Collection & Document 기본 문법

소프트웨어공학과 / 201332001 강인성 / hogu9401@gmail.com

1. Database 기본 문법

Database는 Collection들을 모아 둔 물리적 모델링으로 볼 수 있는데 Database를 기본적으로 생성하는 문법을 알아보도록 하자. 우선 문법들을 공부하기 앞서 cmd 창에 mongod를 이용해 MongoDB 서버에 접속을 완료한 후, 또 다른 cmd 창에서 mongo를 이용해서 MongoDB Shell를 실행을 해서 실습 환경을 준비 해두록 한다.

A. Database 생성

MongoDB에서 Database를 생성하는 문법은 use (데이터베이스 이름)을 통해 생성을 할 수 있다.

(참고로 MongoDB와 관련된 문법들에 대해서는 일반 소스 코드와 구분을 하기 위해 >를 이용해서 작성을 하겠다.)

```
> use example01
```

현재 MongoDB에 존재하는 데이터베이스의 이름이 example01이 존재한다면 이를 반환하고 없다면 새롭게 생성을 해 주는 역할로 보면 된다.

B. db, show dbs 명령어

현재 어느 데이터베이스를 가리키는지에 대하여 확인을 하고 싶은 경우에 db란 명령어를 이용해서 알 수 있다.

그리고 현재 MongoDB에 저장된 데이터베이스 목록을 확인하고 싶은 경우에는 show dbs를 통하여 확인이 가능하다.

각 명령어 순서는

(1) 현재 모든 database 목록들을 참고할 수 있는 show dbs 명령어,

(2) 다음은 local database를 쓰겠다고 선언을 하여

(3) local를 가리키고 난 후 어느 database를 가리키는가에 대한 명령어인 db를 입력하면 local이 나오게 된다.

(+) 참고로 database를 use 명령어를 통해 추가를 한 경우에 show dbs 명령어로 조회를 한다면 곧바로 새로 생성된 database가 나오지 않는다는 점을 명심하고 있자. 새로 추가한 데이터베이스에 최소 하나의 document가 생성을 하고 난 후에 show dbs 명령어를 입력하면 새로운 database가 나오게 된다.

C. Database 제거

데이터베이스를 제거하는 과정은 간단하다. 그렇지만 주의를 해야 하는 점이 db라는 명령어를 통해서 현재 어느 데이터베이스를 가리키는가에 대해서 인지를 하고 난 후에 삭제를 진행을 하길 바란다. 실수로 다른 데이터베이스가 삭제 되어 후회하는 일이 없길 바란다.

```
> db.dropDatabase()
```

실제로 db라는 명령어 중에 dropDatabase 명령어는 말 그대로 Database를 영구 삭제를 하는 역할을 해 준다. 이는 Relational Database 명령어 중에 Drop를 활용해서 쓰는 원리와 같다고 볼 수 있는데 그렇지만 db 명령어를 확인해서 삭제를 하는 것을 염두를 하자.

2. Collection 기본 문법

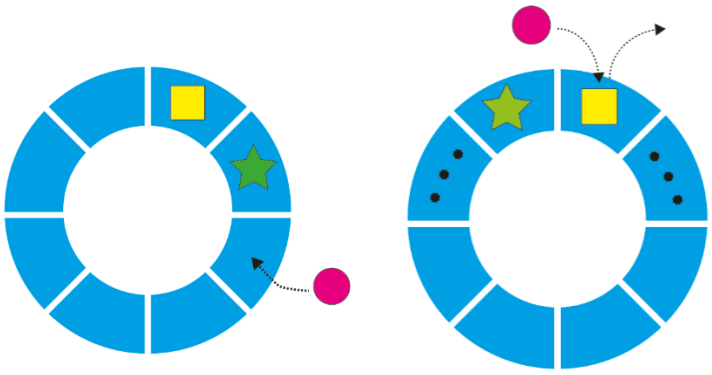
Collection은 Relational Database에서 흔히 이야기 하는 Table과 같은 개념이다. Collection을 추가하거나 삭제를 하기 위해서도 방금 전 명령어 db를 통하여 현재 데이터베이스에 어떤 Collection들이 저장이 되어 있는지에 대해 확인을 할 필요가 있다.

A. Collection 생성하기

Database 내부에서 Collection을 생성하는 경우에는 현재 가리키는 db를 통해서 createCollection 함수를 이용해서 Collection을 생성하면 된다. 우선 방금 전에 새로 만들었던 database인 example01를 통해서 collection을 추가해보도록 하자. 초반에는 use example01를 통해서 연산자 db가 이를 가리키게끔 해야 한다. 기본적인 명령어는 아래의 형식처럼 구성이 되어 있는데 option에 쓰는 값들은 생략 해도 무관은 하다. 그러나 Capped Collection을 통하여 collection에 한계를 주는 경우에는 써 줘야 한다.

```
> db.createCollection("music")
[db.createCollection(콜렉션_이름, {옵션 : 옵션 값 ... })]
```

여기서 option에서 쓰는 값들에 대해서도 한 번 고려를 할 필요가 있는데 Capped Collection을 구축해야 할 상황이 다가온다면 option을 쓰는 란에 꼭 써 줘야 한다. 여기서 Capped Collection은 일반 Collection과는 달리 데이터가 추가, 삭제될 때 Collection의 사이즈가 고정이 되어 있는 Collection으로 이해할 수 있다. Aging-out를 기반으로 Document가 저장되는 순서 그대로 저장을 해서 Circular Queue(원형 큐)와 같은 구조를 취하고 있다.



[그림] Circular Queue(원형 큐)는 왼쪽처럼 새로운 데이터가 들어오면 맨 마지막에 추가가 되고, Queue의 Size를 범람하게 되면 가장 오래된 데이터가 삭제되고 그 자리에 새로운 데이터가 들어가게 된다.

Capped Collection을 위하여 Option 매개 변수에 대해서는 어떻게 쓰이는가에 대해 표로 언급을 해 두겠다. 자세한 내용은 Capped Collection을 다룰 때 알아보겠다.

Field	Type	설명
capped	Boolean	Capped Collection 으로 생성하기 위해서 꼭 true 로 설정을 해야 하는데 이를 true 로 할 시 size 를 설정해줘야 한다.
size	number	해당 Collection 의 최대 byte 사이즈.
max	number	해당 Collection 에 있는 Document 의 최대 개수.

참고로 MongoDB 이전 버전(현재는 3.6 버전)에서는 autoIndexID란 값을 설정하는 란이 있었는데 _id를 통하여 인덱스를 자동으로 생성을 하는가에 대한 여부를 작성을 해야만 했다. Relational Database에서 Table 내에 존재하는 Primary Key(기본 키)를 통하여 자동으로 Index를 생성해주는 역할을 하였지만, 이를 반영해서 MongoDB 3.2 버전 이후로는 안 적어도 무관하게 되었다.

이 옵션을 통해서 Collection을 생성한다면 아래와 같이 작성할 수 있다.

```
> db.createCollection("music02", { capped : true, size : 5242880, max : 150 })
[db.createCollection(콜렉션_이름, {옵션 : 옵션 값 ... })]
```

이처럼 작성을 하게 된다면 music02 Collection은 Document의 개수가 150개로 한정이 되며, Collection 내부에 있는 총합 용량의 제한이 5MB가 된다

.

B. Collection 제거하기

Collection을 삭제하는 방법은 간단하다. 현재 가리키고 있는 db 명령어를 이용해서 collection 이름과 drop() 함수를 통해 Collection을 삭제할 수 있다. 방금 전에 만들었던 music02 Collection을 삭제하기 위해서는 방금 전에 db 명령어를 example01로 가리키게끔 하고 난 후에 아래와 같은 문장으로 Collection을 삭제하면 된다.

```
> db.music02.drop()
[db.콜렉션_이름.drop()]
```

그리고 방금 삭제된 Collection이 처리가 되었으면 어떤 Collection들이 남아 있는지 여부를 참고할 때 쓰는 show collections로 작성해서 확인하면 된다. 현재 db가 무엇을 가리키는지에 대해 확인을 하고 작성을 한다.

3. Document 기본 문법

Document는 Collection 내부에 존재하는 각 데이터들을 Key-Value Model 처럼 정리한 개념인데 Relational Database에서 이용했던 Record / Tuple / Row로 인식할 수 있다. 우선 Document의 기본 문법을 알아보기 위해서 이 내부에서 주로 쓰는 데이터 타입들에 대하여 인지를 하고 넘어가도록 하자.

A. Document Data Type

MongoDB에서는 많은 데이터 타입들을 제공을 하는데 어떤 종류들이 있는지 알아보자.

변수 Type	설명
String	객체 지향 언어에서도 String을 쓰지 않는 언어는 없을 것이다. 또한 MongoDB에서도 String을 많이 쓰는 편으로서 UTF-8로 인코딩이 되어 있는 문자열을 저장한다.
Integer	정수를 저장할 때 쓰는 변수 타입으로 크기는 int에 해당되는 32비트(4byte), long에 해당되는 64비트(8byte)로 나뉘게 된다.(32-bit Integer / 64-bit Integer)
Boolean	true / false 를 저장할 때 쓴다.
Double	부동 소수점이 지정된 유리수를 저장할 때 쓴다.
Mix / Max Key	BSON 요소들 중에서 최솟값, 최댓값을 구분할 때 쓰이는 Key.
Array	한 Field 내부에 부지기수한 데이터들을 넣을 때 쓰는데 즉 배열, List를 저장할 때 쓴다.
Timestamp	이는 Java를 기반할 때 java.sql.Timestamp에서 생성 / 수정된 날짜를 저장할 때 쓴다.
Object	각 Document를 통해 내장 된 객체를 넣을 때 쓴다. Relational Database에서 1:1, 1:N, M:N으로 Mapping된 객체를 생성할 때처럼 생각하면 된다.
Null	말 그대로 Null 값을 넣을 때 쓴다.
Symbol	String으로 작성된 값들 중에서 변경이 불가능한 값을 넣을 때 쓴다. Enum으로 묶은 String을 쓸 때 용이하다.
Date	UNIX 시간 포맷, Java에서는 java.util.Date를 통해 생성된 날짜를 저장한다고 보면 된다.
Object ID	다른 Document의 Object ID를 저장한다. Relational Database에서 쓰는 Primary Key와 같은 맥락인 건 말 하지 않아도 알 것이다.
Binary Data	MySQL에서도 BLOB 데이터 형이 있듯이 저장하는 파일을 byte 배열로 받아와서 이를 저장할 때 쓴다고 보면 된다.
Code	JavaScript 함수를 저장할 때 쓴다.
Regular Expression	JavaScript에서 쓰는 정규식을 넣는다. 그러나 정규식은 어느 프로그래밍 언어에서도 같은 약속이니 같은 정규식을 넣어도 무관하다.

참고로 이 데이터 타입들이 실제로 MongoDB에서 Document 내부에 저장되는 BSON(Binary JSON)에서 쓰는 데이터 타입으로 이해할 수 있다.

B. Collection에 Document 생성

방금 우리가 만든 music Collection 내부에 다양한 데이터들을 넣어보는 연습을 해보도록 하자.

현재 가리키고 있는 db가 example01인데 필자는 use music_example란 명령어를 통해 새로운 database를 만들어서 작성을 할 예정이다. 우선 Document를 생성하는 문장은 아래와 같이 쓰인다. 참고로 필드 이름에 큰 따옴표로 구분을 하는 경우가 있는데 이는 선택이다.

```
> db.music.insert({
    title : "To Heaven",
    singer : "조성모",
    year : 1998,
    genre : "발라드"
});
```

db 명령어를 추가하고 싶은 collection의 database로 따 와야 한다.

[db.콜렉션_이름.insert({ 추가 필드 이름 : 추가 내용, ... })]

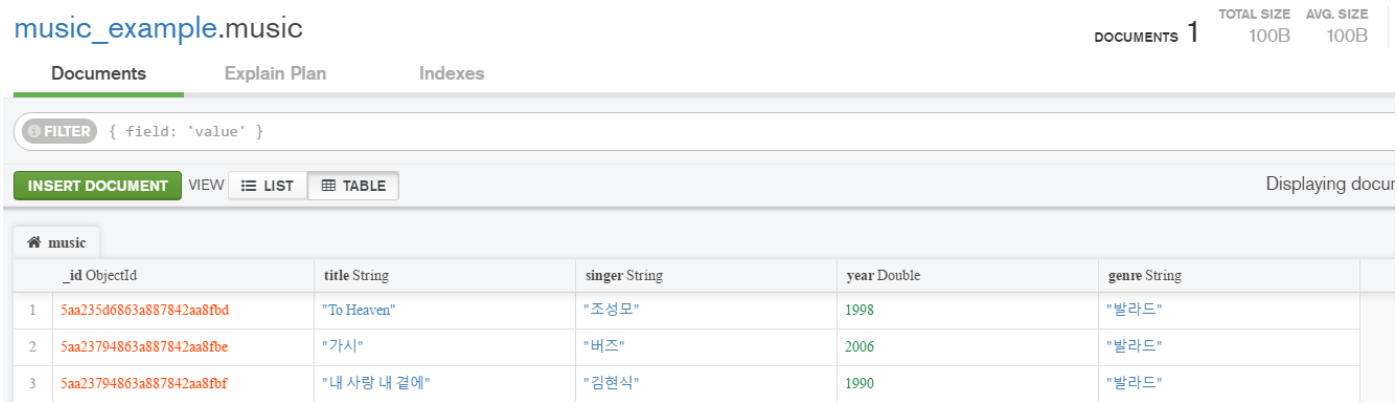
이 명령어를 통해서 데이터를 추가한다면 초기 music Collection 내부에서 쓰는 Field 목록들은 title, singer, year, genre 4가지로 나뉘게 된다. 참고로 NoSQL에서는 Join 연산을 안 쓰기 때문에 이를 대체할 객체를 쓰는 방안에 대해서는 NoSQL 데이터 모델링에서 자세하게 알아보도록 하자.

물론 한 Document 이외에도 여러 개의 Document를 추가하는 문장을 쓸 수 있다.

```
> db.music.insert([
  {
    title : "가시",
    singer : "버즈",
    year : 2006,
    genre : "발라드"
  },
  {
    title : "내 사랑 내 곁에",
    singer : "김현식",
    year : 1990,
    genre : "발라드"
  }
);
```

[db.콜렉션_이름.insert([{ 추가 필드 이름 : 추가 내용, ... }, ...])]

여러 개의 음악들을 추가 할 때 _id에 대한 값이 겹쳐질 의심이 있어서 실제로 MongoDB Compass를 통해 실험을 하게 된다면 다음 페이지와 같은 결과가 나오게 되어 insert를 여러 Document들을 추가를 해도 상관은 없다.



[그림] 여러 데이터들을 한꺼번에 추가하고 실행을 한 결과. _id는 여러 데이터들을 추가를 하더라도 증차 번호에 맞춰서 구분을 해 준다.

또한 Collection을 안 만들었다고 하더라도 Document를 추가하는데 상관은 없다. 그렇지만 Capped Collection을 형성하는 경우에는 설정을 따로 해 주고 삽입을 하길 권장을 하고 일반 Collection을 추가하면서 Document를 동시에 추가를 하는 경우에는 무관하게 진행을 해도 된다.

```
> db.music02.insert(
  {
    title : "사랑할수록",
    singer : "부활",
    year : 1993,
    genre : "락"
  }
);
```

Document를 추가 완료하고 난 후에 각 Collection에는 어떤 데이터들이 존재하는지 인지하기 위해서는 다음과 같은 명령어를 통해 확인을 하면 된다. 그리고 find 메소드는 다음 노트에 조회 쿼리에 대하여 작성할 때 쓰인다.

```
> db.music.find()
[ db.콜렉션_이름.find() ]
> 실행 결과
{ "_id" : ObjectId("5aa235d6863a887842aa8fbd"), "title" : "To Heaven", "singer" : "조성모", "year" : 1998, "genre" : "발라드" }
{ "_id" : ObjectId("5aa23794863a887842aa8fbe"), "title" : "가시", "singer" : "버즈", "year" : 2006, "genre" : "발라드" }
{ "_id" : ObjectId("5aa23794863a887842aa8fbf"), "title" : "내 사랑 내 곁에", "singer" : "김현식", "year" : 1990, "genre" : "발라드" }
```

C. Document 삭제하기

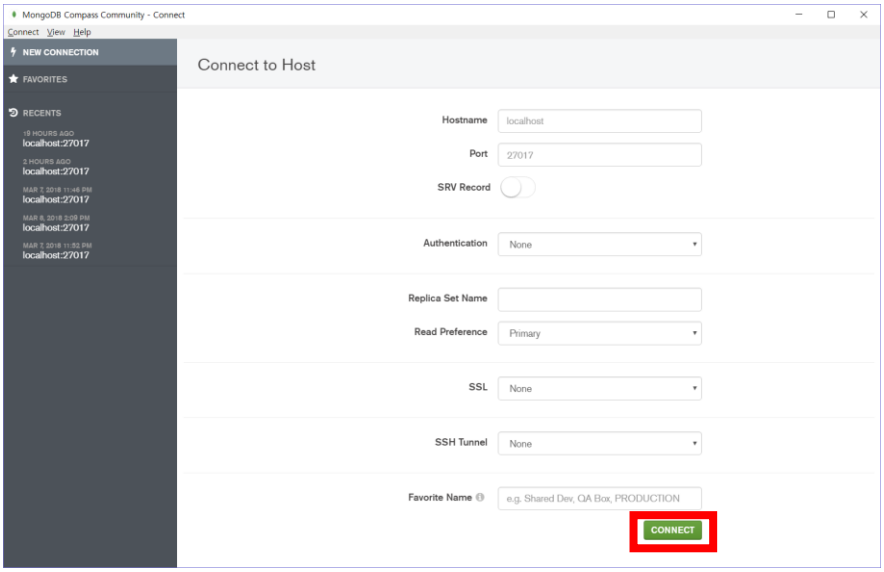
Document를 삭제하는 방법은 remove() 함수를 이용해서 삭제를 한다. 이처럼 적용을 하면 된다.

```
> db.music.remove(
  {
    title : "가시"
  }, false
);
[ db.콜렉션_이름.remove({ 삭제 Document 조건. 필드끼리 정리해서 쓰도록 함. }, 1개만 삭제 여부 );
[ db.콜렉션_이름.remove() -> 이 함수는 모든 Document를 삭제한다. Collection은 그대로 남아 있다. ]
```

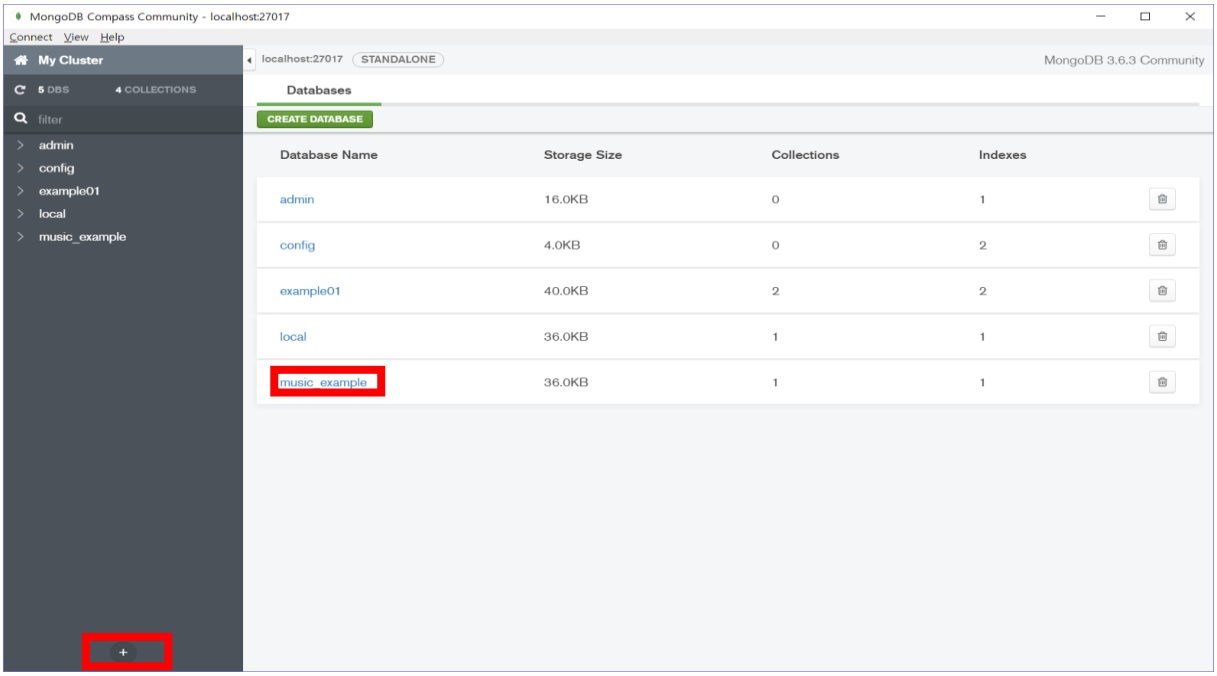
Relational Database에서 생각했던 Delete 문과 비슷하게 생각할 수 있는데 삭제 Document 조건은 Delete 문에서 Where 문과 같이 적용할 수 있다고 보면 되고, 1개만 삭제 여부는 삭제 Document 조건에 해당되는 Document 목록들 중에서 가장 오래 된 데이터를 삭제하게 된다. 그러면 Collection 내부에 존재하는 Document들을 모조리 삭제하는 경우에는 어떻게 써야 할까? 바로 삭제 Document 조건에 { } 를 써서 모두를 삭제하는 방법이나 그냥 remove() 함수를 작성해 삭제를 하는 방법이 있다. 이러한 방법이 Relational Database에서 Delete 문에 어느 조건 없이 삭제하는 Truncate와 같은 원리로 상기할 수 있다. 고로 Truncate와 Drop은 서로 다른 개념이다. Truncate는 Table에 현존하는 Record들만 모두 삭제를 하는 개념이고, Drop은 Table와 그 속에 있는 Record를 아예 없애는 개념이다.

D. MongoDB Compass를 활용한 Database, Collection, Document 생성

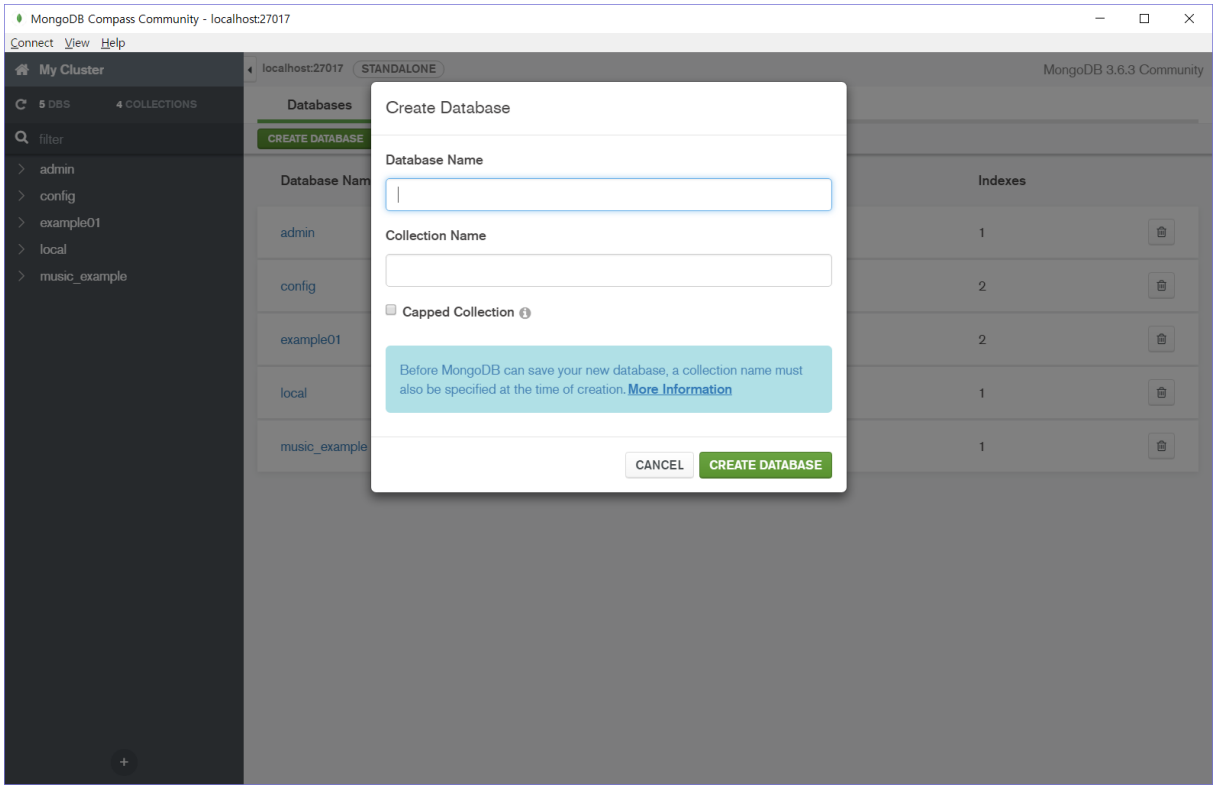
MongoDB 문법에 대하여 하나도 인지를 안 한 상태에서도 Database, Collection, Document를 생성하는 방법은 Compass를 이용해서 활용하는 방법이다. Compass를 통해서 NoSQL 데이터 모델링을 하는 경우가 과반수이지만 어떠한 문장으로 돌아가는지에 대해서는 어느 정도 인지를 하고 있어야 실제로 MongoDB를 이용하는 방법을 배우는 데 도움이 된다. 일단 MongoDB에서 제공하는 Compass를 실행을 하기 이전 cmd 창에서 mongod란 문장을 입력해서 MongoDB 서버에 접속을 해주고 난 후에 바로 CONNECT를 클릭해 실행하면 된다.



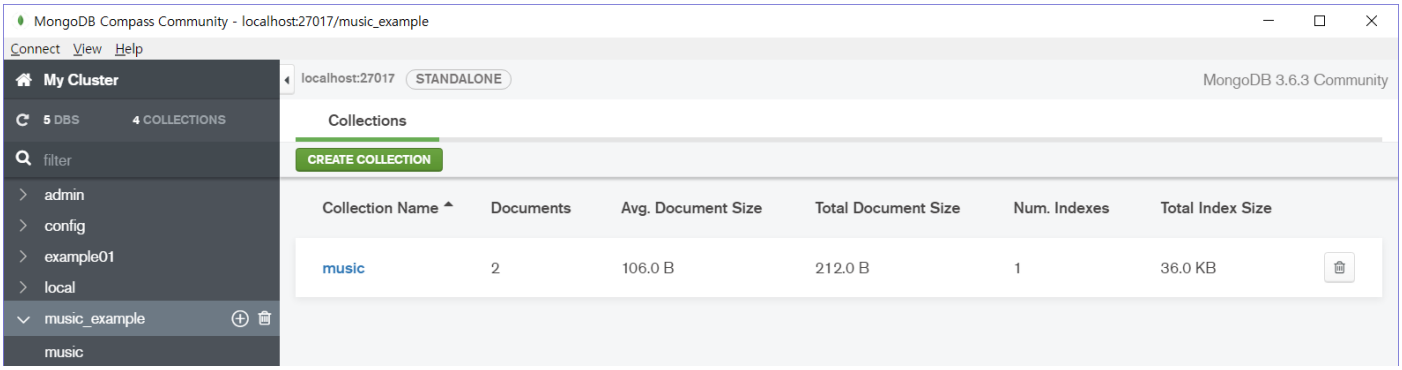
[그림] 초기 MongoDB Compass를 실행하고 난 후에 서버에 접속을 하고 난 후 Connect 버튼을 누르고 실행.



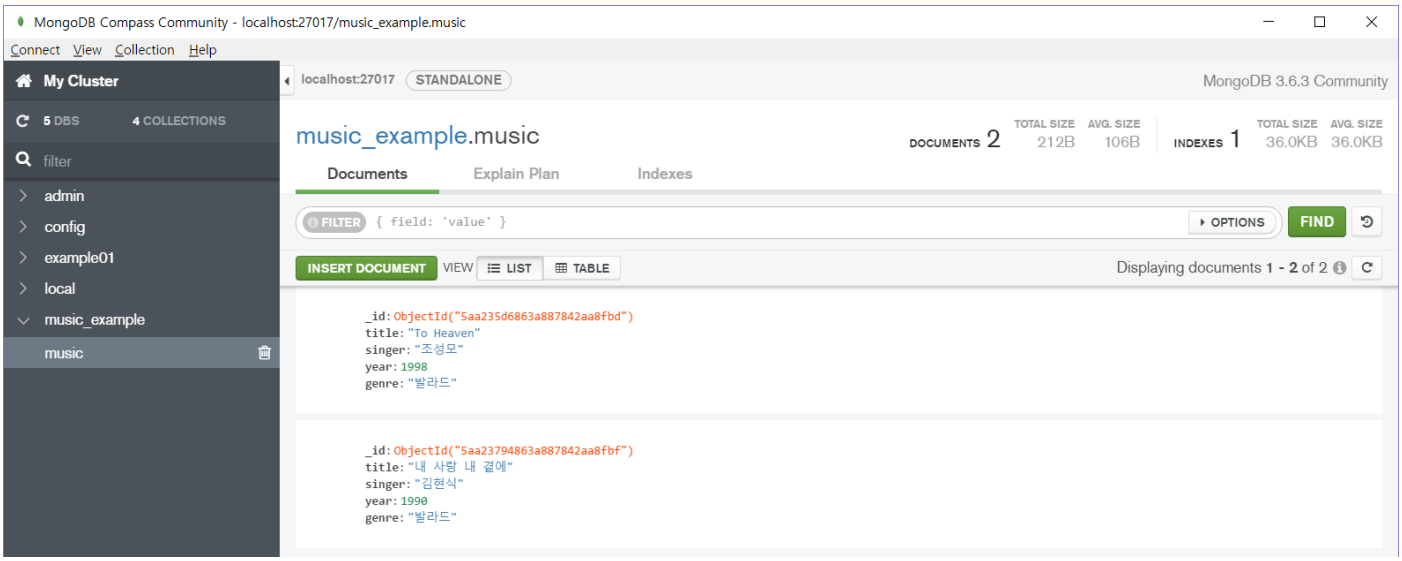
[그림] 초기에 실행하게 된다면 왼쪽 하단부 +를 클릭하면 Database를 생성할 수 있고, 오른쪽 표에서 Database를 클릭하면 Collection 목록들을 보여준다. 그리고 Database 목록들 중에 휴지통을 클릭하면 Database를 삭제할 수 있다.

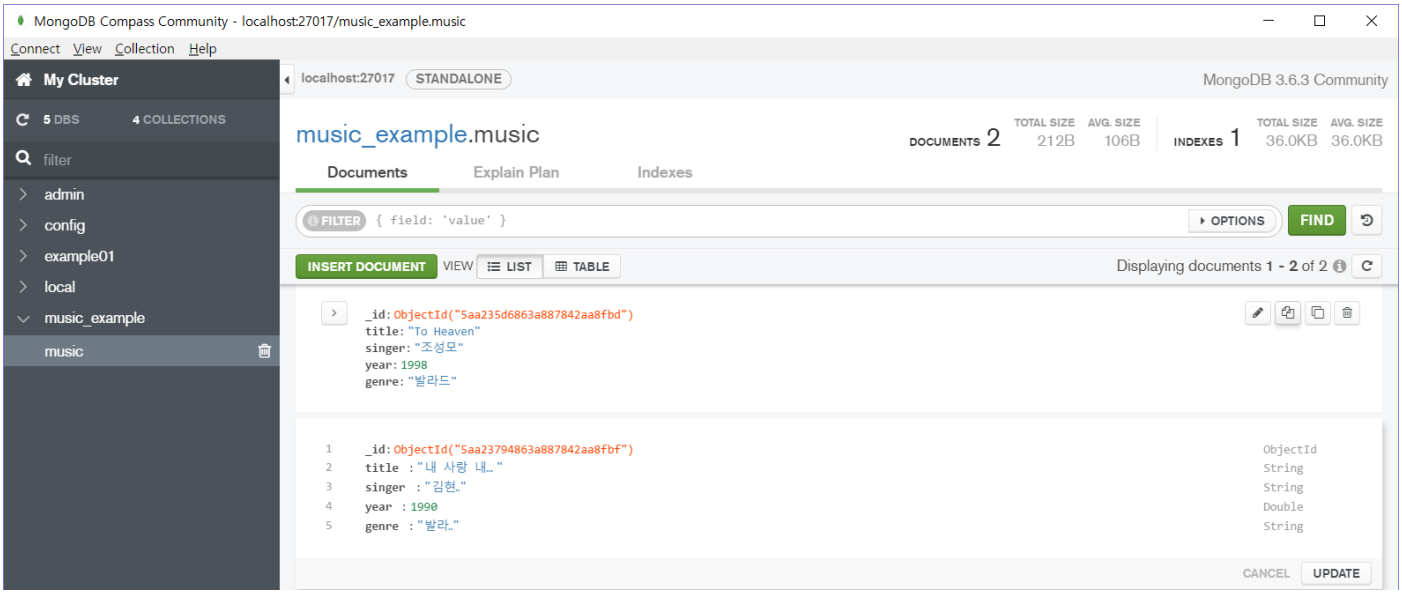


[그림] 데이터베이스를 초기에 추가할 때에는 Database 이름과 Collection 이름 2개를 입력하게 된다. Database는 그 Database 이름, Collection 이름은 추가하는 Database에 초기에 추가할 Collection 이름이 되고 여기서 Capped Collection 여부에 대한 설정도 가능하다.



[그림] 데이터베이스를 클릭하게 되면 Collection 목록들이 나오는데 좌측에 데이터베이스 이름 옆에 둥근 + 버튼을 클릭하면 Collection을 추가할 수 있으며 우측에는 현재 클릭한 데이터베이스의 Collection 목록들이 나오고 Document 크기와 인덱스 여부들을 확인할 수 있다.





[그림] 현재 현존하는 music Collection을 클릭하면 방금 전에 추가한 노래 2곡들이 저장되어 있음을 확인할 수 있다. 여기서 List와 Table 2개의 보기 형식이 있는데 말 그대로 List로 볼 것인가 Table로 볼 것인가에 대하여 선택을 할 수 있다. 그리고 각 요소들을 클릭하면 데이터 값을 수정할 수 있다. 물론 삭제도 가능하다.

[출처]

MongoDB Tutorial - https://www.tutorialspoint.com/mongodb/index.htm

튜토리얼 한글 번역판 - https://velopert.com/457

Capped Collection에 대한 설명 - http://cinema4dr12.tistory.com/373

MongoDB Compass 캡쳐 – 본인이 작업하여 설명과 함께 작성함.