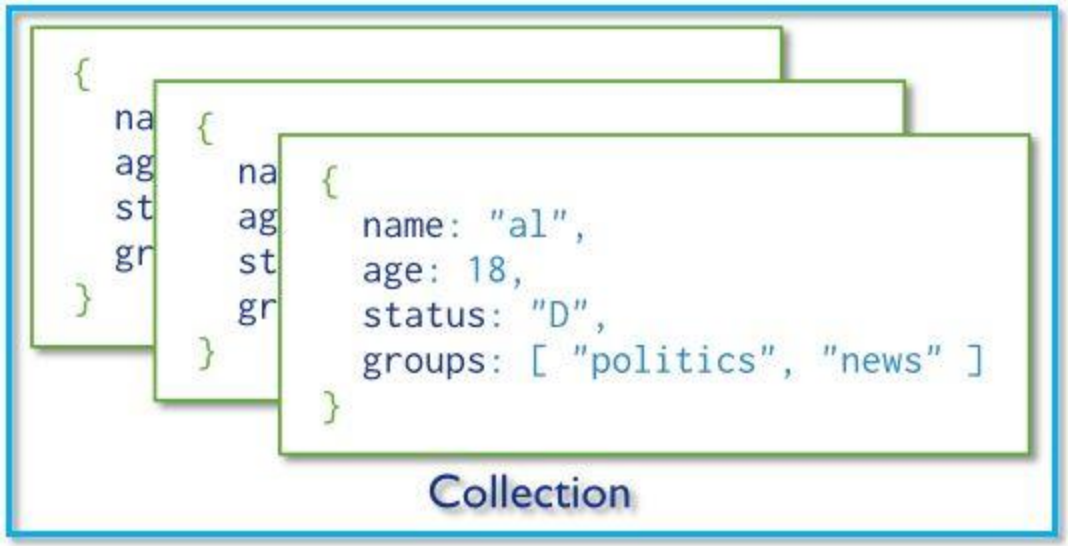
○금융 등의 critical한 분야에서 ACID의 미지원은 치명적으로 작용

○rollback 기능 또한 RDBMS보다 부족하기 때문에 데이터 손실의 위험을 가지고 있음

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo



- RDBMS과 유사하게 field의 이름과 그에 따른 값을 저장
- 데이터의 타입에 대한 별도적 명시 X
- 하나의 field값에 대해서 리스트 형태로 여러 개의 데이터를 관리할 수 있음
- Document 내에 embedded sub-document를 생성하여 관리



■ **Document들의 집합**

■ **Collection 내의 document는 서로 다른 schema를 가질 수 있음**

○ 같은 field 명을 가진 document이지만 데이터의 type이 달라도 문제가 없음

○ 같은 collection 내의 데이터끼리 field명이 다를 수도 있음

○ 같은 collection 내의 document 구조가 달라도 됨

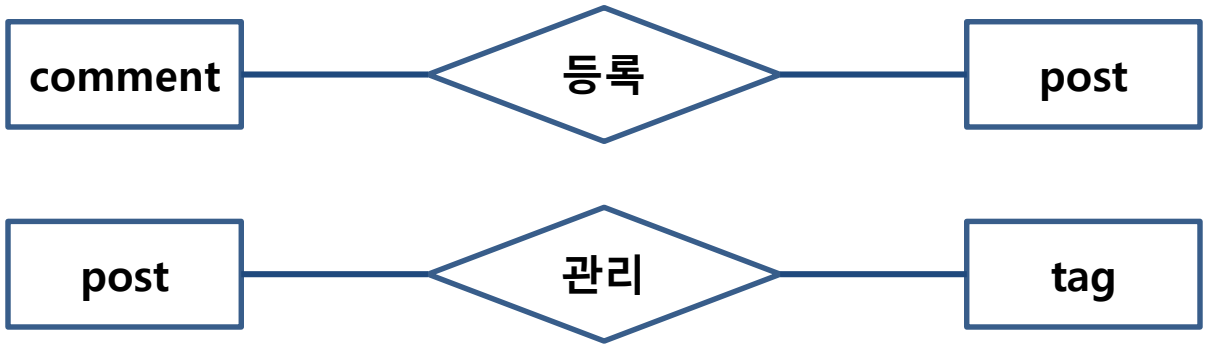
■ 예시 : 블로그 페이지는 다음과 같은 특징을 만족해야 함

- ○ 각각의 post는 제목을 가지고 있어야 하고, post에 대한 내용이 있어야 하며, 작성 시간을 기록해야 하고, post를 가리키는 주소인 url을 가지고 있어야 하며, 추천 횟수를 관리한다.
- ○ 모든 포스트는 한 개 이상의 tag를 가질 수 있다.
- ○ 모든 포스트는 comment를 관리하는데, 각각의 comment는 작성한 message 내용, 작성 시간, 그리고 comment를 추천한 횟수를 가진다.
- ○ 각각의 post에는 0개 이상의 comment를 작성할 수 있다.

■ 관계

○post는 tag를 관리한다.

○comment를 post에 등록한다.



■ Mapping Cardinality

○post는 반드시 comment를 가질 필요는 없다.

○post는 여러 개의 comment를 가질 수 있다.

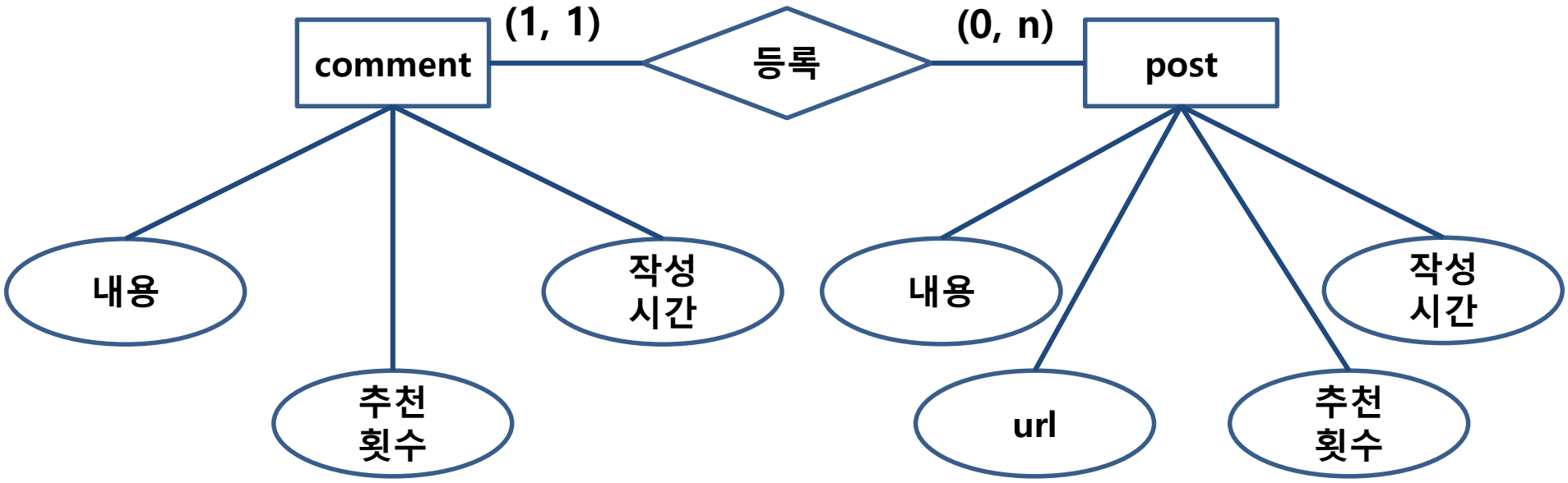
○comment는 반드시 post에 등록되어야 한다.

○comment는 오직 하나의 post에 등록될 수 있다.

■ 속성

○post는 제목과 내용, 작성시간, 추천 횟수, 그리고 url을 가지고 있어야 한다.

○comment는 내용과, 작성시간, 그리고 추천 횟수를 가지고 있어야 한다.




comment
comment_id
*post_id
내용
추천횟수
작성시간
n
1
post
post_id
내용
url
추천횟수
작성시간

■ Mapping Cardinality

○post는 반드시 tag를 관리할 필요는 없다.

○post는 여러 개의 tag를 가질 수 있다.

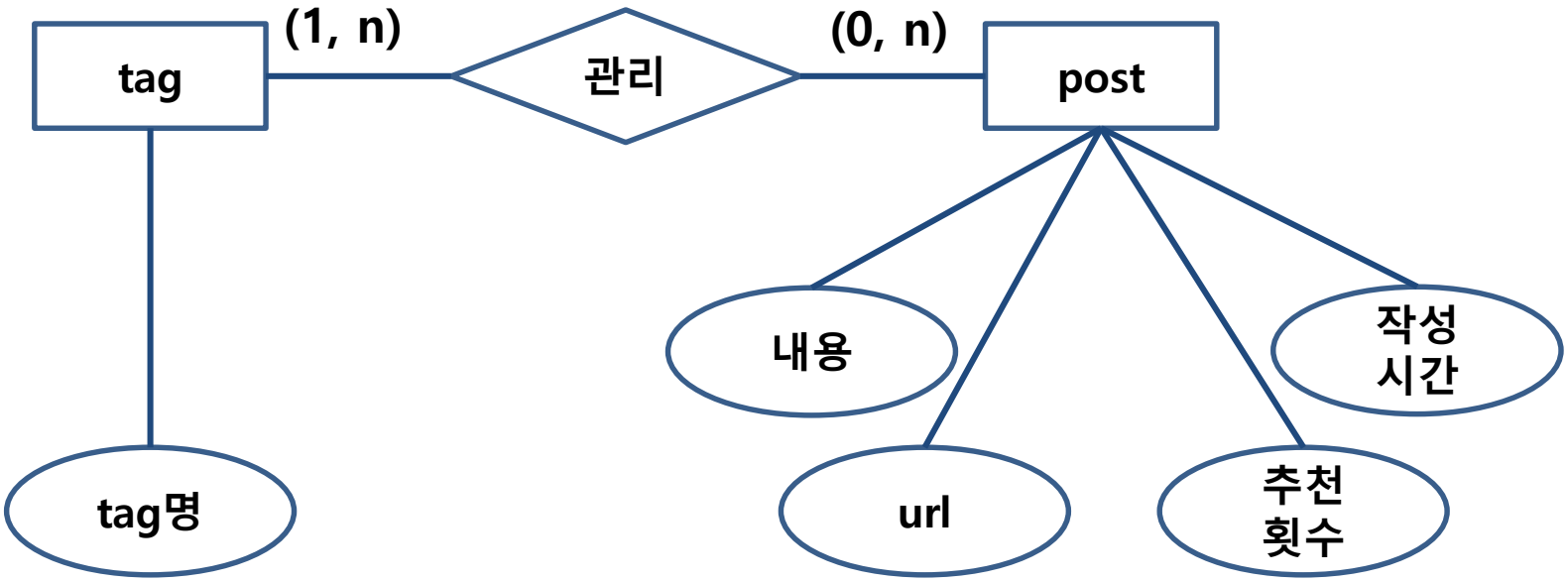
○tag는 반드시 post에 등록되어야 한다.

○tag는 여러 개의 post에 등록될 수 있다.

■ 속성

○post는 제목과 내용, 작성시간, 추천 횟수, 그리고 url을 가지고 있어야 한다.

○tag는 tag 명을 가지고 있어야 한다.




tag
tag_id
tag명
tag 관리
*tag_id
*post_id
post
post_id
내용
url
추천횟수
작성시간

■ RDBMS의 문제

○ 하나의 application이 관리하는 데이터를 여러 table에서 나눠서 관리

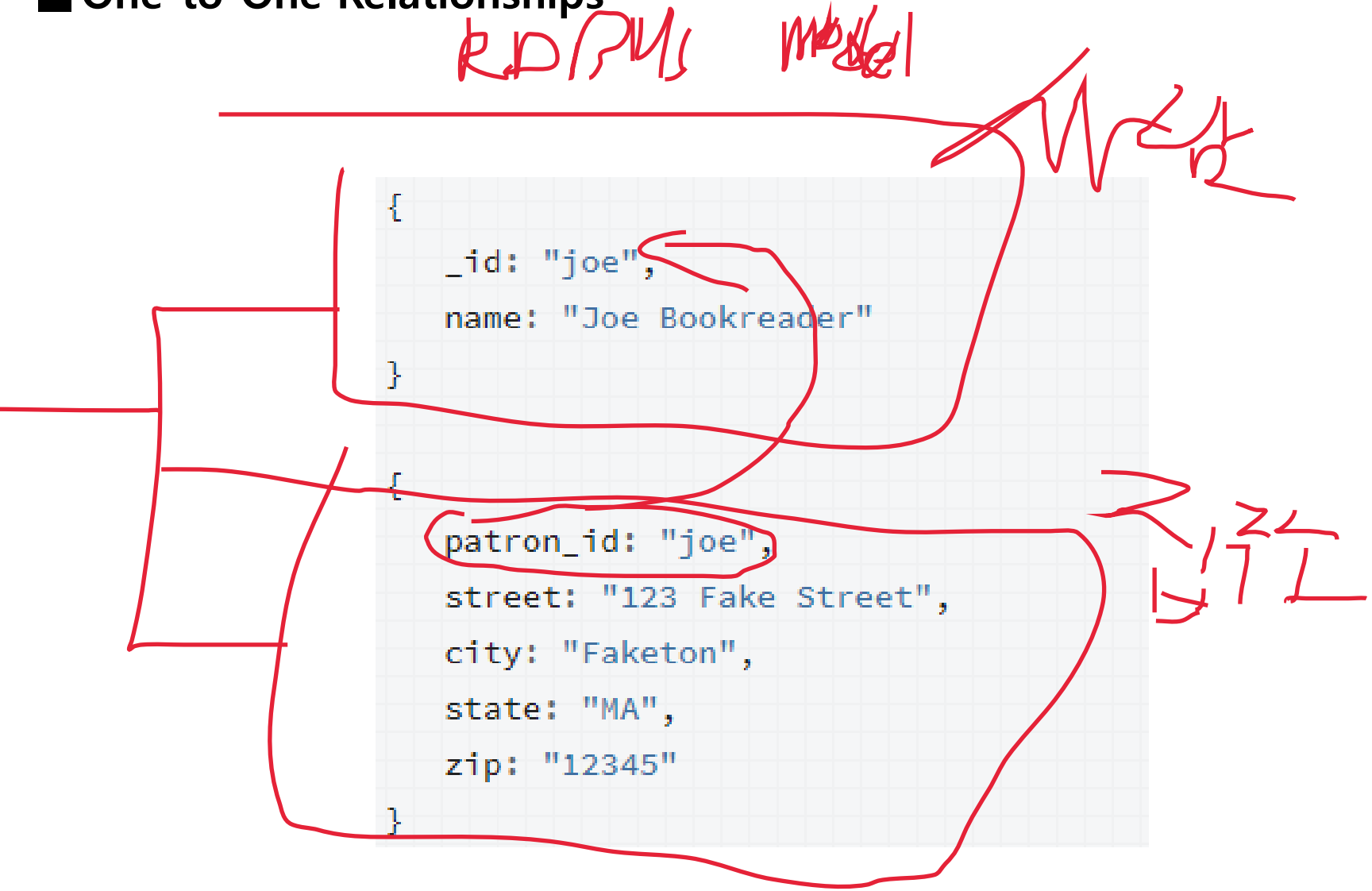
○ 여러 table에서 나눠서 관리하기 때문에 필요한 data를 완성시키기 위해서는 여러 번의 JOIN 연산을 필요로 함.

■ Document 모델의 접근 방식

○ 한 application이 관리하는 데이터는 가능한 하나의 document에서 관리

○ 가능한 다른 document를 reference하기보다는, embedded document로 관리

One-to-One Relationships



```
{
    _id: "joe",
    name: "Joe Bookreader"
}

{
    patron_id: "joe",
    street: "123 Fake Street",
    city: "Faketon",
    state: "MA",
    zip: "12345"
}
```

■ One-to-One Relationships

○ 별도로 document를 관리하지 않고 Embedded Document로 처리

```
{
    _id: "joe",
    name: "Joe Bookreader",
    address: {
                street: "123 Fake Street",
                city: "Faketon",
                state: "MA",
                zip: "12345"
              }
}
```

One-to-Many Relationships

```
{
    _id: "joe",
    name: "Joe Bookreader"
}

{
    patron_id: "joe",
    street: "123 Fake Street",
    city: "Faketon",
    state: "MA",
    zip: "12345"
}

{
    patron_id: "joe",
    street: "1 Some Other Street",
    city: "Boston",
    state: "MA",
    zip: "12345"
}
```

■ One-to-Many Relationships

○마찬가지로 별도의 document로 나누는 것이 아니라 Embedded Document로 처리

```
{
    _id: "joe",
    name: "Joe Bookreader",
    addresses: [
                 {
                   street: "123 Fake Street",
                   city: "Faketon",
                   state: "MA",
                   zip: "12345"
                 },
                 {
                   street: "1 Some Other Street",
                   city: "Boston",
                   state: "MA",
                   zip: "12345"
                 }
               ]
}
```

Many-to-Many Relationships

```
{
    title: "MongoDB: The Definitive Guide",
    author: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    publisher: {
                name: "O'Reilly Media",
                founded: 1980,
                location: "CA"
              }
}

{
    title: "50 Tips and Tricks for MongoDB Developer",
    author: "Kristina Chodorow",
    published_date: ISODate("2011-05-06"),
    pages: 68,
    language: "English",
    publisher: {
                name: "O'Reilly Media",
                founded: 1980,
                location: "CA"
              }
}
```

■ Many-to-Many Relationships

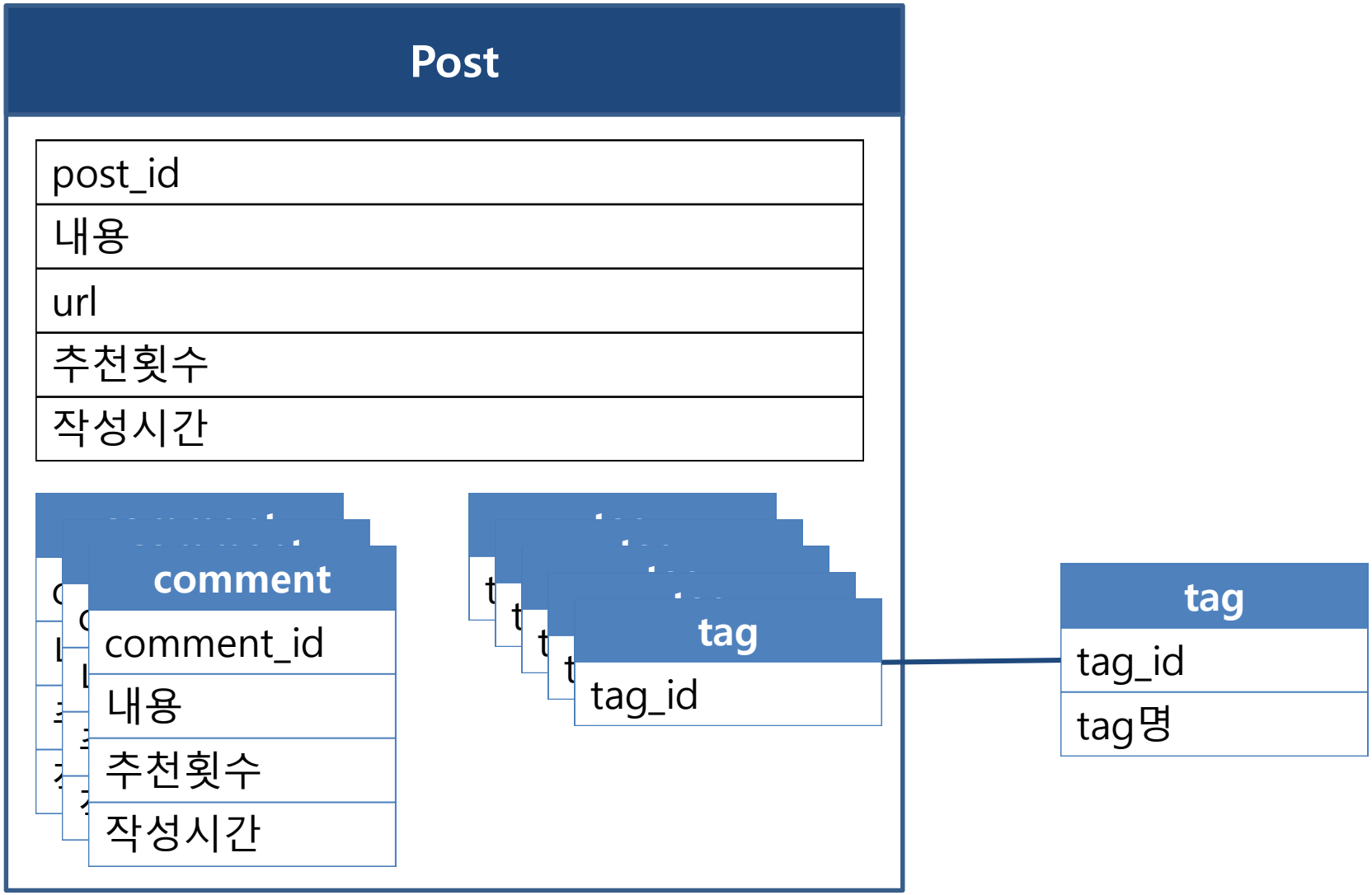
○Many-to-Many에서는 Embedded document로 처리시, 데이터의 중복이 발생

○중복을 방지하기 위해 Document reference로 처리

```
{
    _id: "oreilly",
    name: "O'Reilly Media",
    founded: 1980,
    location: "CA"
}
```

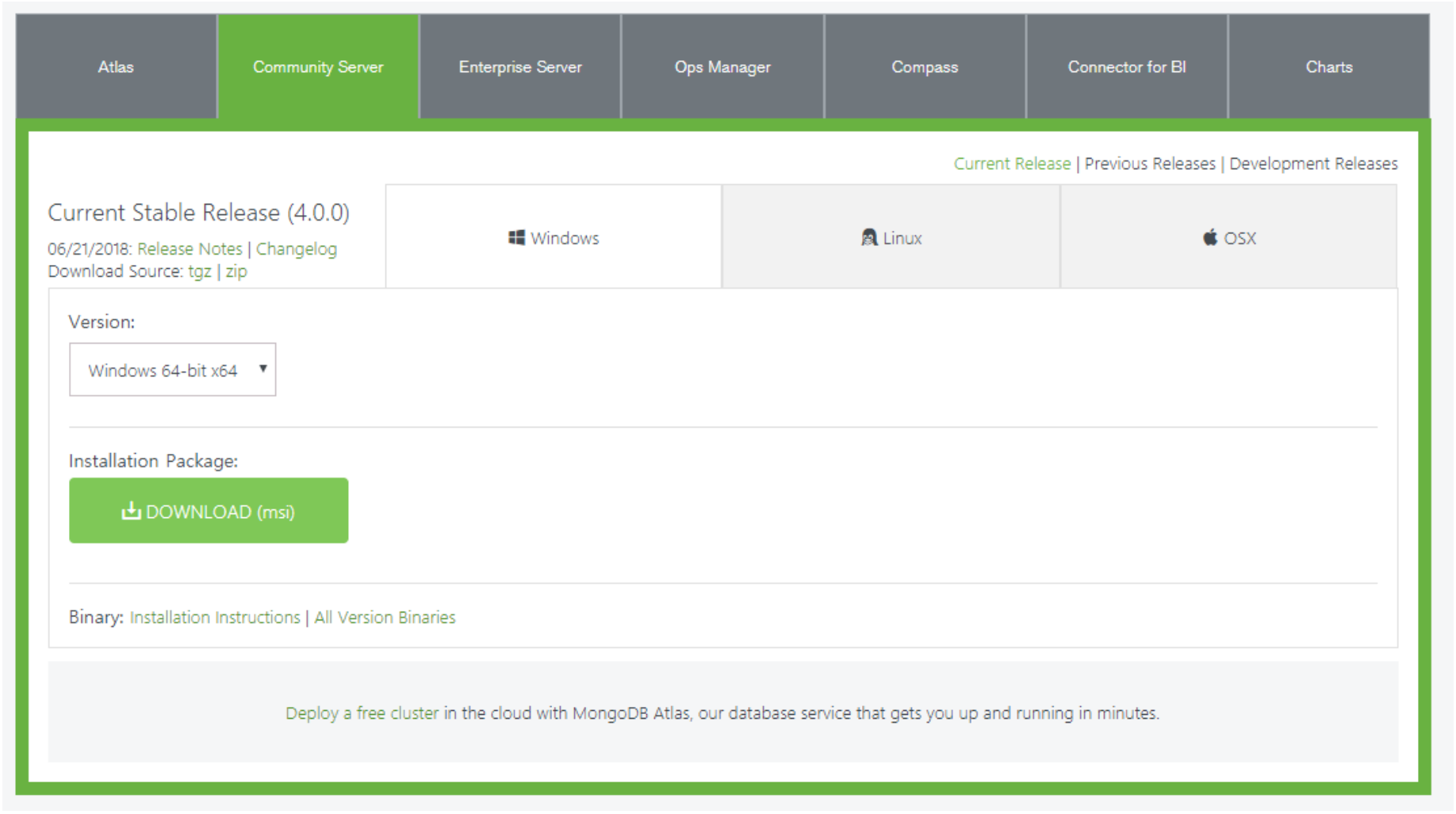
```
{
    _id: 123456789,
    title: "MongoDB: The Definitive Guide",
    author: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    publisher_id: "oreilly"
}
```

```
{
    _id: 234567890,
    title: "50 Tips and Tricks for MongoDB Developer",
    author: "Kristina Chodorow",
    published_date: ISODate("2011-05-06"),
    pages: 68,
    language: "English",
    publisher_id: "oreilly"
}
```


Post
post_id
내용
url
추천횟수
작성시간
comment
comment_id
내용
추천횟수
작성시간
tag
tag_id
tag
tag_id
tag명

MongoDB Basic

■ **https://www.mongodb.com/download-center?jmp=homepage#community**



■ mongod [인자]

■ 주요 인자

○ --bind_ip "IP 주소"(기본 localhost)

○ --dbpath "데이터베이스 저장 위치"(기본 현재위치₩data₩db)

○ --port 포트 번호 지정(기본 27017)

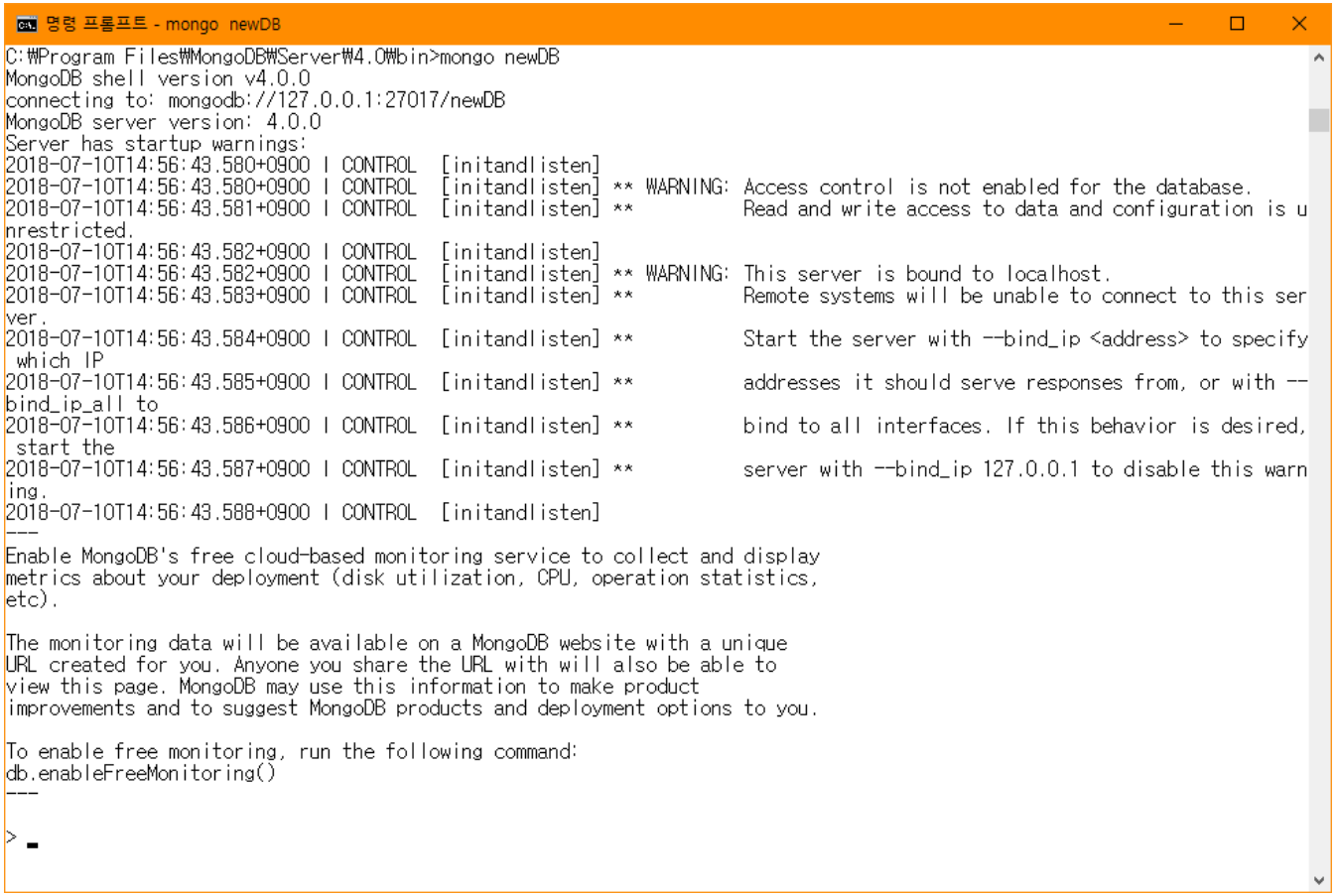
○ --auth 서버에 접근 시 인증 필요

■ mongo [인자] [DB명]

■ 주요 인자

○ --host "IP 주소"

○ --port 포트 번호 지정(기본 27017)



■ show databases (show dbs)

```
> show databases
admin   0.000GB
config  0.000GB
local   0.000GB
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

■ use “DB명”

○ DB가 없을 경우, 새로운 DB를 생성하면서 해당 DB를 선택

○ DB가 있을 경우, 해당 DB를 선택

```
> use newDB
switched to db newDB
> db
newDB
```

■ db.dropDatabase()

○ 현재 선택중인 Database를 제거

```
> use tempDB
switched to db tempDB
> db
tempDB
> db.dropDatabase()
{ "ok" : 1 }
```

■ show roles

○ 사용 가능한 권한 출력

○ DB에 따라 설정 가능한 권한이 다름

```
> use admin
switched to db admin
> show roles
{
        "role" : "__queryableBackup",
        "db" : "admin",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "__system",
        "db" : "admin",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "backup",
        "db" : "admin",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "clusterAdmin",
        "db" : "admin",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
```

```
> use newDB
switched to db newDB
> show roles
{
        "role" : "dbAdmin",
        "db" : "newDB",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "dbOwner",
        "db" : "newDB",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "enableSharding",
        "db" : "newDB",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "read",
        "db" : "newDB",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
```

■ db.createUser({
user: “사용자이름”,
pwd: “비밀번호”,
roles: [{role: ‘역할’, db: ‘관리 DB’}, {role: ‘역할’, db: ‘관리 DB’}, …]
});

```
> use newDB
switched to db newDB
> show users
> db.createUser( { user: "newadmin",
...              pwd: "newadmin",
...              roles: ["dbAdmin",
...                      "readWrite"] } )
Successfully added user: { "user" : "newadmin", "roles" : [ "dbAdmin", "readWrite" ] }
```

■ mongo –u ‘ID’ –p ‘비밀번호’ –authenticationDatabase “접속 DB명”

```
C:₩Program Files₩MongoDB₩Server₩4.0₩bin>mongo -u "newadmin" -p "newadmin" --authenticationDatabase "newDB"
MongoDB shell version v4.0.0
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 4.0.0
> show dbs
newDB  0.000GB
> use admin
switched to db admin
> show users
2018-07-11T01:37:24.973+0900 E QUERY    [js] Error: not authorized on admin to execute command { usersInfo: 1.0, $db: "admin" } :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
DB.prototype.getUsers@src/mongo/shell/db.js:1757:1
shellHelper.show@src/mongo/shell/utils.js:848:9
shellHelper@src/mongo/shell/utils.js:755:15
@(shellhelp2):1:1
> use newDB
switched to db newDB
> show users
2018-07-11T01:37:37.225+0900 E QUERY    [js] Error: not authorized on newDB to execute command { usersInfo: 1.0, $db: "newDB" } :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
DB.prototype.getUsers@src/mongo/shell/db.js:1757:1
shellHelper.show@src/mongo/shell/utils.js:848:9
shellHelper@src/mongo/shell/utils.js:755:15
@(shellhelp2):1:1
> db.item.find();
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd851"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd852"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd853"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
{ "_id" : ObjectId("5b44cba275e536b8f25fd854"), "name" : "New item 9", "price" : 55000, "ratio" : NumberDecimal("3.70000000000000") }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd855"), "name" : "New item 0", "price" : 75908.9534522125, "ratio" : 1.0602640630415583 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd856"), "name" : "New item 1", "price" : 1132.514621706926, "ratio" : 4.405430158902593 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd857"), "name" : "New item 2", "price" : 16258.006358517585, "ratio" : 0.6734067234422486 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd858"), "name" : "New item 3", "price" : 80015.79927329937, "ratio" : 3.3353775161357886 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd859"), "name" : "New item 4", "price" : 60853.369399334726, "ratio" : 3.8337186228394557 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd85a"), "name" : "New item 5", "price" : 52208.11113964458, "ratio" : 1.5374085725804014 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd85b"), "name" : "New item 6", "price" : 58706.9125257064, "ratio" : 3.0698360582663438 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd85c"), "name" : "New item 7", "price" : 97939.81377485346, "ratio" : 0.4053242624493886 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd85d"), "name" : "New item 8", "price" : 90149.41812108323, "ratio" : 2.8955203316480445 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd85e"), "name" : "New item 9", "price" : 98283.54589113563, "ratio" : 2.0452123439410186 }
{ "_id" : ObjectId("5b44cfde75e536b8f25fd85f"), "name" : "New item 10", "price" : 78899.52625133448, "ratio" : 2.6426361655636144 }
Type "it" for more
```

■ **db.createCollection('collection명', [인자])**

■ **주요 인자**

○ capped : 크기가 고정된 collection으로 생성, 용량이 초과될 경우 가장 오래된 데이터를 덮어씀

○ size : collection의 크기를 bytes 단위로 지정

○ max : collection에 저장할 최대 document의 개수

```
> db.createCollection('user', {max: 1000});
{ "ok" : 1 }
```

■ **db. 'collection명'.drop()**

```
> db.user.drop()
true
```

MongoDB CRUD

Create

■ **db.'collection 명'.save({dictionary 형태 데이터});**
■ **db.'collection 명'.insert({dictionary 형태 데이터});**

○ 만약 해당 collection이 없을 경우 자동적으로 collection을 생성

```
> db.item.save({name: "New item 1", price: 100000, ratio: 4.5});
WriteResult({ "nInserted" : 1 })
> show collections
item

> db.item.insert({name: "New item 2", price: 30000, ratio: 4.2});
WriteResult({ "nInserted" : 1 })
```

■ 리스트 형태로 한 번에 여러 document를 생성 가능

```
> db.item.save([{name: "New item 3", price: 70000, ratio: 3.8},
... {name: "New item 4", price: 45000, ratio: 3.6}]);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

■ 이미 존재하는 key값에 새롭게 document를 작성하려고 할 때

○ insert : 이미 존재하는 키 값이기 때문에 새로운 document를 삽입하지 않음

○ save : 기존에 존재하는 키 값의 document를 덮어쓰기 함

```
> db.item.insert({"_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 80000, "ratio" : 4.6 });
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 11000,
                "errmsg" : "E11000 duplicate key error collection: newDB.item index: _id_ dup key: { : ObjectId('5b44507e3ce931baad431f
45') }"
        }
})
> db.item.save({"_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 80000, "ratio" : 4.6 });
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 80000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ **db.'collection 명'.insertOne({dictionary 형태 데이터});**
■ **db.'collection 명'.insertMany(리스트);**

○ Insert는 특정 ODM에서는 **deprecate된** 명령어이기 때문에 insertOne 또는 insertMany 사용을 권장

```
> db.item.insertOne({name: "New item 5", price: 20000, ratio: 2.2});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5b4456fa3ce931baad431f54")
}
> db.item.insertMany([{name: "New item 6", price: 30000, ratio: 2.8},
... {name: "New item 7", price: 230000, ratio: 5}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5b4457623ce931baad431f55"),
                ObjectId("5b4457623ce931baad431f56")
        ]
}
```

■ BSON의 Data Types과 동일

Type	Number	Alias	Notes	Type	Number	Alias	Notes
Double	1	"double"		DBPointer	12	"dbPointer"	Deprecated.
String	2	"string"		JavaScript	13	"javascript"	
Object	3	"object"		Symbol	14	"symbol"	Deprecated.
Array	4	"array"		JavaScript (with scope)	15	"javascriptWithScope"	
Binary data	5	"binData"		32-bit integer	16	"int"	
Undefined	6	"undefined"	Deprecated.	Timestamp	17	"timestamp"	
ObjectId	7	"objectId"		64-bit integer	18	"long"	
Boolean	8	"bool"		Decimal128	19	"decimal"	New in version 3.4.
Date	9	"date"		Min key	-1	"minKey"	
Null	10	"null"		Max key	127	"maxKey"	
Regular Expression	11	"regex"					

■ Date();

○ 현재 시간을 string값으로 반환

```
> Date();
Tue Jul 10 2018 21:34:12 GMT+0900
> typeof Date();
string
> Date() instanceof Date;
false
```

■ ISODate();

○ 현재 시간을 object type으로 반환

```
> ISODate();
ISODate("2018-07-10T12:35:43.451Z")
> typeof ISODate();
object
> ISODate() instanceof Date;
true
```

■ ObjectId();

○ Document의 key값으로 사용되는 data type

○ 12 바이트의 hexadecimal로 id값을 생성 가능

○ 지정하지 않을 경우 index값을 순차적으로 생성

```
> ObjectId("112233445566778899aabbcc")
ObjectId("112233445566778899aabbcc")
> ObjectId();
ObjectId("5b44a9d53ce931baad431f5b")
> ObjectId();
ObjectId("5b44aa263ce931baad431f5c")
> ObjectId();
ObjectId("5b44aa263ce931baad431f5d")
> ObjectId();
ObjectId("5b44aa273ce931baad431f5e")
> typeof ObjectId();
object
> ObjectId() instanceof ObjectId;
true
```

■ MongoDB에서 별도로 지정하지 않을 경우, 모든 숫자는 Double type으로 설정

■ NumberInt

○ 32비트형 정수 data type

```
> NumberInt(100);
NumberInt(100)
> typeof NumberInt(100);
object
> NumberInt(100) instanceof NumberInt
true
> 100 instanceof NumberInt
false
```

■ NumberLong

○ 64비트형 정수 data type

```
> NumberLong(10000000000000);
NumberLong("10000000000000")
> typeof NumberLong(100000000000);
object
> NumberLong(100000000000) instanceof NumberLong;
true
> 10000000 instanceof NumberLong;
false
```

■ NumberDecimal

○ 128비트형 실수 data type

```
> NumberDecimal(10.01);
NumberDecimal("10.0100000000000")
> typeof NumberDecimal(10.01);
object
> NumberDecimal(10.01) instanceof NumberDecimal;
true
```

Read

■ db.'collection 명'.find({query});

○ 아무것도 입력하지 않을 경우 collection 내의 모든 값을 출력

○ And 연산자처럼, 모든 조건을 만족하는 데이터 출력

```
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 100000, "ratio" : 4.5 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }

> db.item.find({name: "New item 1"});
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 100000, "ratio" : 4.5 }
> db.item.find({name: "New item 1", price: 40000});
>
```

■ db.'collection 명'.find({query}).pretty()

○ 출력을 보기 쉽게 깔끔하게 보정

```
> db.item.find().pretty()
{
        "_id" : ObjectId("5b44507e3ce931baad431f45"),
        "name" : "New item 1",
        "price" : 100000,
        "ratio" : 4.5
}
{
        "_id" : ObjectId("5b44547c3ce931baad431f50"),
        "name" : "New item 2",
        "price" : 30000,
        "ratio" : 4.2
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f52"),
        "name" : "New item 3",
        "price" : 70000,
        "ratio" : 3.8
}
```

■ 비교 연산

operator	설명
$eq	(equals) 주어진 값과 일치하는 값
$gt	(greater than) 주어진 값보다 큰 값
$gte	(greather than or equals) 주어진 값보다 크거나 같은 값
$lt	(less than) 주어진 값보다 작은 값
$lte	(less than or equals) 주어진 값보다 작거나 같은 값
$ne	(not equal) 주어진 값과 일치하지 않는 값
$in	주어진 배열 안에 속하는 값
$nin	주어빈 배열 안에 속하지 않는 값

```
> db.item.find({price: {$gte: 40000}}).pretty();
{
        "_id" : ObjectId("5b44507e3ce931baad431f45"),
        "name" : "New item 1",
        "price" : 100000,
        "ratio" : 4.5
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f52"),
        "name" : "New item 3",
        "price" : 70000,
        "ratio" : 3.8
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f53"),
        "name" : "New item 4",
        "price" : 45000,
        "ratio" : 3.6
}
{
        "_id" : ObjectId("5b4457623ce931baad431f56"),
        "name" : "New item 7",
        "price" : 230000,
        "ratio" : 5
}
```

```
> db.item.find({name: {$in: ["New item 1",
... "New item 3", "New item 7"]}}).pretty();
{
        "_id" : ObjectId("5b44507e3ce931baad431f45"),
        "name" : "New item 1",
        "price" : 100000,
        "ratio" : 4.5
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f52"),
        "name" : "New item 3",
        "price" : 70000,
        "ratio" : 3.8
}
{
        "_id" : ObjectId("5b4457623ce931baad431f56"),
        "name" : "New item 7",
        "price" : 230000,
        "ratio" : 5
}
```

■ 논리 연산

operator	설명
$or	주어진 조건중 하나라도 true 일 때 true
$and	주어진 모든 조건이 true 일 때 true
$not	주어진 조건이 false 일 때 true
$nor	주어진 모든 조건이 false 일 때 true

```
> db.item.find({$or: [{price: 45000},
... {name: "New item 7"}]}).pretty();
{
        "_id" : ObjectId("5b4454df3ce931baad431f53"),
        "name" : "New item 4",
        "price" : 45000,
        "ratio" : 3.6
}
{
        "_id" : ObjectId("5b4457623ce931baad431f56"),
        "name" : "New item 7",
        "price" : 230000,
        "ratio" : 5
}
```

```
> db.item.find({price: {$not: {$eq: 230000}}}).pretty();
{
        "_id" : ObjectId("5b44507e3ce931baad431f45"),
        "name" : "New item 1",
        "price" : 100000,
        "ratio" : 4.5
}
{
        "_id" : ObjectId("5b44547c3ce931baad431f50"),
        "name" : "New item 2",
        "price" : 30000,
        "ratio" : 4.2
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f52"),
        "name" : "New item 3",
        "price" : 70000,
        "ratio" : 3.8
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f53"),
        "name" : "New item 4",
        "price" : 45000,
        "ratio" : 3.6
}
{
        "_id" : ObjectId("5b4456fa3ce931baad431f54"),
        "name" : "New item 5",
        "price" : 20000,
        "ratio" : 2.2
}
```

■ $exists: true/false

○ 해당 field의 존재하는지 여부를 확인

```
> db.item.find({name: {$exists: true}}).pretty();
{
        "_id" : ObjectId("5b44507e3ce931baad431f45"),
        "name" : "New item 1",
        "price" : 80000,
        "ratio" : 4.6
}
{
        "_id" : ObjectId("5b44547c3ce931baad431f50"),
        "name" : "New item 2",
        "price" : 30000,
        "ratio" : 4.2
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f52"),
        "name" : "New item 3",
        "price" : 70000,
        "ratio" : 3.8
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f53"),
        "name" : "New item 4",
        "price" : 45000,
        "ratio" : 3.6
}
{
        "_id" : ObjectId("5b4456fa3ce931baad431f54"),
        "name" : "New item 5",
        "price" : 20000,
        "ratio" : 2.2
}
```

■ $type

○ 해당 file의 자료형이 일치하는 document만 선택

```
> db.item.find({name: {$type: 'bool'}}).pretty();
> db.item.find({price: {$type: 1}}).pretty();
{
        "_id" : ObjectId("5b44507e3ce931baad431f45"),
        "name" : "New item 1",
        "price" : 80000,
        "ratio" : 4.6
}
{
        "_id" : ObjectId("5b44547c3ce931baad431f50"),
        "name" : "New item 2",
        "price" : 30000,
        "ratio" : 4.2
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f52"),
        "name" : "New item 3",
        "price" : 70000,
        "ratio" : 3.8
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f53"),
        "name" : "New item 4",
        "price" : 45000,
        "ratio" : 3.6
}
```

■ More Example

```
> db.typetest.insert([{ "_id" : 1, "val" : NumberDecimal( "9.99" ), "description" : "Decimal" },
...  { "_id" : 2, "val" : 9.99, "description" : "Double" },
... { "_id" : 3, "val" : 10, "description" : "Double" },
... { "_id" : 4, "val" : NumberLong(10), "description" : "Long" },
... { "_id" : 5, "val" : NumberDecimal( "10.0" ), "description" : "Decimal" }]);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.typetest.find();
{ "_id" : 1, "val" : NumberDecimal("9.99"), "description" : "Decimal" }
{ "_id" : 2, "val" : 9.99, "description" : "Double" }
{ "_id" : 3, "val" : 10, "description" : "Double" }
{ "_id" : 4, "val" : NumberLong(10), "description" : "Long" }
{ "_id" : 5, "val" : NumberDecimal("10.0"), "description" : "Decimal" }
> db.typetest.find({val: {$type: 'double'}});
{ "_id" : 2, "val" : 9.99, "description" : "Double" }
{ "_id" : 3, "val" : 10, "description" : "Double" }
> db.typetest.find({val: {$type: 'long'}});
{ "_id" : 4, "val" : NumberLong(10), "description" : "Long" }
> db.typetest.find({val: {$type: 'decimal'}});
{ "_id" : 1, "val" : NumberDecimal("9.99"), "description" : "Decimal" }
{ "_id" : 5, "val" : NumberDecimal("10.0"), "description" : "Decimal" }
```

■ Another Example

```
> db.typetest.find();
{ "_id" : 1, "val" : NumberDecimal("9.99"), "description" : "Decimal" }
{ "_id" : 2, "val" : 9.99, "description" : "Double" }
{ "_id" : 3, "val" : 10, "description" : "Double" }
{ "_id" : 4, "val" : NumberLong(10), "description" : "Long" }
{ "_id" : 5, "val" : NumberDecimal("10.0"), "description" : "Decimal" }
> db.typetest.find({val: 9.99});
{ "_id" : 2, "val" : 9.99, "description" : "Double" }
> db.typetest.find({val: NumberDecimal(9.99)});
{ "_id" : 1, "val" : NumberDecimal("9.99"), "description" : "Decimal" }
> db.typetest.find({val: 10});
{ "_id" : 3, "val" : 10, "description" : "Double" }
{ "_id" : 4, "val" : NumberLong(10), "description" : "Long" }
{ "_id" : 5, "val" : NumberDecimal("10.0"), "description" : "Decimal" }
> db.typetest.find({val: NumberDecimal(10)});
{ "_id" : 3, "val" : 10, "description" : "Double" }
{ "_id" : 4, "val" : NumberLong(10), "description" : "Long" }
{ "_id" : 5, "val" : NumberDecimal("10.0"), "description" : "Decimal" }
> db.typetest.find({val: NumberInt(10)});
{ "_id" : 3, "val" : 10, "description" : "Double" }
{ "_id" : 4, "val" : NumberLong(10), "description" : "Long" }
{ "_id" : 5, "val" : NumberDecimal("10.0"), "description" : "Decimal" }
```

■ 정수 꼴의 data는 type이 달라도 값으로 find 가능

■ $regex 연산

○ 정규식을 이용하여 Document 검색 가능

○ { <field>: { $regex: /pattern/} }

○ { <field>: { $regex: 'pattern' } }

○ { <field>: /pattern/}

```
> db.item.find({name: {$regex: /New item [1-4]/}}).pretty();
{
        "_id" : ObjectId("5b44507e3ce931baad431f45"),
        "name" : "New item 1",
        "price" : 100000,
        "ratio" : 4.5
}
{
        "_id" : ObjectId("5b44547c3ce931baad431f50"),
        "name" : "New item 2",
        "price" : 30000,
        "ratio" : 4.2
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f52"),
        "name" : "New item 3",
        "price" : 70000,
        "ratio" : 3.8
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f53"),
        "name" : "New item 4",
        "price" : 45000,
        "ratio" : 3.6
}
```

```
> db.item.find({name:/New item [1,3,7]/}).pretty();
{
        "_id" : ObjectId("5b44507e3ce931baad431f45"),
        "name" : "New item 1",
        "price" : 100000,
        "ratio" : 4.5
}
{
        "_id" : ObjectId("5b4454df3ce931baad431f52"),
        "name" : "New item 3",
        "price" : 70000,
        "ratio" : 3.8
}
{
        "_id" : ObjectId("5b4457623ce931baad431f56"),
        "name" : "New item 7",
        "price" : 230000,
        "ratio" : 5
}
```

■ 특정한 field를 선택하여 출력

```
> db.item.find({}, {"_id": false, "name": true, "price": true, "ratio": true})
{ "name" : "New item 1", "price" : 100000, "ratio" : 4.5 }
{ "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "name" : "New item 7", "price" : 230000, "ratio" : 5 }
> db.item.find({}, {"_id": false, "name": true, "price": true})
{ "name" : "New item 1", "price" : 100000 }
{ "name" : "New item 2", "price" : 30000 }
{ "name" : "New item 3", "price" : 70000 }
{ "name" : "New item 4", "price" : 45000 }
{ "name" : "New item 5", "price" : 20000 }
{ "name" : "New item 6", "price" : 30000 }
{ "name" : "New item 7", "price" : 230000 }
> db.item.find({}, {"name": true})
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1" }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2" }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3" }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4" }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5" }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6" }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7" }
```

■ sort()

○ 선택한 field에 대해서 정렬하여 출력

○ 1 : 오름차순, -1 : 내림차순

```
> db.item.find();
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd851"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd852"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd853"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
> db.item.find().sort({price: 1});
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd851"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd853"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd852"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
> db.item.find().sort({price: 1, name: -1});
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd851"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd853"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd852"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
```

■ limit()

- ○ 선택한 document의 출력 개수에 제한을 둠

```
> db.item.find();
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd851"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd852"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd853"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
> db.item.find().limit(4);
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
```

■ skip()

○ 선택한 document의 head를 건너뛰고 출력

```
> db.item.find();
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd851"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd852"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd853"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
> db.item.find().skip(3);
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd851"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd852"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd853"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ count()

○ 선택한 document의 개수 확인

```
> db.item.find();
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd851"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd852"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd853"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
> db.item.find({name: /New item [1-5]/});
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
> db.item.find({name: /New item [1-5]/}).count()
5
```

Update

■ **db.'collection 명'.update({query}, {update값}, {인자});**

■ **주요 인자**

○ upsert : true일 경우, query에 해당하는 문서가 없을 경우, 새롭게 document 생성

○ multi : true일 경우, 여러 개의 document를 수정

■ **document의 내용이 update값 자체로 변한다는 점에 주의**

```
> db.item.update({name: "New item 1"}, {price: 90000, ratio: 4.6})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.item.find({name: "New item 1"})
> db.item.find()
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "price" : 90000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
```

```
> db.item.update({price: 60000}, {name: "New item 8", price: 60000, ratio: 4.1}, {upsert: true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60")
})
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 90000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ $set

○ 특정 field만 수정

```
> db.item.update({name: "New item 1"}, {$set : {price: 85000}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.item.find()
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 85000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ $unset

○ 특정 field 제거

```
> db.item.update({name: "New item 1"}, {$unset : {price: 0}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.item.find()
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ $inc

○ 특정 field의 값을 증감

```
> db.item.update({name: "New item 1"}, {$inc: {price: 1000}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.item.find()
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 101000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
> db.item.update({name: "New item 1"}, {$inc: {price: -2000}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.item.find()
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 99000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ $mul

○ 특정 field의 값에 곱

```
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 80000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
> db.item.update({name: /New item[1-8]/}, {$mul: {ratio: 0.5}});
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 80000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
> db.item.update({name: /New item [1-8]/}, {$mul: {ratio: 0.5}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 80000, "ratio" : 2.3 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ $rename

○ field의 이름을 새롭게 교체

```
> db.item.update({name: /New item [1-4]/},
... {$rename : {name: "item", ratio: "star"}}, {multi: true});
WriteResult({ "nMatched" : 4, "nUpserted" : 0, "nModified" : 3 })
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "price" : 80000, "item" : "New item 1", "star" : 2.3 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "price" : 30000, "item" : "New item 2", "star" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "price" : 70000, "item" : "New item 3", "star" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "price" : 45000, "item" : "New item 4", "star" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ $min

○ Field 값이 주어진 값보다 클 경우 새 값으로 교체

```
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "price" : 80000, "item" : "New item 1", "star" : 2.3 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "price" : 30000, "item" : "New item 2", "star" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "price" : 70000, "item" : "New item 3", "star" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "price" : 45000, "item" : "New item 4", "star" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
> db.item.update({},
... {$min : {price: 65000}}, {multi: true});
WriteResult({ "nMatched" : 8, "nUpserted" : 0, "nModified" : 3 })
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "price" : 65000, "item" : "New item 1", "star" : 2.3 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "price" : 30000, "item" : "New item 2", "star" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "price" : 65000, "item" : "New item 3", "star" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "price" : 45000, "item" : "New item 4", "star" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 65000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ $max

○ Field 값이 주어진 값보다 작을 경우 새 값으로 교체

```
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "price" : 65000, "item" : "New item 1", "star" : 2.3 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "price" : 30000, "item" : "New item 2", "star" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "price" : 65000, "item" : "New item 3", "star" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "price" : 45000, "item" : "New item 4", "star" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 65000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
> db.item.update({},
... {$max : {price: 35000}}, {multi: true});
WriteResult({ "nMatched" : 8, "nUpserted" : 0, "nModified" : 3 })
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "price" : 65000, "item" : "New item 1", "star" : 2.3 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "price" : 35000, "item" : "New item 2", "star" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "price" : 65000, "item" : "New item 3", "star" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "price" : 45000, "item" : "New item 4", "star" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 35000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 35000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 65000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ db.'collection 명'.updateOne()

○ 검색된 항목 중 최상위 항목만 수정

○ update의 multi를 false로 설정한 것과 동일하게 작동

■ db.'collection 명'.updateMulti()

○ 검색된 항목 모두 수정

○ update의 multi를 true로 설정한 것과 동일하게 작동

■ db.'collection 명'.replaceOne()

○ 검색된 항목을 주어진 값으로 수정

○ update에서 $set이나 $inc 인자 없이 교체하는 것과 동일하게 작동

Delete

■ db.'collection 명'.remove({query}, {인자})

■ 주요 인자

○ justone : query 조건을 만족하는 document가 여러 개일 경우 하나만 제거

```
> db.item.remove({name: /New item [5-8]/})
WriteResult({ "nRemoved" : 4 })
> db.item.find();
{ "_id" : ObjectId("5b44507e3ce931baad431f45"), "name" : "New item 1", "price" : 99000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
```

■ db.'collection 명'.deleteOne({query})

○ query를 만족하는 document가 여러 개일 경우 하나만 삭제

○ justone을 설정했을 때와 동일하게 작동

```
> db.item.deleteOne({name: /New item [1-4]/});
{ "acknowledged" : true, "deletedCount" : 1 }
> db.item.find();
{ "_id" : ObjectId("5b44547c3ce931baad431f50"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b4454df3ce931baad431f52"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b4454df3ce931baad431f53"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

■ db.'collection 명'.deleteMany({query})

○ query를 만족하는 document가 여러 개일 경우 전부 삭제

○ remove와 동일하게 가동

```
> db.item.deleteMany({name: /New item [1-4]/});
{ "acknowledged" : true, "deletedCount" : 3 }
> db.item.find();
{ "_id" : ObjectId("5b4456fa3ce931baad431f54"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b4457623ce931baad431f55"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b4457623ce931baad431f56"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b4471fa5ebf0f0cb068cb60"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
```

Shell Script

■ JavaScript 포맷의 script를 shell상에서 직접 실행 가능

```
> var item_9_name = "New item 9"
> var item_9_price = NumberInt(55000)
> var item_9_ratio = NumberDecimal(3.7)
> var col_item = db.item
> col_item.insert({name: item_9_name, price: item_9_price, ratio: item_9_ratio});
WriteResult({ "nInserted" : 1 })
> col_item.find();
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84c"), "name" : "New item 1", "price" : 100000, "ratio" : 4.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84d"), "name" : "New item 2", "price" : 30000, "ratio" : 4.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84e"), "name" : "New item 3", "price" : 70000, "ratio" : 3.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd84f"), "name" : "New item 4", "price" : 45000, "ratio" : 3.6 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd850"), "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd851"), "name" : "New item 6", "price" : 30000, "ratio" : 2.8 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd852"), "name" : "New item 7", "price" : 230000, "ratio" : 5 }
{ "_id" : ObjectId("5b44c2a475e536b8f25fd853"), "name" : "New item 8", "price" : 60000, "ratio" : 4.1 }
{ "_id" : ObjectId("5b44cba275e536b8f25fd854"), "name" : "New item 9", "price" : 55000, "ratio" : NumberDecimal("3.70000000000000") }
```

■ 자주 사용하는 query 기법을 함수로 작성하여 사용

```
> searchItemByName = function(item_name) {
... return db.item.find({name: item_name}, {_id: false, name: true, price: true, ratio: true})
... }
function (item_name) {
return db.item.find({name: item_name}, {_id: false, name: true, price: true, ratio: true})
}
> searchItemByName("New item 5");
{ "name" : "New item 5", "price" : 20000, "ratio" : 2.2 }
> searchItemByName("New item 11");
```

Batch Insertion Example

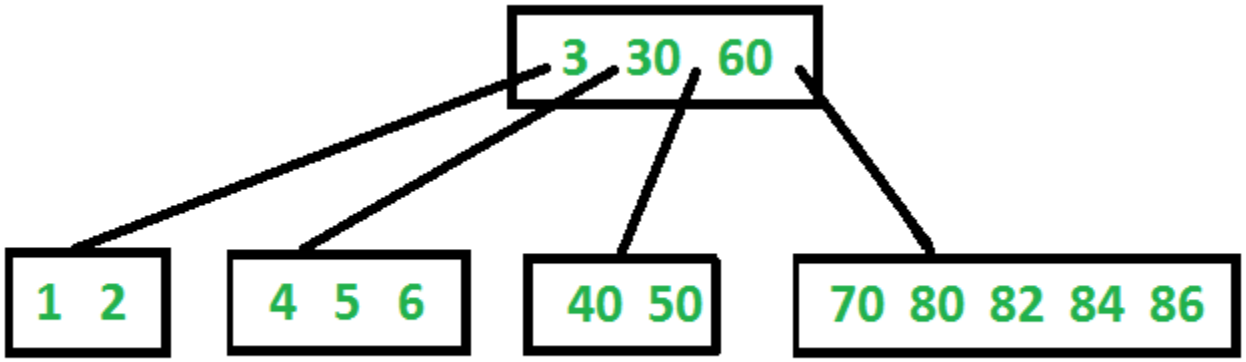

```
> for(count = 0 ; count < 1000 ; count++) {
... db.newitem.insertOne({name: "New item " + count, price: NumberInt(Math.random() * 100000), ratio: Math.random() * 5});
... }
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5b44d06c75e536b8f25fe40c")
}
> db.newitem.find().count();
1000
> db.newitem.find().limit(20);
{ "_id" : ObjectId("5b44d06c75e536b8f25fe025"), "name" : "New item 0", "price" : 88819, "ratio" : 4.601065883000835 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe026"), "name" : "New item 1", "price" : 53940, "ratio" : 3.7037770543499464 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe027"), "name" : "New item 2", "price" : 72711, "ratio" : 3.251513453946886 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe028"), "name" : "New item 3", "price" : 73124, "ratio" : 3.4799628029063645 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe029"), "name" : "New item 4", "price" : 50232, "ratio" : 0.29569244438989895 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe02a"), "name" : "New item 5", "price" : 36643, "ratio" : 3.5882089308202665 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe02b"), "name" : "New item 6", "price" : 50141, "ratio" : 0.7454403460706754 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe02c"), "name" : "New item 7", "price" : 71748, "ratio" : 3.5142484519622075 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe02d"), "name" : "New item 8", "price" : 65596, "ratio" : 2.4160448948776274 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe02e"), "name" : "New item 9", "price" : 26883, "ratio" : 2.472029355812166 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe02f"), "name" : "New item 10", "price" : 6332, "ratio" : 1.9941886706155314 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe030"), "name" : "New item 11", "price" : 80360, "ratio" : 4.3481713552109635 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe031"), "name" : "New item 12", "price" : 7972, "ratio" : 1.1227743552204883 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe032"), "name" : "New item 13", "price" : 85428, "ratio" : 2.021236000465243 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe033"), "name" : "New item 14", "price" : 42400, "ratio" : 4.085037092175 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe034"), "name" : "New item 15", "price" : 9737, "ratio" : 0.3186860528515173 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe035"), "name" : "New item 16", "price" : 90764, "ratio" : 1.4920834425396134 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe036"), "name" : "New item 17", "price" : 49880, "ratio" : 2.8192461514566345 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe037"), "name" : "New item 18", "price" : 65470, "ratio" : 4.3435105264153115 }
{ "_id" : ObjectId("5b44d06c75e536b8f25fe038"), "name" : "New item 19", "price" : 76991, "ratio" : 1.7017224130421176 }
```

Indexing

■ Indexing

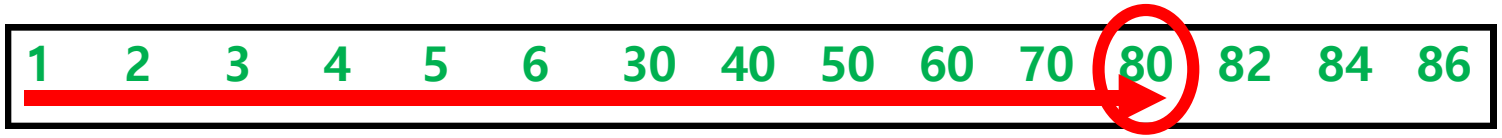
○ document에 index를 부여하지 않을 경우 read에 오랜 시간 소요

○ 찾고자하는 document를 빠르게 찾기 위해 index를 작성



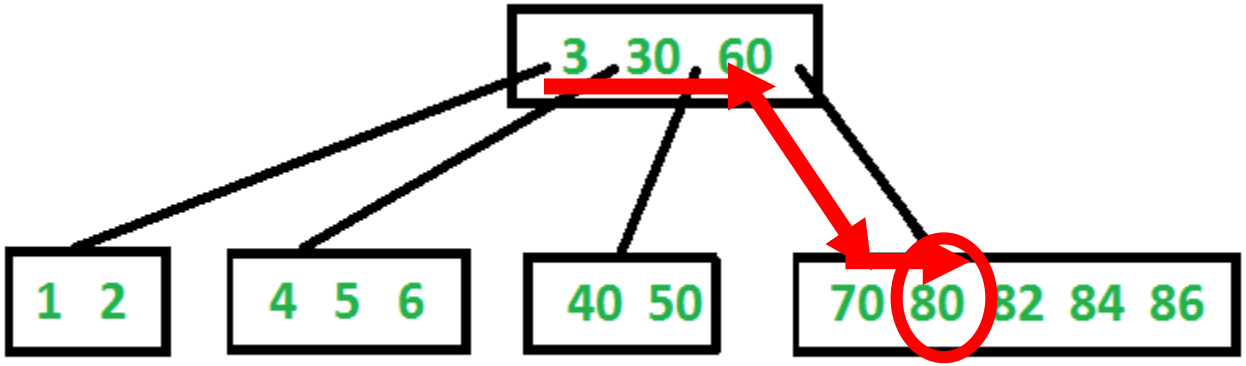
■ Indexing 작업 전

○ collection으로부터 document를 순차적으로 탐색



■ Indexing 작업 후

○ index의 key값을 통해 document의 위치를 추적 가능



■ db. 'collection 명'.getIndexes();

○ 현재 collection에 생성된 모든 index 출력

○ 기본적으로 '_id'값을 기준으로 한 index가 생성되어있음

```
> db.newitem.getIndexes();
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "newDB.newitem"
        }
]
>
```

■ db. 'collection 명'.createIndex({key: 1(-1)})

○ 1 : 오름차순, -1 : 내림차순

```
> db.newitem.createIndex({price: 1});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.newitem.getIndexes();
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "newDB.newitem"
        },
        {
                "v" : 2,
                "key" : {
                        "price" : 1
                },
                "name" : "price_1",
                "ns" : "newDB.newitem"
        }
]
```

■ db. 'collection 명'.createIndex({key 값들})

○ 다양한 key값들을 기준으로 index를 생성

○ 자주 사용하는 query 조합에 맞춰 indexing 할 경우 성능 향상

○ 복합 index로는 단일 key값의 값을 찾을 수 없음

```
> db.newitem.createIndex({price: 1, ratio: -1});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 3,
        "numIndexesAfter" : 3,
        "note" : "all indexes already exist",
        "ok" : 1
}
```

■ db.'collection 명'.totalIndexSize();

○ 생성된 index의 크기(byte 단위) 확인 가능

```
> db.newitem.totalIndexSize();
3006464
```

■ db. 'collection 명'.createIndex({key 값들})

○ 다양한 key값들을 기준으로 index를 생성

○ 자주 사용하는 query 조합에 맞춰 indexing 할 경우 성능 향상

○ 복합 index로는 단일 key값의 값을 찾을 수 없음

```
> db.newitem.createIndex({price: 1, ratio: -1});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 3,
        "numIndexesAfter" : 3,
        "note" : "all indexes already exist",
        "ok" : 1
}
```

■ db.'collection 명'.dropIndex({key 값});

○ 해당 key값 또는 key값의 조합을 만족하는 index 삭제

```
> db.newitem.getIndexes();
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "newDB.newitem"
        },
        {
                "v" : 2,
                "key" : {
                        "price" : 1
                },
                "name" : "price_1",
                "ns" : "newDB.newitem"
        },
        {
                "v" : 2,
                "key" : {
                        "price" : 1,
                        "ratio" : -1
                },
                "name" : "price_1_ratio_-1",
                "ns" : "newDB.newitem"
        }
]
```

```
> db.newitem.dropIndex({price: 1});
{ "nIndexesWas" : 3, "ok" : 1 }
> db.newitem.getIndexes();
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "newDB.newitem"
        },
        {
                "v" : 2,
                "key" : {
                        "price" : 1,
                        "ratio" : -1
                },
                "name" : "price_1_ratio_-1",
                "ns" : "newDB.newitem"
        }
]
```

■ db.'collection 명'.dropIndexes();

○ 해당 collection의 모든 index 제거

```
> db.newitem.totalIndexSize();
3006464
> db.newitem.dropIndexes();
{
        "nIndexesWas" : 2,
        "msg" : "non-_id indexes dropped for collection",
        "ok" : 1
}
> db.newitem.getIndexes();
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "newDB.newitem"
        }
]
> db.newitem.totalIndexSize();
942080
```

Practice

■ 다음 명세를 만족하는 데이터베이스를 MongoDB로 구축해보기

■ 소셜 미디어

○ 사용자는 회원가입 할 때, 이름과 ID, 비밀번호, 그리고 이메일 주소를 반드시 입력해서 가입해야 한다.

○ 회원가입 후 주소에 대한 개인정보를 입력할 수 있게 되어있다. 주소설명과 주소지, 그리고 해당 주소의 전화번호를 기입하게 되어있으며, 한 사람은 여러 주소를 등록할 수 있다.

○ 회원가입 후 프로필사진을 등록할 수 있다. (사진은 url 주소로 관리한다.)

○ 사용자는 친구를 등록할 수 있다.(편의상 내가 친구를 일방적으로 등록하는 경우로 가정) 이 때 범주(가족, 친한 친구, 직장 동료)를 만들어서 친구를 관리할 수 있다.

○ 사용자는 소셜미디어에 피드를 작성할 수 있다. 피드에는 피드를 작성한 사람과 내용, 피드를 작성한 시간이 기록되며, 사진을 첨부할 경우 여러 장 첨부할 수 있다. (사진은 url 주소로 관리한다.)

○ 비밀 그룹을 생성할 수 있다. 그룹은 그룹 생성자와 그룹 관리자, 그리고 그룹 참가자로 구성된다. 이 때 각 사용자는 비밀 그룹 내에서만 사용할 수 있는 닉네임과 프로필 사진을 등록할 수 있다.

■ 구축된 MongoDB를 기반으로 다음의 작업 수행해보기

■ 임의의 개인정보를 가진 사용자 5명 생성

○ 각각의 사용자에 대해 프로필 사진 정보를 입력(아무런 값을 넣어도 됨)

○ 이 중 2명은 주소지를 등록하지 않고, 2명은 1개의 주소만을 등록하며, 나머지 1명은 3개의 주소를 등록하도록 설정

○ 모든 사용자가 최소한 1명 이상의 친구를 가지도록 설정. 단, 2명 이상의 친구를 가진 사용자가 적어도 한 명은 있어야함. 각각의 사용자에 대해서 임의의 친구 범주를 가지도록 설정

■ 각각의 사용자에 대해서 피드를 적어도 1개씩 생성

○ 단 2개 이상의 피드를 가진 사용자가 적어도 한 명은 있어야함

■ 1개의 비밀 그룹을 생성

○ 비밀 그룹은 한 명의 그룹 생성자와, 한 명의 그룹 관리자, 그리고 3명의 그룹 참여자를 가지고 있음.

○ 각각의 참여자에 대해 임의의 닉네임과 프로필 사진 부여