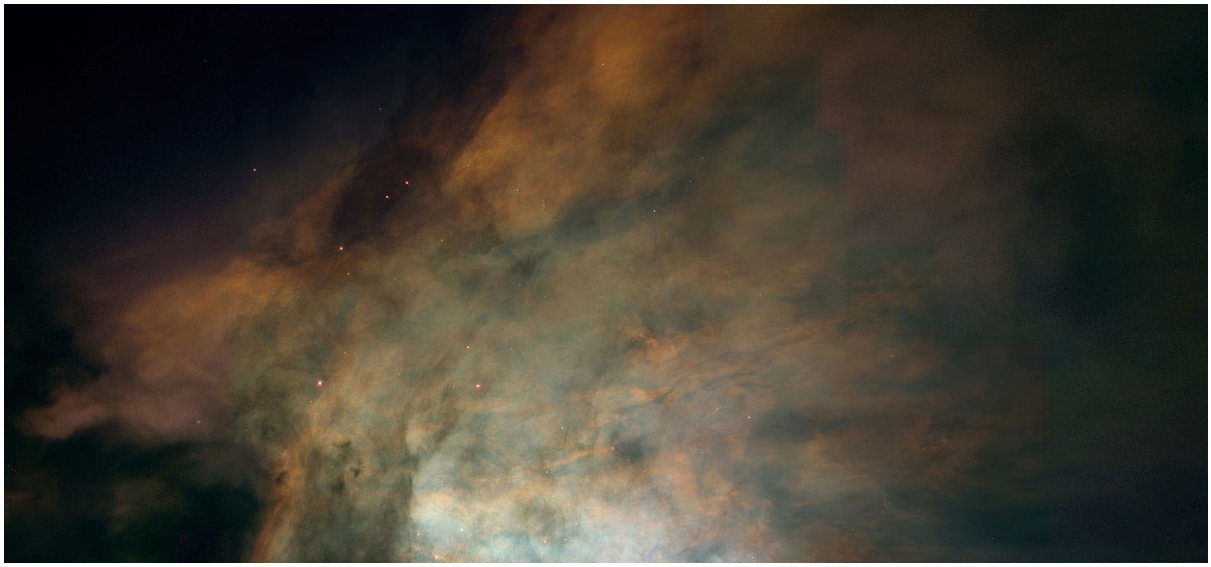

5. MongoDB CRUD 쓰기 연산

. 기본 데이터 처리

. ObjectId

- MongoDB는 하나의 컬렉션을 생성한 후 데이터를 입력하면 각 도큐멘트마다 고유한 값으로 구성된 ObjectID할당. 하나의 컬렉션 내에 저장된 도큐멘트를 다른 도큐멘트와 구분할 수 있는 식별자로 사용. (12byte binary data)
- 해당 ObjectId를 사용하여 CRUD 컨트롤 가능
- RDBMS에서 Primary key를 만들 때 DB서버로 중복되지 않는 키를 골라서 그 값을 key로 저장하는 방식
- MongoDB는 자신이 필요한 shard에게만 query를 요청, mongos가 config server의 data를 토대로 각 shard가 어떤 값을 가졌는지 가지고 있다가 query를 요청할 때 자신이 필요한 shard에게만 요청 따라서 shard key에 해당하는 data가 미완성인 상태로 서버에 저장하도록 요청할 수 없고 client가 ObjectId를 생성하여 값을 저장
- ObjectId는 크게 4부분으로 나뉨
  - 처음 4byte는 시간정보
  - 다음 3byte는 머신별 고유 Id(mac address + ip address or hostname)을 md5 hash 한 값
  - 다음 2byte는 process id 정보
  - 마지막 3byte는 mongos에서 생성한 값

- 이러한 ObjectId의 값이 같은 확률은 희박하지만 낮은 확률로 혹은 의도적으로 같은 ObjectId가 발생하는 일이 있는데 이런 경우 shard에서 E11000 duplicated key error 발생

. 기본 CRUD 쓰기 쿼리의 기본 동작

. Write Concern

- 보통 MongoDB는 Client가 보낸 데이터를 Primary에 저장 후 Secondary로 동기화 → 시간차 발생
- Sync 완료전 Primary에 이슈가 있을 시 일관성이 깨짐 → Write Concern(쓰기 고려) 옵션 지원
- Sync가 다 완료된 이후에 Client에게 데이터를 반환하여 일관성을 보장해줌
- w option
  - Replica set에 속한 멤버중 지정된 수만큼의 멤버에게 데이터 쓰기가 완료되었는지 확인
- j option
  - 디스크 상의 journal에 기록된 후 완료로 판단하는옵션
- wtimeout Option
  - primary에서 Secondary로 데이터 동기화시 timeout 설정 → 초과 시 에러(설정 단위 : msec)

```
{ w:<value> , j : <boolean>, wtimeout:<number>}
```

. 쓰기 명령어 사용법

. upsert

- upsert 명령어를 사용하여 수정할 대상이 없는 경우 insert 동작을 수행할 수 있도록 함

- Upsert 기능을 하려면 update, updateOne, updateMany, replaceOne 메소드에 옵션으로 { upsert: true } 를 사용 또는 findAndModify, findOneAndUpdate, findOneAndReplace 메소드에 upsert: true를 추가 가능

. bulkwrite

- 3.2 버전 부터 db.collection.bulkWrite() 대량 쓰기 작업을 수행하는 방법도 제공

- 작업을 순차적으로 실행, 쓰기 작업 도중 오류가 발생하면 나머지 쓰기 작업을 처리하지 않고 작업 종료

```
try {
   db.characters.bulkWrite(
      [
         { insertOne :
            {
               "document" :
               {
                  "_id" : 4, "char" : "Dithras", "class" : "barbarian", "lvl" : 4
               }
            }
         },
         { insertOne :
            {
               "document" :
               {
                  "_id" : 5, "char" : "Taeln", "class" : "fighter", "lvl" : 3
               }
            }
         },
         { updateOne :
            {
               "filter" : { "char" : "Eldon" },
               "update" : { $set : { "status" : "Critical Injury" } }
            }
         },
         { deleteOne :
            { "filter" : { "char" : "Brisbane"} }
         },
         { replaceOne :
            {
               "filter" : { "char" : "Meldane" },
               "replacement" : { "char" : "Tanys", "class" : "oracle", "lvl" : 4 }
            }
         }
      ]
   );
}
catch (e) {
   print(e);
}
```

. $isolated 연산자

- 여러 문서에 영향을 미치는 쓰기 작업이 첫 번째 도큐먼트가 작성되면 다른 읽기 또는 쓰기에 영향을주지 않음

- $ isolated 옵션 을 사용하면 작업이 완료되거나 오류가 발생할 때까지 클라이언트가 변

경 사항을 볼 수 없도록 함
  - 샤딩에서 작동하지 않음

. 배열(Array)

- 몽고디비에서는 일반적으로 도큐먼트는 **{ }**로 구분하고, 배열은**[ ]**로 구분

```
{
  fruits: [
  "banana", "apple","peach"
  ]
}
```

```
{
  # 1차원 배열
  lang: [
  "C", "C++", "Python", "Java"
  ],
  # 2차원 배열
  comment: [
    {
      name: "Jun",
      write: "hmm"
    },
    {
      name: "Lee",
      write: "Sleep"
    }
  ]
}
```

. 기존 언어처럼 몇번째 값으로 찾거나 필드명으로 검색 등 컨드롤 가능

. 배열 관련 연산자

$all

배열 내에 하나 이상의 요소가 일치하는 배열을 찾을 때 사용한다.

> db.person.find({ favor_no : { $all : [1,2,3] } })

$size

주어진 크기인 배열을 반환

> db.person.find({ favor_no : { $size : 2 } })

$slice

문서 내의 배열의 값을 지정된 수 만큼만 조회할 때 사용

> db.person.find({}, { favor_no : { $slice : 1 } })

1만큼 제외하고 나머지 두 개를 조회

> db.person.find({}, { favor_no : { $slice : [1,2] } })

1만큼을 뒤에서부터 제외하고, 앞에서부터 두 개를 조회

> db.person.find({}, { favor_no : { $slice : [-1,2] } })

$push

배열 내에 지정된 키가 존재한다면 끝에, 존재하지 않는다면 새로운 배열을 생성하여 특정 값을 추가한다.

> db.person.update({ name : "neo" }, { $push : { "contact" : { "mobile" : "010-1111-1111", "phone" : "02-111-1111" } } })

$addToSet

$push와 동일한 기능을 하지만, 중복된 데이터는 추가되지 않는다.

> db.person.update({ name : "neo" }, { $addToSet : { "emaillist" : "neo@neo.com" } })

$each

$addToSet 내에 사용되며 배열에 여러 값들을 추가한다.

> db.person.update({ name : "neo" }, { $addToSet : { "emaillist" : { $each : ["neo2@neo.com", "neo3@neo.com"] } } })

$pull

지정한 조건에 따른 배열의 요소를 제거한다.

> db.person.update({ age : 20 }, { $pull : { "emaillist" : "neo@neo.com" } })

$pop

배열의 처음이나 마지막 요소를 제거한다.

> db.person.update({ name : "neo" }, { $pop : { emaillist : 1 } }) // 마지막

> db.person.update({ name : "neo" }, { $pop : { emaillist : -1 } }) // 처음

※ 출처

https://blog.seulgi.kim/2014/09/mongodb-objectid.html : ObjectId

[MongoDB 입문] 배열로 도큐먼트 다루기|작성자 IML : 배열

http://egloos.zum.com/tiger5net/v/5709661 : 배열

https://dlaudtjr03.tistory.com/18 : Write Concern