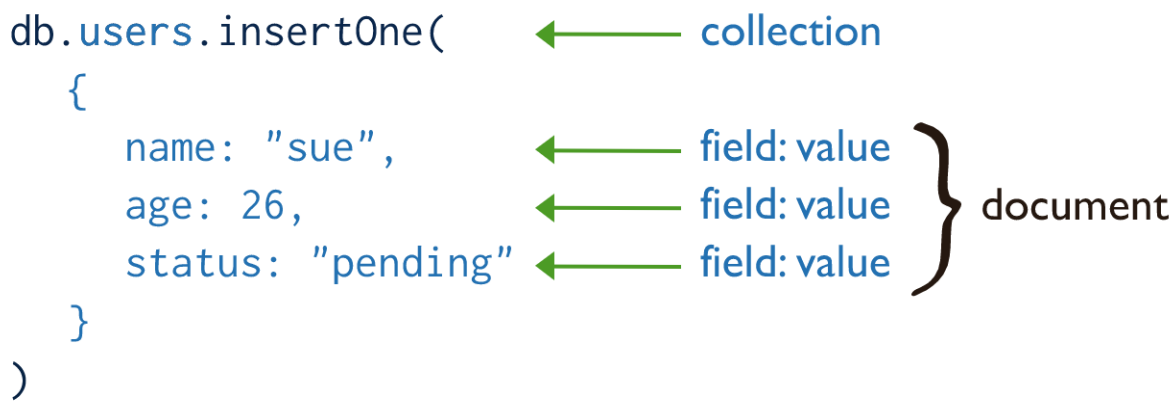
1. mongoDB 데이터 입력/검색/수정/삭제 (CRUD)

1.1 Document 입력 - insertOne, insertMany

- insertOne : 한개의 document 생성
- insertMany : list of document 생성

Document 입력 문법



SQL INSERT 문법과 비교

SQL INSERT Statements	MongoDB insertOne() Statements
`INSERT INTO people(user_id, age, status) VALUES ("bcd001", 45, "A")`	`db.people.insertOne( { user_id: "bcd001", age: 45, status: "A" } )`

- insertOne 예제

```
db.articles.insertOne(
     { subject: "coffee", author: "xyz", views: 50 }
)
```

- insertMany 예제

```
db.articles.insertMany(
   [
     { subject: "coffee", author: "xyz", views: 50 },
     { subject: "Coffee Shopping", author: "efg", views: 5 },
     { subject: "Baking a cake", author: "abc", views: 90  },
     { subject: "baking", author: "xyz", views: 100 },
     { subject: "Café Con Leche", author: "abc", views: 200 },
     { subject: "Сырники", author: "jkl", views: 80 },
     { subject: "coffee and cream", author: "efg", views: 10 },
```

```
        { subject: "Cafe con Leche", author: "xyz", views: 10 },
        { subject: "coffees", author: "xyz", views: 10 },
        { subject: "coffee1", author: "xyz", views: 10 }
      ]
  )
```

실습

1. employees Collection 생성 {capped:true, size:100000} Capped Collection, size는 100000 으로 생성

2. 다음 Document 데이터 넣기

- user_id : AB001, age : 45, status : A (Document)
- user_id : AB002, age : 25, status : B (Document)
- user_id : AA003, age : 50, status : A (Document)
- user_id : AA004, age : 35, status : A (Document)
- user_id : BB001, age : 28, status : B (Document)

1.2 Document 읽기(검색) - findOne, find

- findOne : 매칭되는 한개의 document 검색
- find : 매칭되는 list of document 검색

Document 읽기(검색) 문법

- find()/findOne 명령과 - SQL 문 비교

```
db.people.find() - SELECT * FROM people
db.people.find({ }, { user_id: 1, status: 1 }) - SELECT _id,
user_id, status FROM people
db.people.find({ },{ user_id: 1, status: 1, _id: 0 }) - SELECT
user_id, status FROM people
db.people.find({ status: "A" }) - SELECT * FROM people WHERE
status = "A"
db.people.find({ status: "A", age: 50 }) - SELECT * FROM people
WHERE status = "A" AND age = 50
db.people.find({ $or: [ { status: "A" } , { age: 50 } ] }) -
SELECT * FROM people WHERE status = "A" OR age = 50
```

실습

1. employees Collection 에서 user_id 가 bcd002 인 Document의 user_id, age, status, _id 출력

2. employees Collection 에서 user_id 가 bcd003 인 Document의 user_id, age, status 출력

3. employees Collection 에서 user_id 가 bcd004 이거나, age가 28인 Document 의 모든 필드 출력

비교 문법

```
$eq     =    Matches values that are equal to a specified value.
$gt     >    Matches values that are greater than a specified
value.
$gte    >=   Matches values that are greater than or equal to a
specified value.
$in          Matches any of the values specified in an array.
$lt     <    Matches values that are less than a specified
value.
$lte    <=   Matches values that are less than or equal to a
specified value.
$ne     !=   Matches all values that are not equal to a
specified value.
$nin         Matches none of the values specified in an array.
```

비교 문법 코드 예제

```
db.people.find({ age: { $gt: 25 } }) - SELECT * FROM people
WHERE age > 25
db.people.find({ age: { $lt: 25 } }) - SELECT * FROM people
WHERE age < 25
db.people.find({ age: { $gt: 25, $lte: 50 } }) - SELECT * FROM
people WHERE age > 25 AND age <= 50
db.people.find( { age: { $nin: [ 5, 15 ] } } )) - SELECT * FROM
people WHERE age = 5 or age = 15
db.people.find( { user_id: /bc/ } )
db.people.find( { user_id: { $regex: /bc/ } } )
                                                    - SELECT *
FROM people WHERE user_id like "%bc%"
db.people.find( { user_id: /^bc/ } )
db.people.find( { user_id: { $regex: /^bc/ } } )
                                                    - SELECT *
FROM people WHERE user_id like "bc%"
db.people.find( { status: "A" } ).sort( { user_id: 1 } ) -
SELECT * FROM people WHERE status = "A" ORDER BY user_id ASC
db.people.find( { status: "A" } ).sort( { user_id: -1 } ) -
SELECT * FROM people WHERE status = "A" ORDER BY user_id DESC
db.people.count()
db.people.find().count()
                                                    - SELECT
COUNT(*) FROM people
db.people.count( { user_id: { $exists: true } } )
db.people.find( { user_id: { $exists: true } } ).count()
                                                    - SELECT
COUNT(user_id) FROM people
db.people.count( { age: { $gt: 30 } } )
db.people.find( { age: { $gt: 30 } } ).count()
                                                    - SELECT
COUNT(*) FROM people WHERE age > 30
db.people.distinct( "status" ) - SELECT DISTINCT(status) FROM
people
db.people.findOne()
db.people.find().limit(1)
                                                    - SELECT *
FROM people LIMIT 1
```

실습

1. 다음 Document 데이터 넣기
- age 가 20 보다 큰 Document 의 user_id 만 출력하기
- age 가 50 이고 status 가 A 인 Document 의 user_id 만 출력하기
- age 가 60 보다 작은 Document 의 user_id 와 age 출력하기
- user_id 종류 출력하기
- user_id 가 bcd 로 시작하는 전체 Document 출력하기

Document 수정 - updateOne, updateMany

- updateOne - 매칭되는 한개의 document 업데이트
- updateMany - 매칭되는 list of document 업데이트

1.3 Document 수정 문법

- set : $field$값설정—inc: field 값을 증가시키거나, 감소시킴 - 예: $inc: { age: 2 } - age 값을 본래의 값에서 2 증가

Document 수정 코드 예제

```
- db.people.updateMany( { age: { $gt: 25 } }, { $set: { status:
"C" } } )
- SQL 변환하면,
  - UPDATE people SET status = "C" WHERE age > 25
- 한 Document만 수정하려면 updateOne을 사용함
- db.people.updateOne( { age: { $gt: 25 } }, { $set: { status:
"C" } } )
- db.people.updateMany( { status: "A" } , { $inc: { age: 3 } } )
- SQL 변환하면,
  - UPDATE people SET age = age + 3 WHERE status = "A"
```

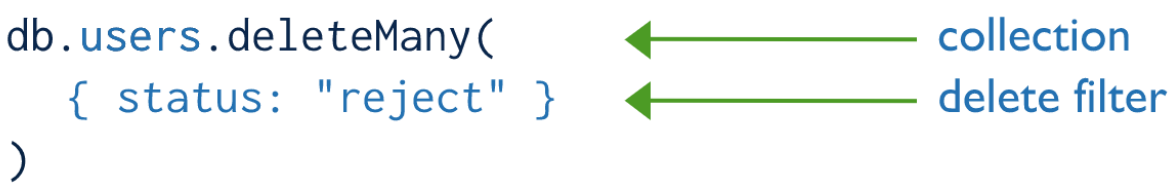
실습

1. 다음 Document 데이터 수정하기
- age 가 40 보다 큰 Document 의 status 를 B 로 변환하기

1.4 Document 삭제 - removeOne, removeMany

- removeOne - 매칭되는 한개의 document 삭제
- removeMany - 매칭되는 list of document 삭제

Document 삭제 문법



- db.people.deleteMany({ status: "D" })
- SQL로 변환하면,
  - DELETE FROM people WHERE status = "D"
- db.people.deleteMany({})
- SQL로 변환하면,
  - DELETE FROM people

실습

1. 다음 Document 데이터 삭제하기
- age 가 30 보다 작은 Document 삭제하기

참고: mongo shell

- 로컬에서 서버가 돌아갈 경우,
  - mongo
- 원격 서버에 접속할 경우
  - mongo --host 'host_address' --port 'port'
  - 예) mongo --host 192.10.21.3 --port 27017

2. 파이썬으로 mongoDB 제어하기 - pymongo 라이브러리

- mongodb python module
- https://api.mongodb.com/python/current/
- pip install pymongo

참고

1. pymongo 라이브러리 import
2. mongodb 접속 (주소)
3. 내가 사용할 database, collection 생성 또는 선택
4. 해당 database의 collection에 CRUD 명령하는 방법

mongodb with using pymongo (코드 실행 실습)

In [49]:
```
import pymongo
#conn = pymongo.MongoClient()
```

2.1 연결하기

In [50]:
```
import pymongo
```

In [54]:
```
#mongo DB 사용자 no인증 연결
#host_info1 = 'mongodb://127.0.0.1'
#conn = pymongo.MongoClient(host_info1)
#mongo_server = 'mongodb://myUser:password1234@127.0.0.1'
#conn = pymongo.MongoClient(mongo_server, 27017)
#mongo DB 사용자 인증 연결
#host_info2 = 'mongodb://myUser:password1234@127.0.0.1:27017'
host_info2 = 'mongodb://rapa00:1234@127.0.0.1:27017'
conn = pymongo.MongoClient(host_info2)
print(conn)
```

```
MongoClient(host=['127.0.0.1:27017'], document_class=dict, tz_aware=False, connect=True)
```

2.2 mydbs Database 사용하기

In [55]:
```
knowledge = conn.mydbs
```

In [57]:
```
knowledge = conn["mydbs"]    # 이렇게도 가능하다.
```

In [58]:
```
print(knowledge)
```

```
Database(MongoClient(host=['127.0.0.1:27017'], document_class=dict, tz_aware=False, conn
ect=True), 'mydbs')
```

In [59]:
```
print(dir(knowledge))
```

```
['_BaseObject__codec_options', '_BaseObject__read_concern', '_BaseObject__read_preferenc
e', '_BaseObject__write_concern', '_Database__client', '_Database__incoming_copying_mani
pulators', '_Database__incoming_manipulators', '_Database__name', '_Database__outgoing_c
opying_manipulators', '_Database__outgoing_manipulators', '__call__', '__class__', '__de
lattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattr
__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__init__', '__init_subcla
ss__', '__iter__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__next__', '_
_reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subc
lasshook__', '__weakref__', '_apply_incoming_copying_manipulators', '_apply_incoming_man
ipulators', '_command', '_create_or_update_user', '_current_op', '_default_role', '_fix_
incoming', '_fix_outgoing', '_list_collections', '_read_preference_for', '_retryable_rea
d_command', '_write_concern_for', 'add_son_manipulator', 'add_user', 'aggregate', 'authe
nticate', 'client', 'codec_options', 'collection_names', 'command', 'create_collection',
'current_op', 'dereference', 'drop_collection', 'error', 'eval', 'get_collection', 'inco
ming_copying_manipulators', 'incoming_manipulators', 'last_status', 'list_collection_nam
es', 'list_collections', 'logout', 'name', 'next', 'outgoing_copying_manipulators', 'out
going_manipulators', 'previous_error', 'profiling_info', 'profiling_level', 'read_concer
n', 'read_preference', 'remove_user', 'reset_error_history', 'set_profiling_level', 'sys
tem_js', 'validate_collection', 'watch', 'with_options', 'write_concern']
```

In [60]:
```
print(knowledge.name)
```

```
mydbs
```

2.3 test_collection 이라는 collection 사용하기 (없으면 만들어진다.)

In [65]:
```
knowledge_it = knowledge.it_collection
```

In [69]:
```
knowledge_it = knowledge["it_collection"]
```

In [70]:
```
knowledge_it
```

Out[70]:
```
Collection(Database(MongoClient(host=['127.0.0.1:27017'], document_class=dict, tz_aware=
False, connect=True), 'mydbs'), 'it_collection')
```

In [71]:
```
knowledge_it.name
```

Out[71]:
```
'it_collection'
```

2.4 Document INSERT 하기 (insert_one() 과 insert_many())

- insert_one()
  - mongodb shell 명령어: insertOne()

In [38]:
```
post = {"author": "Mike", "text": "My first blog post!", "tags": ["mongodb", "python", "pymo
```

In [39]:
```
knowledge_it.insert_one(post)
```

Out[39]:
```
<pymongo.results.InsertOneResult at 0x1048e9d48>
```

In [40]:
```
knowledge_it.insert_one( { "author":"joy Lee", "age":45 } )
```

Out[40]:
```
<pymongo.results.InsertOneResult at 0x1048e9bc8>
```

- insert_many()

In [43]:
```
knowledge_it.insert_many(
    [
        { "author":"joy Ahn", "age":25 },
        { "author":"joy", "age":35 }
    ]
)
```

Out[43]:
```
<pymongo.results.InsertManyResult at 0x1048e9a08>
```

- Document INSERT 하면, _id (primary key)를 확인하는 방법

In [49]:
```
post = {"author": "Joy", "text": "My first blog post!"}
```

In [50]:
```
post_id = knowledge_it.insert_one(post)
```

In [51]:
```
post_id
```

Out[51]:
```
<pymongo.results.InsertOneResult at 0x1042fce48>
```

In [52]:
```
post_id.inserted_id
```

Out[52]:
```
ObjectId('5d32a3abc92b6508c3f5d306')
```

- estimated_document_count() 메서드는 컬렉션 객체와 함께 쓰여서 총 Document 수를 알려줌
  - count_documents({})
  - count() 함수는 최신 pymongo 라이브러리에서는 사용 권장되지 않음

In [61]:
```
knowledge_it.count_documents({})
```

Out[61]:
```
6
```

- list와 dictionary 를 활용하여 insert 하기

In [63]:
```
# 리스트, 객체 삽입 가능
knowledge_it.insert_one({'title' : '암살', 'castings' : ['이정재', '전지현', '하정우']})
knowledge_it.insert_one(
    {
        'title' : '실미도',
        'castings' : ['설경구', '안성기'],
        'datetime' :
        {
            'year' : '2003',
            'month' : 3,
            'val' :
            {
                'a' :
                {
                    'b' : 1
                }
            }
        }
    }
)
```

Out[63]:
```
<pymongo.results.InsertOneResult at 0x1048e9908>
```

In [64]:
```
data = list()
data.append({'name' : 'aaron', 'age' : 20})
data.append({'name' : 'bob', 'age' : 30})
data.append({'name' : 'cathy', 'age' : 25})
data.append({'name' : 'david', 'age' : 27})
data.append({'name' : 'erick', 'age' : 28})
data.append({'name' : 'fox', 'age' : 32})
data.append({'name' : 'hmm'})

knowledge_it.insert_many(data)
```

Out[64]:
```
<pymongo.results.InsertManyResult at 0x104975188>
```

In [65]:
```
knowledge_it.estimated_document_count()
```

Out[65]:
```
17
```

2.5 Document 검색 하기(읽기) (find_one() 과 find())

- find_one() 메서드 : 가장 빨리 검색되는 하나 검색하기

In [69]:
```
knowledge_it.find_one()
```

Out[69]:
```
{'_id': ObjectId('5d329c5fc92b6508c3f5d300'),
 'author': 'Mike',
 'text': 'My first blog post!',
 'tags': ['mongodb', 'python', 'pymongo']}
```

In [1]:
```
joy = knowledge_it.find_one( {"author":"Joy"} )
joy
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_10913/2285641812.py in <module>
----> 1 joy = knowledge_it.find_one( {"author":"Joy"} )
      2 joy

NameError: name 'knowledge_it' is not defined
```

In [ ]:
```
* find_one( 안에 조건을 넣을 때는 사전 형식으로 해야 합니다. { 키:값 } )
```

- find() 메서드 : 검색되는 모든 Document 읽어오기

In [73]:
```
docs = knowledge_it.find()
```

In [74]:
```
for doc in docs:
    print(doc)
```

```
{'_id': ObjectId('5d329c5fc92b6508c3f5d300'), 'author': 'Mike', 'text': 'My first blog p
ost!', 'tags': ['mongodb', 'python', 'pymongo']}
{'_id': ObjectId('5d329d62c92b6508c3f5d301'), 'author': 'Dave Lee', 'age': 45}
{'_id': ObjectId('5d329da9c92b6508c3f5d302'), 'author': 'Dave Lee', 'age': 45}
{'_id': ObjectId('5d329de7c92b6508c3f5d303'), 'author': 'Dave Ahn', 'age': 25}
{'_id': ObjectId('5d329de7c92b6508c3f5d304'), 'author': 'Dave', 'age': 35}
{'_id': ObjectId('5d32a3abc92b6508c3f5d306'), 'author': 'Dave', 'text': 'My first blog p
ost!'}
{'_id': ObjectId('5d32a6fdc92b6508c3f5d307'), 'title': '암살', 'castings': ['이정재', '전
지현', '하정우']}
{'_id': ObjectId('5d32a6fdc92b6508c3f5d308'), 'title': '실미도', 'castings': ['설경구',
'안성기'], 'datetime': {'year': '2003', 'month': 3, 'val': {'a': {'b': 1}}}}
{'_id': ObjectId('5d32a7acc92b6508c3f5d309'), 'title': '암살', 'castings': ['이정재', '전
지현', '하정우']}
{'_id': ObjectId('5d32a7acc92b6508c3f5d30a'), 'title': '실미도', 'castings': ['설경구',
'안성기'], 'datetime': {'year': '2003', 'month': 3, 'val': {'a': {'b': 1}}}}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30b'), 'name': 'aaron', 'age': 20}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30c'), 'name': 'bob', 'age': 30}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30d'), 'name': 'cathy', 'age': 25}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30e'), 'name': 'david', 'age': 27}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30f'), 'name': 'erick', 'age': 28}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d310'), 'name': 'fox', 'age': 32}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d311'), 'name': 'hmm'}
```

In [75]:
```
docs = knowledge_it.find( {"author":"Joy"} )
```

In [76]:
```
for doc in docs:
    print(doc)
```

```
{'_id': ObjectId('5d329de7c92b6508c3f5d304'), 'author': 'Dave', 'age': 35}
{'_id': ObjectId('5d32a3abc92b6508c3f5d306'), 'author': 'Dave', 'text': 'My first blog p
ost!'}
```

- count_documents() 함수로 조건에 맞는 검색 데이터 갯수 알아내기

In [82]:
```
knowledge_it.count_documents({"author": "Joy"})
```

Out[82]:
```
2
```

- sort() 와 함께 쓰기

In [86]:
```
for post in knowledge_it.find().sort("age"):
    print(post)
```

```
{'_id': ObjectId('5d329c5fc92b6508c3f5d300'), 'author': 'Mike', 'text': 'My first blog p
ost!', 'tags': ['mongodb', 'python', 'pymongo']}
{'_id': ObjectId('5d32a3abc92b6508c3f5d306'), 'author': 'Dave', 'text': 'My first blog p
ost!'}
{'_id': ObjectId('5d32a6fdc92b6508c3f5d307'), 'title': '암살', 'castings': ['이정재', '전
지현', '하정우']}
{'_id': ObjectId('5d32a6fdc92b6508c3f5d308'), 'title': '실미도', 'castings': ['설경구',
'안성기'], 'datetime': {'year': '2003', 'month': 3, 'val': {'a': {'b': 1}}}}
{'_id': ObjectId('5d32a7acc92b6508c3f5d309'), 'title': '암살', 'castings': ['이정재', '전
지현', '하정우']}
{'_id': ObjectId('5d32a7acc92b6508c3f5d30a'), 'title': '실미도', 'castings': ['설경구',
'안성기'], 'datetime': {'year': '2003', 'month': 3, 'val': {'a': {'b': 1}}}}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d311'), 'name': 'hmm'}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30b'), 'name': 'aaron', 'age': 20}
{'_id': ObjectId('5d329de7c92b6508c3f5d303'), 'author': 'Dave Ahn', 'age': 25}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30d'), 'name': 'cathy', 'age': 25}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30e'), 'name': 'david', 'age': 27}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30f'), 'name': 'erick', 'age': 28}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d30c'), 'name': 'bob', 'age': 30}
{'_id': ObjectId('5d32a7e7c92b6508c3f5d310'), 'name': 'fox', 'age': 32}
{'_id': ObjectId('5d329de7c92b6508c3f5d304'), 'author': 'Dave', 'age': 35}
{'_id': ObjectId('5d329d62c92b6508c3f5d301'), 'author': 'Dave Lee', 'age': 45}
{'_id': ObjectId('5d329da9c92b6508c3f5d302'), 'author': 'Dave Lee', 'age': 45}
```

2.6 Document Update 하기 (update_one() 과 update_many())

- update_one() : 가장 먼저 검색되는 한 Document만 수정하기

In [92]:
```
knowledge_it.find_one( {"author":"Joy"} )
```

Out[92]:
```
{'_id': ObjectId('5d329de7c92b6508c3f5d304'),
 'author': 'Dave',
 'age': 40,
 'text': 'Hi Dave'}
```

In [91]:
```
knowledge_it.update_one( { "author" : "Joy" },
    { "$set" :
        { "text" : "Hi Joy" }
```

```
    }
)
```

Out[91]: <pymongo.results.UpdateResult at 0x1048dd948>

In [101…
```
docs = knowledge_it.find( {"author":"Joy Lee"} )
```

In [102…
```
for doc in docs:
    print(doc)
```

```
{'_id': ObjectId('5d329d62c92b6508c3f5d301'), 'author': 'Dave Lee', 'age': 30}
{'_id': ObjectId('5d329da9c92b6508c3f5d302'), 'author': 'Dave Lee', 'age': 30}
{'_id': ObjectId('5d329de7c92b6508c3f5d304'), 'author': 'Dave Lee', 'age': 30, 'text':
'Hi Dave'}
{'_id': ObjectId('5d32a3abc92b6508c3f5d306'), 'author': 'Dave Lee', 'text': 'My first bl
og post!', 'age': 30}
```

- update_many() : 조건에 맞는 모든 Document 수정하기

In [100…
```
knowledge_it.update_many( {"author": "Joy Lee"}, {"$set": { "age": 30}})
```

Out[100… <pymongo.results.UpdateResult at 0x1047c6b88>

2.7 Document 삭제 하기 (delete_one() 과 delete_many())

- delete_one() 메서드 : 가장 먼저 검색되는 한 Document만 삭제하기

In [103…
```
docs = knowledge_it.find( {"author":"Joy Lee"} )
```

In [104…
```
for doc in docs:
    print(doc)
```

```
{'_id': ObjectId('5d329d62c92b6508c3f5d301'), 'author': 'Dave Lee', 'age': 30}
{'_id': ObjectId('5d329da9c92b6508c3f5d302'), 'author': 'Dave Lee', 'age': 30}
{'_id': ObjectId('5d329de7c92b6508c3f5d304'), 'author': 'Dave Lee', 'age': 30, 'text':
'Hi Dave'}
{'_id': ObjectId('5d32a3abc92b6508c3f5d306'), 'author': 'Dave Lee', 'text': 'My first bl
og post!', 'age': 30}
```

In [105…
```
knowledge_it.delete_one( {"author":"Joy Lee"} )
```

Out[105… <pymongo.results.DeleteResult at 0x1048dd5c8>

In [108…
```
docs = knowledge_it.find( {"author":"Joy Lee"} )
```

In [109…
```
for doc in docs:
    print(doc)
```

```
{'_id': ObjectId('5d329da9c92b6508c3f5d302'), 'author': 'Dave Lee', 'age': 30}
{'_id': ObjectId('5d329de7c92b6508c3f5d304'), 'author': 'Dave Lee', 'age': 30, 'text':
'Hi Dave'}
```

```
{'_id': ObjectId('5d32a3abc92b6508c3f5d306'), 'author': 'Dave Lee', 'text': 'My first bl
og post!', 'age': 30}
```

- delete_many() 메서드 : 조건에 맞는 모든 Document 삭제하기

In [110…
```
knowledge_it.delete_many( {"author":"Joy Lee"} )
```

Out[110…
```
<pymongo.results.DeleteResult at 0x104985248>
```

In [112…
```
knowledge_it.count_documents( {"author":"Joy Lee"} )
```

Out[112…
```
0
```

In [113…
```
import pymongo
conn = pymongo.MongoClient()
books = conn.books
it_book = books.it_book

data = list()
for index in range(100):
    data.append({"author":"Joy Lee", "publisher":"fun-coding.org", "number": index })
```

In [115…
```
# CRUD - Create(Insert)
it_book.insert_many(data)
```

Out[115…
```
<pymongo.results.InsertManyResult at 0x104985508>
```

In [121…
```
# CRUD - Read
docs = it_book.find()
for doc in docs:
    print (doc)
```

```
{'_id': ObjectId('5d32bc4fc92b6508c3f5d313'), 'author': 'Dave Lee', 'publisher': 'www.fu
n-coding.org', 'number': 0}
{'_id': ObjectId('5d32bc4fc92b6508c3f5d314'), 'author': 'Dave Lee', 'publisher': 'www.fu
n-coding.org', 'number': 1}
{'_id': ObjectId('5d32bc4fc92b6508c3f5d315'), 'author': 'Dave Lee', 'publisher': 'www.fu
n-coding.org', 'number': 2}
{'_id': ObjectId('5d32bc4fc92b6508c3f5d316'), 'author': 'Dave Lee', 'publisher': 'www.fu
n-coding.org', 'number': 3}
{'_id': ObjectId('5d32bc4fc92b6508c3f5d317'), 'author': 'Dave Lee', 'publisher': 'www.fu
n-coding.org', 'number': 4}
{'_id': ObjectId('5d32bc4fc92b6508c3f5d318'), 'author': 'Dave Lee', 'publisher': 'www.fu
n-coding.org', 'number': 5}
```

In [118…
```
# CRUD - Update
it_book.update_many( {}, { "$set": { "publisher":"www.fun-coding.org"} } )
```

Out[118…
```
<pymongo.results.UpdateResult at 0x104999e48>
```

실습

number 가 6 이상(>=)인 doc 삭제하기

In [120…
```
# CRUD - Delete
```

```
it_book.delete_many( { "number": { "$gte": 6} } )
```

Out[120… <pymongo.results.DeleteResult at 0x1047eeec8>