KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

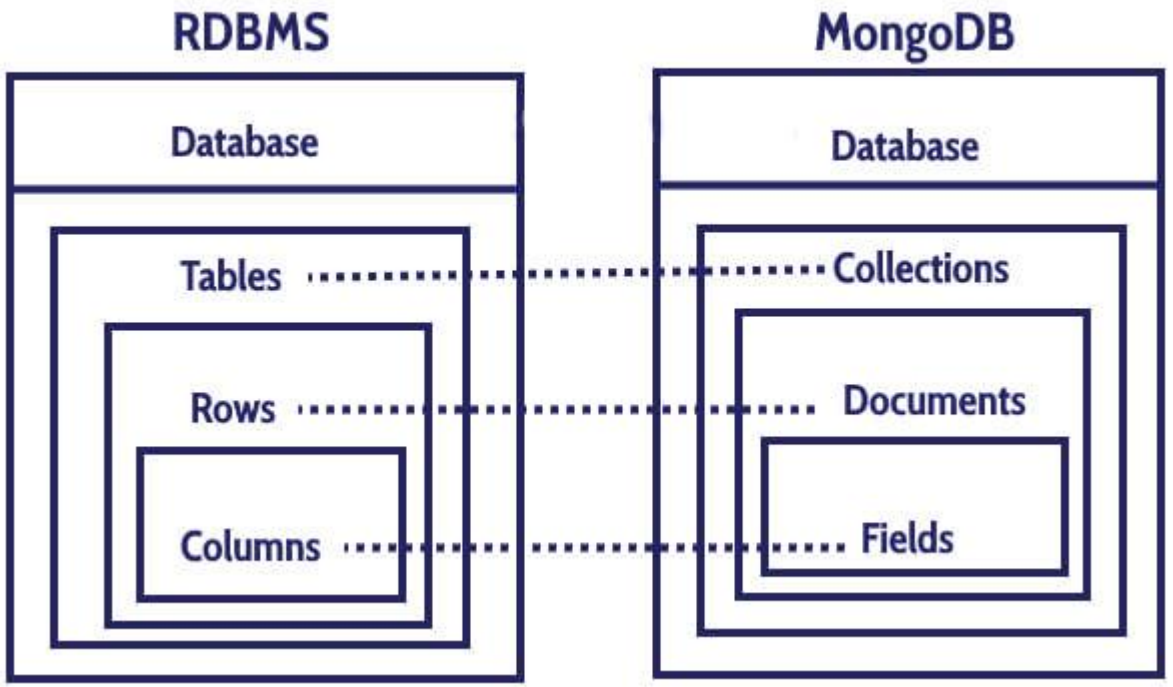
자바스크립트 셸을 통한 MongoDB

[KB] IT's Your Life

KB 국민은행

데이터베이스

- 컬렉션들의 집합
- 컬렉션을 구분하기 위한 단순 네임스페이스
- 질의를 위해서는 대상 도큐먼트가 존재하는 데이터베이스와 컬렉션을 알아야 함



데이터베이스의 선정

- use 데이터베이스명
- 현재 사용 중인 데이터베이스는 db라는 전역 변수에 설정됨

```
> use tutorial
switched to db tutorial
```

삽입과 질의

- db.컬렉션명.메서드
  - insert(): 문서 삽입
  - find(): 문서 추출

- tutorial 데이터베이스의 users 컬렉션에 문서 저장

```
> db.users.insert({username: "smith"})
```

- tutorial 데이터베이스의 users 컬렉션에 문서 질의

```
> db.users.find()

{ _id : ObjectId("4bf9bec50e32f82523389314"), username : "smith" }
```

MongoDB의 _id 필드

- 도큐먼트의 프라이머리 키
- 삽입 시 지정하지 않으면 MongDB 객체 ID(ObjectID)라는 특별한 값을 생성해서 자동으로 추가

```
> db.users.insert({username: "jones"})
> db.users.count()
2
> db.users.find()
{ _id : ObjectId("4bf9bec50e32f82523389314"), username : "smith" }
{ _id : ObjectId("4bf9bec90e32f82523389315"), username : "jones" }
```

질의 술어 넘겨주기

쿼리 셀렉터

- 컬렉션에 있는 모든 도큐먼트에 대해 일치 여부를 검사하기 위한 조건으로 사용되는 도큐먼트
- 사용자 이름이 jones인 모든 도큐먼트에 대한 질의

```
> db.users.find({username: "jones"})
{ _id : ObjectId("4bf9bec90e32f82523389315"), username : "jones" }
```

비어 있는 술어

- 모든 문서를 리턴
- db.users.find()와 db.users.find({})는 동일

질의 술어 넘겨주기

AND 조건 검사

```
> db.users.find({_id : ObjectId("4bf9bec50e32f82523389314") , username: "smith" });

> db.users.find({ $and: [
  {_id : ObjectId("4bf9bec50e32f82523389314") },
  { username: "smith" }
]})

{ _id : ObjectId("4bf9bec50e32f82523389314"), username : "smith" }
```

OR 조건 검사

```
> db.users.find({ $or: [
  { username: "smith" },
  { username: "jones" }
]})

{ _id : ObjectId("4bf9bec50e32f82523389314"), username : "smith" }
{ _id : ObjectId("4bf9bec90e32f82523389315"), username : "jones" }
```

도큐먼트 업데이트

- 컬렉션.update(쿼리문서, 갱신문서, upsert여부, 다중적용여부)
- 업데이트 유형
  - 부분 업데이트
  - 대체 업데이트

$set/$unset 연산자 업데이트

문서의 한 부분 만 수정

- $set : { 키: 값, ... }
- 키가 존재하지 않으면 추가

```
> db.users.find({username: "smith"})
{
  "_id" : ObjectId("4bf9ec440e32f82523389316"),
  "username" : "smith"
}

// update
> db.users.update({username: "smith"}, {$set: {country: "Canada"}})

> db.users.find({username: "smith"})
{
  "_id" : ObjectId("4bf9ec440e32f82523389316"),
  "country" : "Canada",
  "username" : "smith"
}
```

대체 업데이트

- 도큐먼트를 다른 것으로 대체

```
> db.users.replaceOne({username: "smith"}, {country: "Canada"})  // 구 버전: db.users.update(...)

> db.users.find({country: "Canada"})
{
  "_id" : ObjectId("4bf9ec440e32f82523389316"),
  "country" : "Canada"
}

> db.users.update({country: "Canada"}, {username: "smith", country: "Canada"})
```

$unset 연산자

해당 키를 삭제

- $unset: { 키: 1, ... }

```
> db.users.update({username: "smith"}, {$unset: {country: 1}})

> db.users.find({username: "smith"})
{
  "_id" : ObjectId("4bf9ec440e32f82523389316"),
  "username" : "smith"
}
```

복잡한 데이터 업데이트

```
{
    username: "smith",
    favorites: {
        cities: ["Chicago", "Cheyenne"],
        movies: ["Casablanca", "For a Few Dollars More", "The Sting"]
    }
}
```

```
db.users.update( {username: "smith"},
 {
  $set: {
   favorites: {
    cities: ["Chicago", "Cheyenne"],
    movies: ["Casablanca", "For a Few Dollars More", "The Sting"]
   }
  }
 })
```

복잡한 데이터 업데이트

```
> db.users.update( {username: "jones"},
 {
  $set: {
   favorites: {
    movies: ["Casablanca", "Rocky"]
   }
  }
 })

> db.users.find().pretty()
```

```
{
    "_id" : ObjectId("552e458158cd52bcb257c324"),
    "username" : "smith",
    "favorites" : {
        "cities" : [
            "Chicago",
            "Cheyenne"
        ],
        "movies" : [
            "Casablanca",
            "For a Few Dollars More",
            "The Sting"
        ]
    }
}
{
    "_id" : ObjectId("552e542a58cd52bcb257c325"),
    "username" : "jones",
    "favorites" : {
        "movies" : [
            "Casablanca",
            "Rocky"
        ]
    }
}
```

내부 문서 검색

- "부모키.자식키"
  - 반드시 문자열 표시를 해줘야 함
  - "favorites.movies"

- 배열 검색
  - 배열 요소 명으로 검색 가능

```
> db.users.find({"favorites.movies": "Casablanca"}).pretty()
```

더 발전된 업데이트

- 세 번째 파라미터:
  - upsert 여부, 해당 문서가 없는 경우 insert 할지 여부. 디폴트는 false
- 네번째 파라미터:
  - 다중 업데이트 여부. 디폴트는 false
- 배열에 요소 추가
  - $push
    - 배열에 무조건 추가, 중복 가능
  - $addToSet
    - 중복 없이 배열에 추가

```
db.users.update( {"favorites.movies": "Casablanca"},
    {$addToSet: {"favorites.movies": "The Maltese Falcon"} },
    {upsert: false,   // insert if not found?
     multi:true })   // update all found? (if false, updates just first it finds)
또는
db.users.updateMany( {"favorites.movies": "Casablanca"},
    {$addToSet: {"favorites.movies": "The Maltese Falcon"} },
    {upsert: false})   // insert if not found?
```

데이터 삭제

컬렉션.remove(쿼리문서)

- 검색 조건에 해당하는 모든 문서 삭제
- 모두 삭제되도 컬렉션은 유지 됨

```
> db.users.remove({"favorites.cities": "Cheyenne"})
> db.users.remove({})
```

컬렉션.drop()

- 컬렉션 자체를 삭제 함

```
> db.users.drop()
```

대용량 컬렉션 생성

- numbers 컬렉션에 20,000개의 문서 생성

```
for(let i = 0; i < 200000; i++) {
    db.numbers.insert({num: i});
}

> db.numbers.count()
20000

> db.numbers.find()

> db.numbers.find({num: 500})

> db.numbers.find( {num: {"$gt": 199995 }} )
{ "_id" : ObjectId("4bfbf1dedba1aa7c30afcade"), "num" : 199996 }
{ "_id" : ObjectId("4bfbf1dedba1aa7c30afcadf"), "num" : 199997 }
{ "_id" : ObjectId("4bfbf1dedba1aa7c30afcae0"), "num" : 199998 }
```

범위 쿼리

$gt, $gte, $lt, $lte 연산자

```
> db.numbers.find( {num: {"$gt": 199995 }} )
{ "_id" : ObjectId("4bfbf1dedba1aa7c30afcade"), "num" : 199996 }
{ "_id" : ObjectId("4bfbf1dedba1aa7c30afcadf"), "num" : 199997 }
...

> db.numbers.find( {num: {"$gt": 20, "$lt": 25 }} )
{ "_id" : ObjectId("4bfbf132dba1aa7c30ac831f"), "num" : 21 }
{ "_id" : ObjectId("4bfbf132dba1aa7c30ac8320"), "num" : 22 }
{ "_id" : ObjectId("4bfbf132dba1aa7c30ac8321"), "num" : 23 }
{ "_id" : ObjectId("4bfbf132dba1aa7c30ac8322"), "num" : 24 }
```

인덱싱와 explain()

- explain() : 쿼리 수행 성능 통계 자료 출력

> db.numbers.find({num: {"$gt": 199995}}).explain("executionStats")

```
"executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 4,
    "executionTimeMillis" : 8,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 20000,
    "executionStages" : {
```

인덱스 생성

- 컬렉션.createIndex({키: 1, ... })
  - 지정한 키로 오름 차순 정렬된 인덱스 생성
  - 내림 차순 정렬된 인덱스 생성시 -1 설정
  - 여러 개의 키를 제시한 경우 조합 키에 대해 인덱스 생성

```
> db.numbers.ensureIndex({num: 1})
```

인덱스 확인

- 컬렉션.getIndexes()

```
> db.numbers.getIndexes()
```

인덱스 쿼리에 대한 explain() 결과

> db.numbers.find({num: {"$gt": 199995 }}).explain("executionStats")

```
"executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 4,
    "executionTimeMillis" : 0,
    "totalKeysExamined" : 4,
    "totalDocsExamined" : 4,
    "executionStages" : {
        "stage" : "FETCH",
        "nReturned" : 4,
        "executionTimeMillisEstimate" : 0,
        "works" : 5,
        "advanced" : 4,
        "needTime" : 0,
        "needFetch" : 0,
```

오직 네 개의 도큐먼트만 스캔

데이터베이스 정보 얻기

데이터베이스 목록 보기

> show dbs

현재 사용 중인 데이터베이스의 컬렉션 목록 보기

> show collections

현재 사용 중인 데이터베이스 및 컬렉션 상태 보기

> db.status()

> db.numbers.status()