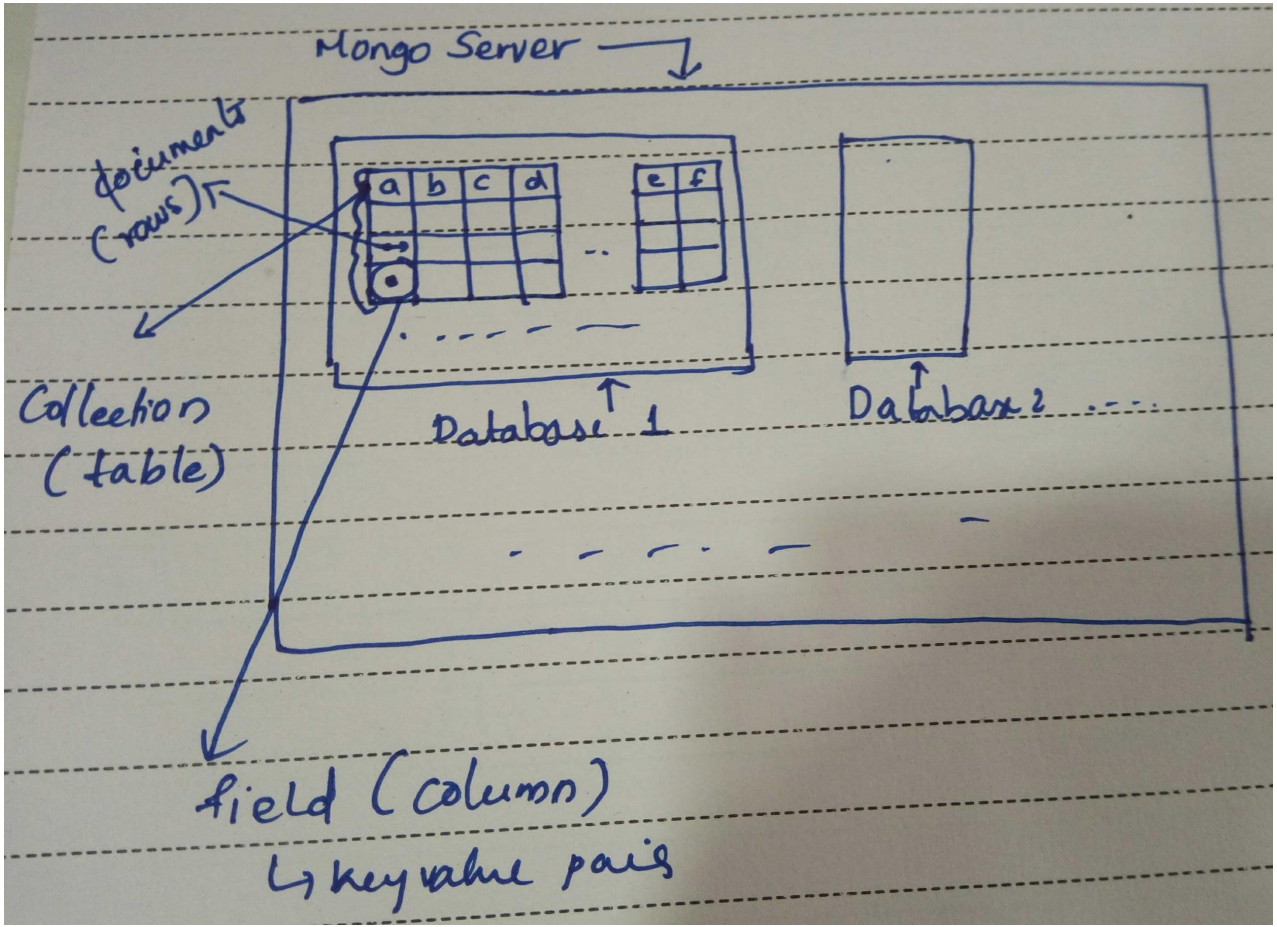
MongoDB

Self Study

Intro about MongoDB

- Works on concept of collection and document
- A MongoDB server can have multiple databases
- Database
  - contains collections
- Collection
  - group of MongoDB documents
  - exists within a single database
  - do not enforce a schema
- Document
  - set of key-value pairs
  - documents in the same collection may not have the same set of fields or structure, and common fields in a collection's documents may hold different types of data

Inside MongoDB Server



RDBMS vs MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

Create Database

To create a DB:

- *use DATABASE_NAME*

To see the current DB:

- *db*

To list all DBs in server:

- *show dbs*

To insert document:

- *db.collection_name.insert(doc_dict)*

```
> use school_db
switched to db school_db
> db
school_db
> show dbs
admin                   0.000GB
local                   0.000GB
movie                   0.000GB
mql_db_development      0.014GB
mql_development         0.000GB
test                    0.000GB
> db.student.insert({'name':'sreelakshmi'})
WriteResult({ "nInserted" : 1 })
> show dbs
admin                   0.000GB
local                   0.000GB
movie                   0.000GB
mql_db_development      0.014GB
mql_development         0.000GB
school_db               0.000GB
test                    0.000GB
>
```

DBs will appear in list only after adding a document

Importing PyMongo Client Library and Creating DB

```
import pymongo
from pymongo import MongoClient
```

```
client = MongoClient()
```

To create database in mongo

```
mydatabase = client['config']#to create a db
```

```
client.list_database_names() #show dbs
```

```
[u'local', u'mql_db_development']
```

Drop Database

To drop a DB:

- *db.dropDatabase()*

```
> use movies
switched to db movies
> db
movies
> db.dropDatabase()
{ "ok" : 1 }
> show dbs
admin                0.000GB
config               0.000GB
local                0.000GB
mql_db_development   0.014GB
>
```

Deletes the currently selected Database

Drop Database using PyMongo

To delete database

```
mydatabase = client['dummy']
mycollection1 = mydatabase['collection2']
mycollection1.insert_one({"a":1})
```

```
<pymongo.results.InsertOneResult at 0x7f47183337e8>
```

```
client.list_database_names() #show dbs
```

```
[u'config', u'dummy', u'local', u'mql_db_development']
```

```
client.drop_database('dummy')
```

```
client.list_database_names()
```

```
[u'config', u'local', u'mql_db_development']
```

Create Collection

To create a Collection:

- *db.createCollection(name, options)*

To list all Collections in current DB:

- *show collections*

Creates a collection automatically when you insert some document:

- *db.collection_name.insert(doc_dict)*

Creates fixed size collection

Max size in bytes for a capped collection

Max no. of docs allowed in capped collection

```
>
> use school_db
switched to db school_db
> show collections
> db.createCollection("student", {capped:true, size:1000, max:100})
{ "ok" : 1 }
> show collections
student
> db.teacher.insert({"name":"teacher name"})
WriteResult({ "nInserted" : 1 })
> show collections
student
teacher
> _
```

Create Collection using PyMongo

To create collection in mongo

```
mycollection = mydatabase['collection1'] #to create a collection
```

To View all colection in DB

```
mydatabase.collection_names()
```

```
[u'collection1']
```

Drop Collection

To drop a Collection in DB:

- *db.collection_name.drop()*

```
>
> db
school_db
> show collections
student
teacher
> db.teacher.drop()
true
> show collections
student
>
```

Drop Collection using PyMongo

```
mydatabase.collection_names()
```

```
[u'collection1']
```

To delete a collection in DB

```
mydatabase.drop_collection('collection1')
```

```
{u'nIndexesWas': 1, u'ns': u'config.collection1', u'ok': 1.0}
```

```
mydatabase.collection_names()
```

```
[]
```

Supported Data Types

- String
- Integer
- Boolean
- Double
- Min/ Max keys
- Arrays
- Timestamp
- Object
- Null
- Symbol
- Date
- Object ID
- Binary data
- Code
- Regular expression

Insert Document

To Insert a document to a Collection:

- *db.collection_name.insert(doc_dict)*
- *db.post.save(doc_dict)*

If "_id" in doc_dict - save( ) works same as insert( )

Else, replace the existing document with same "_id" with new one

```
> use school_db
switched to db school_db
> db.student.insert({name:"sreelakshmi"})
WriteResult({ "nInserted" : 1 })
> db.student.insert([{name:"raju"},{name:"radha"}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.student.save({name:"sreetha"})
WriteResult({ "nInserted" : 1 })
> db.student.find()
{ "_id" : ObjectId("5cefa9cceb5966f8b639cbc8"), "name" : "sreelakshmi" }
{ "_id" : ObjectId("5cefa9d4eb5966f8b639cbc9"), "name" : "raju" }
{ "_id" : ObjectId("5cefa9d4eb5966f8b639cbca"), "name" : "radha" }
{ "_id" : ObjectId("5cefa9d8eb5966f8b639cbcb"), "name" : "sreetha" }
> db.student.save({_id : ObjectId("5cefa9cceb5966f8b639cbc8"), name : "tom" })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find()
{ "_id" : ObjectId("5cefa9cceb5966f8b639cbc8"), "name" : "tom" }
{ "_id" : ObjectId("5cefa9d4eb5966f8b639cbc9"), "name" : "raju" }
{ "_id" : ObjectId("5cefa9d4eb5966f8b639cbca"), "name" : "radha" }
{ "_id" : ObjectId("5cefa9d8eb5966f8b639cbcb"), "name" : "sreetha" }
>
```

Multiple docs inserts

Replaces existing doc using save ( )

Insert Document using PyMongo

To insert document in the collection

```
doc1 = {"dataset": "cdc_data",
              "db_date": datetime.datetime(2009, 11, 12, 11, 14)}
mycollection.insert_one(doc1)
```

```
<pymongo.results.InsertOneResult at 0x7f471834ea28>
```

```
client.list_database_names() #show dbs
```

```
[u'config', u'local', u'mql_db_development']
```

To insert multiple documents in a collection

```
docs=  [{"dataset": "cdc_gender",
              "db_date": datetime.datetime(2007, 11, 12, 11, 14)},
             {"dataset": "cdc_age",
              "db_date": datetime.datetime(2009, 11, 10, 10, 45)}]
mycollection.insert_many(docs)
```

```
<pymongo.results.InsertManyResult at 0x7f471834ebd8>
```

View Document using PyMongo

To view all documents in a collection

```
cursor = mycollection
for document in cursor.find():
    print document
```

```
{u'_id': ObjectId('5cefd218b25ea819ab3db470'), u'db_date': datetime.datetime(2009, 11, 12, 11, 14), u'dataset': u'cdc_dat
a'}
{u'_id': ObjectId('5cefd21fb25ea819ab3db471'), u'db_date': datetime.datetime(2007, 11, 12, 11, 14), u'dataset': u'cdc_gen
der'}
{u'_id': ObjectId('5cefd21fb25ea819ab3db472'), u'db_date': datetime.datetime(2009, 11, 10, 10, 45), u'dataset': u'cdc_age
'}
```

To view one document

```
from bson.objectid import ObjectId
mycollection.find_one({"_id" : ObjectId("5cefd21fb25ea819ab3db472")}) #Querying By ObjectId
```

```
{u'_id': ObjectId('5cefd21fb25ea819ab3db472'),
 u'dataset': u'cdc_age',
 u'db_date': datetime.datetime(2009, 11, 10, 10, 45)}
```

```
mycollection.find_one() #returns the first match
```

```
{u'_id': ObjectId('5cefd218b25ea819ab3db470'),
 u'dataset': u'cdc_data',
 u'db_date': datetime.datetime(2009, 11, 12, 11, 14)}
```

Document Querying

To list the document in a Collection:

- *db.collection_name.find()*

To find one document in a Collection:

- *db.collection_name.findOne()*

```
> db.student.find()
{ "_id" : ObjectId("5cefa9cceb5966f8b639cbc8"), "name" : "tom" }
{ "_id" : ObjectId("5cefa9d4eb5966f8b639cbc9"), "name" : "raju" }
{ "_id" : ObjectId("5cefa9d4eb5966f8b639cbca"), "name" : "radha" }
{ "_id" : ObjectId("5cefa9d8eb5966f8b639cbcb"), "name" : "sreetha" }
> db.student.findOne()
{ "_id" : ObjectId("5cefa9cceb5966f8b639cbc8"), "name" : "tom" }
```

Document Querying

Where conditions inside find( )

- *Equality*
- *Less than*
- *Less than or Equals*
- *Greater than*
- *Greater than or Equals*
- *Not Equals*

```
> db.student.find()
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 9 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
> db.student.find({"name":"abc"})
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 9 }
> db.student.find({"age":{$lt:10}})
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 9 }
> db.student.find({"age":{$lte:10}})
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 9 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
> db.student.find({"age":{$gt:10}})
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
> db.student.find({"age":{$gte:10}})
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
> db.student.find({"age":{$ne:10}})
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 9 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
>
```

Document Querying

AND and OR Condition in find( )

```
> db.student.find()
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 9 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
{ "_id" : ObjectId("5cefb3c5eb5966f8b639cbd2"), "name" : "def", "age" : 12 }
> db.student.find({"$and":[{"age":10},{"name":"def"}]})
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
> db.student.find({"$or":[{"age":10},{"name":"def"}]})
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
{ "_id" : ObjectId("5cefb3c5eb5966f8b639cbd2"), "name" : "def", "age" : 12 }
> db.student.find({"$or":[{"name":"abc"},{"$and":[{"age":10},{"name":"def"}]}]})

{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 9 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
>
```

Update Document

To replace existing document:

- *db.collection_name.save({_id:ObjectId(),new_data})*

```
> db.student.find()
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 100 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
{ "_id" : ObjectId("5cefb3c5eb5966f8b639cbd2"), "name" : "def", "age" : 12 }
> db.student.save({"_id" : ObjectId("5cefaec6eb5966f8b639cbcd"),"name":"ABCD","a
ge":50})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find()
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "ABCD", "age" : 50 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
{ "_id" : ObjectId("5cefb3c5eb5966f8b639cbd2"), "name" : "def", "age" : 12 }
```

Update Document

To update existing document:

- *db.collection_name.update(selection _criteria, updated_data)*

```
> db.student.find()
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 9 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
{ "_id" : ObjectId("5cefb3c5eb5966f8b639cbd2"), "name" : "def", "age" : 12 }
> db.student.update({"name":"abc"},{$set:{"age":100}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find()
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "abc", "age" : 100 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
{ "_id" : ObjectId("5cefb3c5eb5966f8b639cbd2"), "name" : "def", "age" : 12 }
```

Update Document using PyMongo

To update a document

```
mycollection.update_one({'author': u'cdc_age'},{ '$set':{'dataset': u'cdc_data_age'}})
```

```
<pymongo.results.UpdateResult at 0x7f47183a87a0>
```

```
cursor = mycollection
for document in cursor.find():
    print document
```

```
{u'_id': ObjectId('5cefbeecb25ea819ab3db465'), u'db_date': datetime.datetime(2007, 11, 12, 11, 14), u'dataset': u'cdc_gen
der'}
{u'dataset': u'cdc_data_age', u'_id': ObjectId('5cefbeecb25ea819ab3db466'), u'db_date': datetime.datetime(2009, 11, 10, 1
0, 45), u'author': u'cdc_age'}
```

```
mycollection.update_one({'author': u'cdc_age'},{ '$set':{'author': u'cdc_data_age1'}})
```

```
<pymongo.results.UpdateResult at 0x7f47183a8ab8>
```

Update Document using PyMongo

```
cursor = mycollection
for document in cursor.find():
    print document
```

```
{u'_id': ObjectId('5cefbeecb25ea819ab3db465'), u'db_date': datetime.datetime(2007, 11, 12, 11, 14), u'dataset': u'cdc_gen
der'}
{u'dataset': u'cdc_data_age', u'_id': ObjectId('5cefbeecb25ea819ab3db466'), u'db_date': datetime.datetime(2009, 11, 10, 1
0, 45), u'author': u'cdc_data_age1'}
```

```
mycollection.update_one({'_id': ObjectId('5cefd21fb25ea819ab3db471')}, { '$set':{'dataset': u'cdc_data_gender'}})
```

```
<pymongo.results.UpdateResult at 0x7f4718333ea8>
```

```
cursor = mycollection
for document in cursor.find():
    print document
```

```
{u'_id': ObjectId('5cefd21fb25ea819ab3db471'), u'author': u'cdc_data_gender', u'db_date': datetime.datetime(2007, 11, 12,
11, 14), u'dataset': u'cdc_data_gender'}
{u'_id': ObjectId('5cefd21fb25ea819ab3db472'), u'db_date': datetime.datetime(2009, 11, 10, 10, 45), u'dataset': u'cdc_age
'}
```

Deleting Document

To delete document satisfying a condition:

- *db.collection_name.remove(criteria, optional_count)*

If nothing is passed as parameter [remove({})] all documents will be deleted

Removes only one document satisfying the criteria

```
> db.student.find()
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "ABCD", "age" : 50 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbcf"), "name" : "def", "age" : 10 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
{ "_id" : ObjectId("5cefb3c5eb5966f8b639cbd2"), "name" : "def", "age" : 12 }
> db.student.remove({"name":"def"},1)
WriteResult({ "nRemoved" : 1 })
> db.student.find()
{ "_id" : ObjectId("5cefaec6eb5966f8b639cbcd"), "name" : "ABCD", "age" : 50 }
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
{ "_id" : ObjectId("5cefb3c5eb5966f8b639cbd2"), "name" : "def", "age" : 12 }
> db.student.remove({"age":{$gt:11}})
WriteResult({ "nRemoved" : 2 })
> db.student.find()
{ "_id" : ObjectId("5cefaee2eb5966f8b639cbd0"), "name" : "ghi", "age" : 11 }
```

Deleting Document using PyMongo

To delete a document

```
mycollection.delete_one({"_id" : ObjectId("5cefd218b25ea819ab3db470")})
```

```
<pymongo.results.DeleteResult at 0x7f471834efc8>
```

```
cursor = mycollection
for document in cursor.find():
    print document
```

```
{u'_id': ObjectId('5cefd21fb25ea819ab3db471'), u'db_date': datetime.datetime(2007, 11, 12, 11, 14), u'dataset': u'cdc_gen
der'}
{u'_id': ObjectId('5cefd21fb25ea819ab3db472'), u'db_date': datetime.datetime(2009, 11, 10, 10, 45), u'dataset': u'cdc_age
'}
```

Deleting Document using PyMongo

To Delete multiple documents

```
cursor = mycollection
for document in cursor.find():
    print document
```

```
{u'_id': ObjectId('5cefd21fb25ea819ab3db471'), u'db_date': datetime.datetime(2007, 11, 12, 11, 14), u'dataset': u'cdc_gen
der'}
{u'_id': ObjectId('5cefd21fb25ea819ab3db472'), u'db_date': datetime.datetime(2009, 11, 10, 10, 45), u'dataset': u'cdc_age
'}
{u'_id': ObjectId('5cefd254b25ea819ab3db473'), u'db_date': datetime.datetime(2009, 11, 12, 11, 14), u'dataset': u'cdc_dat
a1'}
{u'_id': ObjectId('5cefd254b25ea819ab3db474'), u'db_date': datetime.datetime(2009, 11, 10, 10, 45), u'author': u'cdc_age1
'}
```

```
mycollection.delete_many({'_id': ObjectId('5cefd254b25ea819ab3db474'),
                          '_id': ObjectId('5cefd254b25ea819ab3db473')
                         })
```

```
<pymongo.results.DeleteResult at 0x7f4718333710>
```

```
cursor = mycollection
for document in cursor.find():
    print document
```

```
{u'_id': ObjectId('5cefd21fb25ea819ab3db471'), u'db_date': datetime.datetime(2007, 11, 12, 11, 14), u'dataset': u'cdc_gen
der'}
{u'_id': ObjectId('5cefd21fb25ea819ab3db472'), u'db_date': datetime.datetime(2009, 11, 10, 10, 45), u'dataset': u'cdc_age
'}
```

Projection

To select specific fields/columns:

- *db.collection_name.find({ }, {key:1})*

To choose a field, set value for its key as 1, else 0

```
>
> db.student.find()
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf6"), "name" : "a" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf7"), "name" : "b" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf8"), "name" : "c" }
> db.student.find({},{_id:0,"name":1})
{ "name" : "a" }
{ "name" : "b" }
{ "name" : "c" }
>
```

Projection using PyMongo

To display only certain fields in mongo

```
for x in mycollection.find({},{ "_id": 0, "dataset": 1}):
    print x
```

```
{u'dataset': u'cdc_data_gender'}
{u'dataset': u'cdc_data_age'}
{u'dataset': u'ims_data'}
```

```
for x in mycollection.find({},{ "_id": 0, "author": 1}):
    print x
```

```
{u'author': u'cdc_data_gender'}
{}
{}
```

Limiting Records

To limit number of records:

- *db.collection_name.find().limit()*

To skip number of records:

- *db.collection_name.find().skip()*

```
> db.student.find()
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf6"), "name" : "a" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf7"), "name" : "b" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf8"), "name" : "c" }
> db.student.find().limit(2)
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf6"), "name" : "a" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf7"), "name" : "b" }
> db.student.find().skip(2)
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf8"), "name" : "c" }
>
```

Limiting Records using PyMongo

To limit the number of results

```
limited_doc_list =mycollection.find().limit(1)
for x in limited_doc_list:
    print x
```

```
{u'_id': ObjectId('5cefd21fb25ea819ab3db471'), u'author': u'cdc_data_gender', u'db_date': datetime.datetime(2007, 11, 12,
11, 14), u'dataset': u'cdc_data_gender'}
```

```
### To delete a field in a document
```

```
mycollection.update_one({'_id': ObjectId('5cefd572b25ea819ab3db477')},{"$unset":{u'dataset': u'cdc_data_age'}})
```

```
<pymongo.results.UpdateResult at 0x7f47183281b8>
```

```
cursor = mycollection
for document in cursor.find():
    print document
```

```
{u'_id': ObjectId('5cefd21fb25ea819ab3db471'), u'author': u'cdc_data_gender', u'db_date': datetime.datetime(2007, 11, 12,
11, 14), u'dataset': u'cdc_data_gender'}
{u'_id': ObjectId('5cefd572b25ea819ab3db477'), u'db_date': datetime.datetime(2007, 11, 12, 11, 14)}
{u'_id': ObjectId('5cefd572b25ea819ab3db478'), u'db_date': datetime.datetime(2009, 11, 10, 10, 45), u'dataset': u'ims_dat
a'}
```

Sorting Records

To sort the records:

- *db.collection_name.find().sort({key:1/-1})*

To sort based on a field, set value for its key as 1 or asc , -1 for desc

```
> db.student.find()
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf6"), "name" : "a" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf7"), "name" : "b" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf8"), "name" : "c" }
> db.student.find().sort({"name":-1})
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf8"), "name" : "c" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf7"), "name" : "b" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf6"), "name" : "a" }
> db.student.find().sort({"name":1})
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf6"), "name" : "a" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf7"), "name" : "b" }
{ "_id" : ObjectId("5cefc540ea9a0601a9037bf8"), "name" : "c" }
```

Sorting Records using PyMongo

To sort documents

```
sorted_docs = mycollection.find().sort('dataset')
for x in sorted_docs:
    print x
```

```
{u'_id': ObjectId('5cefd572b25ea819ab3db477'), u'db_date': datetime.datetime(2007, 11, 12, 11, 14), u'dataset': u'cdc_dat
a_age'}
{u'_id': ObjectId('5cefd21fb25ea819ab3db471'), u'author': u'cdc_data_gender', u'db_date': datetime.datetime(2007, 11, 12,
11, 14), u'dataset': u'cdc_data_gender'}
{u'_id': ObjectId('5cefd572b25ea819ab3db478'), u'db_date': datetime.datetime(2009, 11, 10, 10, 45), u'dataset': u'ims_dat
a'}
```

Advantages

- Schema less
- Structure of a single object is clear
- No complex joins
- Deep query-ability
- Tuning
- Ease of scale-out
- Conversion/mapping of application objects to database objects not needed
- Uses internal memory for storing the working set, enabling faster access of data
- Document (JSON) Oriented Storage
- Index on any attribute
- Replication and high availability
- Rich queries

Thanks!