



MONGO DB 스터디

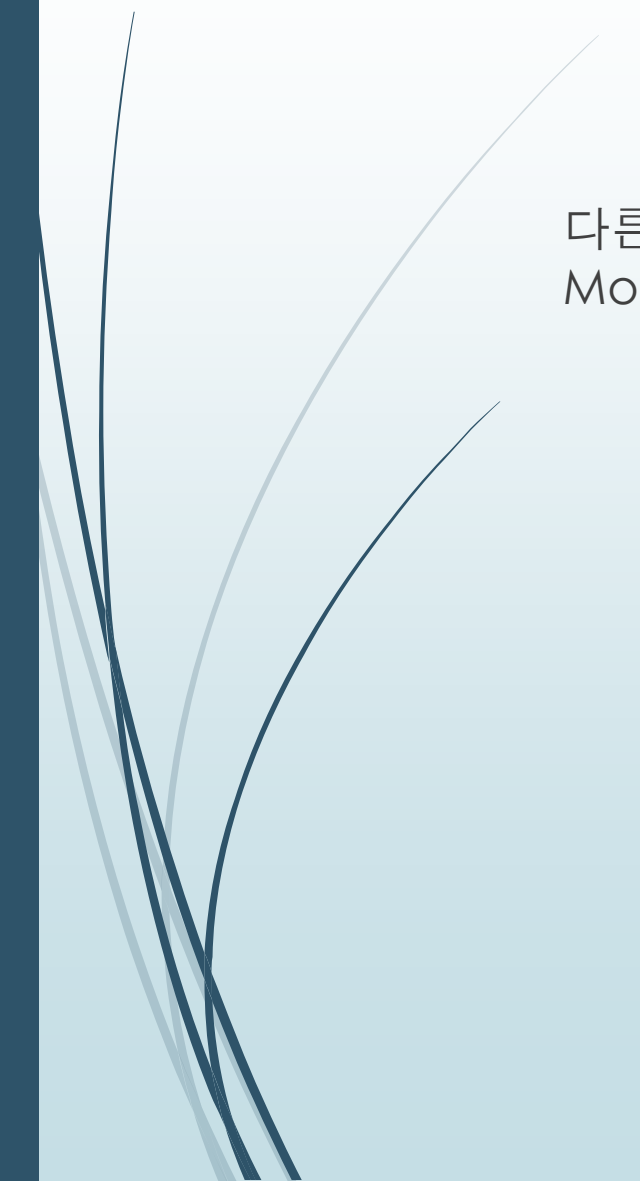
7장 업데이트 원자적 연산, 삭제

13기 엔지니어링 정우담



원자성

다른 작업이 방해하지 않음을 보장한 상태에서 문서를 검색, 업데이트 하는
Mongo DB의 기능





업데이트

MongoDB에서 업데이트를 수행하려면 두가지 방법이 가능하다

- 도큐먼트 전체를 대체
- 특정 필드만을 수정하기 위해 업데이트 연산자 사용

업데이트

1. 대치에 의한 수정

도큐먼트에 대해 질의를 하고,

클라이언트에서 수정을 하고

수정된 도큐먼트로 업데이트 명령을 내린다

```
> user_id = ObjectId("5c95adec16270a2d3077c8c9")
ObjectId("5c95adec16270a2d3077c8c9")
> doc = db.users.findOne({'_id' : user_id})
{
  "_id" : ObjectId("5c95adec16270a2d3077c8c9"),
  "last_name" : "smith",
  "age" : 30
}
> doc['region'] = 'North America'
North America
> print("updating" + user_id)
updating5c95adec16270a2d3077c8c9
> db.users.update({'_id': user_id}, doc)
WriteResult({'nMatched' : 1, 'nUpserted' : 0, 'nModified' : 1 })
> db.users.find({'age':30})
{ "_id" : ObjectId("5c95ad2916270a348052bb00"), "last_name" : "smith", "age" : 30, "email" : "maybe35xxxv@gmail.com", "gender" : "male" }
{ "_id" : ObjectId("5c95adec16270a2d3077c8c9"), "last_name" : "smith", "age" : 30, "region" : "North America" }
{ "_id" : ObjectId("5c95b11416270a37d88c9a64"), "last_name" : "smith", "age" : 30, "gender" : "male" }
```

업데이트

1. 대치에 의한 수정

```
user_id = ObjectId("5c95adec16270a2d3077c8c9")
doc = 유.users.findOne({'_id': user_id})
doc['region'] = "North America"
print("updating" + user_id)
db.users.update({'_id': user_id}, doc)
```

업데이트

2. 연산자에 의한 수정

\$set 연산자를 통해 서버에 한번의 요청으로 문서를 업데이트 한다

```
> db.users.find({age : 30})
{ "_id" : ObjectId("5c95ad2916270a348052bb00"), "last_name" : "smith", "age" : 30, "email" : "maybe35xxxv@gmail.com", "gender" : "male" }
{ "_id" : ObjectId("5c95adec16270a2d3077c8c9"), "last_name" : "smith", "age" : 30 }
{ "_id" : ObjectId("5c95b11416270a37d88c9a64"), "last_name" : "smith", "age" : 30 }
> user_id = ObjectId("5c95b11416270a37d88c9a64")
ObjectId("5c95b11416270a37d88c9a64")
> db.users.update({_id : user_id}, {$set: {gender : "male"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.users.find({age: 30})
{ "_id" : ObjectId("5c95ad2916270a348052bb00"), "last_name" : "smith", "age" : 30, "email" : "maybe35xxxv@gmail.com", "gender" : "male" }
{ "_id" : ObjectId("5c95adec16270a2d3077c8c9"), "last_name" : "smith", "age" : 30 }
{ "_id" : ObjectId("5c95b11416270a37d88c9a64"), "last_name" : "smith", "age" : 30, "gender" : "male" }
```

```
db.users.update({_id: user_id}, {$set: {gender : "male"}})
```

업데이트

두 방법 비교하기

대치에 의한 수정은 좀 더 일반적인 방식이다

어떤 속성을 수정하던지 업데이트를 수행하는 코드는 동일하다

(업데이트를 일반화하는 몽고 DB 객체 매퍼를 작성한다면 대치 방식이 합리적이다)

타겟 방식(연산자 사용)은 좀 더 나은 성능을 보인다

수정할 도큐먼트를 서버에 요청할 필요가 없다

업데이트를 지정할 도큐먼트의 크기가 상대적으로 작다

(200kb의 도큐먼트를 수정하기 위해 업데이트마다 200kb를 이동시켜야 하지만,

연산자 명령만 수행하면 수정할 도큐먼트 크기에 상관없이 100바이트 이내이다)

타겟 방식(연산자 사용)은 원자적 업데이트에 적합하다.

원자적 문서 프로세싱

findAndModify():

문서를 자동으로 업데이트 하고, 업데이트 된 문서를 한번에 반환

원자적 업데이트는, 다른 연산이 중간에 interrupt 할 수 없다.

>>다른 조작들은 원자적 업데이트가 완료될 때까지 기다려야 한다

findAndModify()로 job queue, state machine을 구축할 수 있다.

>>이를 통해 기초적인 transaction 기능 구현이 가능,

구축 가능한 어플리케이션 범위 늘어남

원자적 도큐먼트 프로세싱

주문 상태 전이

상태 전이는, “유효한 초기 상태 확인 쿼리” + “상태를 변경하는 업데이트”

두개로 나뉠 수 있다.

1.findAndModify()에서

query 명령어로 state가 “CART”인지 확인, update로 “PRE_AUTHORIZE”로 변경

```
newDoc = db.orders.findAndModify({
  query: {
    user_id: ObjectId("4c4b1476238d3b4dd5000001"),
    state: 'CART'
  },
  update: {
    $set: {
      state: 'PRE-AUTHORIZE'
    }
  },
  'new': true
})
```

원자적 도큐먼트 프로세싱

주문 상태 전이

2. 주문 확인 및 승인

승인 전 단계에서, 주문 객체를 가지고 총계를 계산하고,
새로운 총액이 이전의 총액과 같은 경우에만(query에서 TOTAL이 99000임을 확인)
PRE-AUTHORIZE에서 AUTHORIZING 으로 상태를 변경한다(update로 state 변경)

만약 승인 요청이 실패했다면(새로운 총액이 이전의 총액과 다를 경우처럼) 주문의 상태를 CART로 변경한다

```
oldDoc = db.orders.findAndModify({
  query: {
    user_id: ObjectId("4c4b1476238d3b4dd5000001"),
    total: 99000,
    state: "PRE-AUTHORIZE"
  },
  update: {
    '$set': {
      state: "AUTHORIZING"
    }
  }
})
```

원자적 도큐먼트 프로세싱

주문 상태 전이

3. 주문 완료하기

auth_doc에 승인 확인을 나타내는 샘플 도큐먼트 객체를 저장,
findAndModify에서 query로 상태가 AUTHORIZING임을 확인하고,
Update로 상태를 PRE-SHIPPING으로 변경,
authorization key에 만들어둔 auth_doc을 value로 첨부한다

```
auth_doc = {
  ts: new Date(),
  cc: 3432003948293040,
  id: 2923838291029384483949348,
  gateway: "Authorize.net"
}
db.orders.findAndModify({
  query: {
    user_id: ObjectId("4c4b1476238d3b4dd5000001"),
    state: "AUTHORIZING"
  },
  update: {
    $set: {
      state: "PRE-SHIPPING",
      authorization: auth_doc
    }
  }
})
```



Mongo DB 업데이트, 삭제 하기

업데이트 타입과 옵션

타겟 업데이트는 하나 이상의 업데이트 연산자를 사용한다

대치 업데이트는 업데이트의 쿼리 선택터와 일치하는 문서를 대치하기 위해 사용할 문서로 정의함

※ 업데이트 문서가 모호하다면 업데이트는 실패하게 된다



Mongo DB 업데이트, 삭제 하기

다중 도큐먼트 업데이트

디폴트로, 업데이트는 쿼리 선택터와 일치하는 여러 도큐먼트 중 첫번째만 업데이트가 된다. 따라서 쿼리와 일치하는 모든 도큐먼트를 업데이트 하기 위해선, 다중 업데이트라고 명시를 해줘야 한다.({multi : true} 옵션을 쿼리에 넣음)

Mongo DB 업데이트, 삭제 하기

업서트(update + insert)

아이템이 없을 때 새로 추가를 해주고 이미 존재한다면 업데이트를 하고 싶을 때
(원자적으로 업데이트 or 내가 찾고자 하는 도큐먼트가 존재하는지 모를때 유용)

쿼리 셀렉터와 일치하는 도큐먼트 ○ >> 정상적인 업데이트

쿼리 셀렉터와 일치하는 도큐먼트 X >> 새로운 도큐먼트가 만들어지고 삽입됨
(`{upsert: true}` 옵션을 쿼리에 넣어준다)

당연히도, 업서트는 한번에 하나의 도큐먼트만 삽입, 업데이트 할 수 있다

Mongo DB 업데이트, 삭제 하기

유용한 연산자들

\$inc: 수치가 증가, 감소할 때 혹은 수치를 임의로 더하거나 뺄 때 사용한다

도큐먼트의 크기를 변경하지 않고 디스크에서 해당 부분만 영향받고 적용이 된다.

Mongo DB 업데이트, 삭제하기

\$set, \$unset

도큐먼트에서 특정 key의 value를 정해줄 때 \$set을 활용한다.

(value에는 정수, 서브 도큐먼트, 배열까지 어떤 것이라도 가능하다)

키의 값이 이미 존재하면 덮어쓰우는 방식, 키가 없었다면 새로 생성되는 방식

\$unset은 도큐먼트에서 key를 삭제한다.

만약 key에 value로 배열이나 서브 도큐먼트가 있다면 닷 표기법으로 내부 객체를 지정할 수 있다.



Mongo DB 업데이트, 삭제하기

\$setOnInsert

업서트 중에 데이터를 덮어쓰지 않고, 문서가 새로운 것이고 삽입을 할 때에만 필드를 수정한다.

Mongo DB 업데이트, 삭제 하기

배열 업데이트 연산자

\$push, \$pushall, \$each

\$push를 통해 key의 value가 배열일 때 단일 요소를 맨 끝에 추가한다

\$push와 \$each로 여러 개의 요소를 배열에 추가한다

\$pushall은 몽고 DB 2.4 버전 이후 사용이 되지 않는다

Mongo DB 업데이트, 삭제하기

\$slice

찾은 업데이트를 해야 할 때, 보다 쉽게 배열을 관리할 수 있다
\$push, \$each 연산자와 함께 사용한다.

\$slice: n

n: 정수

음수일 경우, 앞에서부터 제거

양수일 경우, 뒤에서부터 제거

n의 절대값만큼 배열에서 값을 남긴다

Mongo DB 업데이트, 삭제 하기

\$sort

배열 업데이트에 도움을 주기 위해서 사용, 분할하기 전에 문서를 정렬시 사용

`{ $sort : {key이름: 1} }`

오름차순 정렬

Mongo DB 업데이트, 삭제하기

\$addToSet

\$addToSet은 배열에 중복되는 값이 존재하지 않을 때에만 연산이 수행된다

\$each에서 지정한 값들 중 key 안에 value로 존재하지 않는 값들만 들어가게 된다
(※ \$each는 \$addToSet, \$push 연산자에서만 같이 사용될 수 있다)



Mongo DB 업데이트, 삭제하기

\$pop

마지막으로 첨가된 아이템을 삭제한다(1: 마지막 value, -1: 처음 value)

삭제 한 value를 return하지는 않는다(스택에서의 pop과는 다름)

Mongo DB 업데이트, 삭제하기

\$bit

비트 단위에서 OR 연산과 AND 연산을 가능하게 한다

\$pull, \$pullAll

\$pull은 \$pop과 유사하나, 배열에서의 위치가 아니라 삭제하고 싶은 값을 지정해서 value를 삭제한다

(※ 삭제하고 싶은 값을 쿼리의 형태로 지정해 주는 것도 가능하다)

\$pullAll은 여러 개의 value를 리스트의 형태로 동시에 삭제할 수 있다.



Mongo DB 업데이트, 삭제 하기

위치 업데이트

위치 연산자는 쿼리 선택터에서 닷 표기법을 활용해서
식별한 배열 내의 서브도큐먼트를 업데이트 할 수 있다.

Mongo DB 업데이트, 삭제하기

findAndModify()

findAndModify()를 사용할 때 필요한 옵션들이 있다

query- 도큐먼트 쿼리 셀렉터이다. (default: {})

update- 업데이트할 내용을 지정하는 도큐먼트 (default: {})

remove- true면 객체를 삭제, 삭제한 객체를 반환한다(default : false)

new - true면 업데이트를 수행한 도큐먼트를 반환한다 (default: false)

sort- 정렬의 방향을 결정

fields- 필요한 필드 일부만 지정하고 싶을 때 사용

upsert- true일때, findAndModify를 upsert처럼 쿼리로 찾을 수 없는 도큐먼트라면 그대로 생성해 버린다. 새로 생성한 도큐먼트를 반환하려면 {new: true}를 지정

Mongo DB 업데이트, 삭제하기

삭제

컬렉션 전체를 삭제할 수도 있고,

remove 메서드에 쿼리 선택터를 매개변수로 넘겨서 일부분의 도큐먼트 삭제도 가능

동시성, 원자성, 고립

몽고 DB는 잠금 전략을 통해 한 연산이 다른 연산에 개입을 할 수 없게 하고(원자성)

하나의 쓰기 혹은 다수의 읽기 연산을 수행할 수 있지만 두 작업을 동시에 할 수는 없다(동시성)

만약, 양보를 해서는 안되는 연산이 존재한다면???

\$isolated(): 여러 도큐먼트의 업데이트를 할 때 다른 연산이 끼어드는 것을 허용 X