**PYTHON MONGODB**

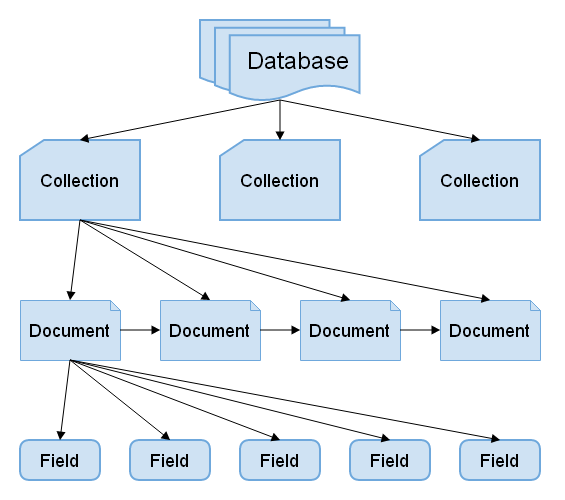
1. **Introduction to MongoDB**

MongoDB is an open-source NoSQL database. It is written in C++. It is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability.

* **Reasons to opt for MongoDB**

1. It supports hierarchical data structure
2. It supports associate arrays like Dictionaries in Python.
3. Python drivers to connect python-application with Database.(Pymongo)
4. It is designed for Big Data.
5. Deployment of MongoDB is very easy.

A single MongoDB server typically has multiple databases. MongoDB works on concept of collection and document.



## Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

## Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

* **Install MongoDB**

MongoDB 3.4.2 .msi

Create folder C:/data/db

Run mongod.exe

Run mongo.exe

MongoChef.msi (GUI for DB)

Test MongoDB Installation

*Eg: db.test.save({Name:’abcd’,age:’30’})*

*db.test.find()*

* **Basic CRUD operations**

use db\_name ->To create a DB

Note:Here db name ‘abcd’ & collection name ‘apple’

1. **Create**

*eg: db.apple.insert({\_id:01,Name:’abc’,age:22,mob:2345})*

*db.apple.insert([{\_id:02,Name:’abc’,age:25,mob:2645},{\_id:03,Name:’cde’}])*

1. **Read**

*eg: db.apple.find()*

*db.apple.find({Name:’zzz’})*

*db.apple.findOne({Name:’zzz’})*

*db.apple.findOne({Name:’zzz’},{\_id:0})*

1. **Update**

*Eg:db.apple.update({Name:’zzz’},{$set:{ Name:’zza’}})*

*db.apple.updateMany({mob:554},{$set:{ Name:’zz’}})*

1. **Delete**

*Eg:db.apple.remove({Name:’zzz’})*

*db.apple.remove({ })*

*Note: db.apple.count() -> Count of documents*

*db.apple.drop() -> delete collection*

*db.dropDatabase() -> delete Database*

**Set Path for python and pip**

My computer-> advanced settings->Env Variable->path edit

C:\python27

C:\python27\scripts

1. **Making a connection with MongoClient**

Pycharm-View-Tool window-Terminal

pip install pymongo

Create a MongoClient to the running **mongod** instance.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

1. **Creating a Database**

To create a database in MongoDB

1. Create a MongoClient object
2. Specify the name of the database using dictionary style

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

sdb= *sample\_client* ["sample\_db"]

**4. Creating a Collection**

To create a collection in MongoDB, use database object and specify the name of the colelction using dictionary style

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

sdb= *sample\_client* ["sample\_db"]

*scol= sdb["students"]*

**5. Inserting a Document**

To insert a document into a collection, insert\_one() method is used.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

sdb= *sample\_client* ["sample\_db"]

*scol= sdb["students"]*

sdoc={name:’abc’,age:22,mob:21234}

*x = scol.insert\_one(sdoc)*

*print(x.inserted\_id)*

1. **Bulk Inserts**

To insert multiple documents into a collection the insert\_many() method is used.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*sdoc=[*

*{name:’abc’,age:22,mob:21234},*

*{name:’dec’,age:25,mob:’2345’},*

*{name:’abc’,age:27,mob:21244},*

*{name:’dec’,age:23,mob:’2245’}*

*]*

*x = scol.insert\_many(sdoc)*

*print(x.inserted\_ids)*

## Insert Multiple Documents, with Specified IDs

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*sdoc=[*

*{\_id:1,name:’abc’,age:22,mob:21234},*

*{\_id:2,name:’dec’,age:25,mob:’2345’},*

*{\_id:3,name:’abc’,age:27,mob:21244},*

*{\_id:4,name:’dec’,age:23,mob:’2245’}*

*]*

*x = scol.insert\_many(sdoc)*

*print(x.inserted\_ids)*

1. **Getting a Single Document with find\_one()**

To select data from a collection in MongoDB, we can use the find\_one() method. It returns the first occurrence in the selection.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find\_one()  
print(x)*

*Eg2: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find\_one ({name=’abc’})*

*for i in x:*

*print(i)*

1. **Querying by object id**

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find\_one ({\_id:1})*

*print(x)*

1. **Querying for more than one Document**

The find() method returns all occurrences in the selection.The first parameter of the find() method is a query object.

*Eg1: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find ()*

*for i in x:*

*print(i)*

*Eg2: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find ({name=’abc’})*

*for i in x:*

*print(i)*

## Return Only Some Fields

The second parameter of the find() method is an object describing which fields to include in the result. This parameter is optional, and if omitted, all fields will be included in the result.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find ({},{ "\_id": 0, "name": 1})*

*for i in x:*

*print(i)*

* **Advanced queries with modifiers**

To find the documents where the "name" field starts with the letter "S" or higher (alphabetically)

*Eg1: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find (name*:{ "$gt": "S" }*)*

*for i in x:*

*print(i)*

To find only the documents where the "name" field starts with the letter "S", use the regular expression {"$regex": "^S"}

*Eg2: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find (name*: { "$regex": "^S" }

*for i in x:*

*print(i)*

1. **Sorting**

Use the sort() method to sort the result in ascending or descending order.

* **Sort ascending**

*Eg1: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find ().sort(“name”)*

*for i in x:*

*print(i)*

## Sort Descending

*Eg2: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find ().sort(“name”,-1)*

*for i in x:*

*print(i)*

1. **Range Queries**

MongoDB supports many different types of advanced queries.

* Limit results based on certain date and also sort the results by name

*import datetime*

*import pprint*

*import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*sdoc={name:’abc’,age:22,mob:21234,date:datetime.datetime.utcnow()}*

*x = scol.insert\_one(sdoc)*

*d=datetime.datetime(2018, 11, 11, 12)*

*for i in sdoc.find({date:{$lt:d}}).sort(“name”):*

*pprint.pprint(i)*

Here we use the special "$lt" operator to do a range query, and also call [sort()](http://api.mongodb.com/python/current/api/pymongo/cursor.html#pymongo.cursor.Cursor.sort) to sort the results by name.

1. **Counting**

A [count()](http://api.mongodb.com/python/current/api/pymongo/cursor.html#pymongo.cursor.Cursor.count) operation shows how many documents match a query.

*Eg1: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find ().count()*

*for i in x:*

*print(i)*

*Eg2: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find ({name=’abc’}).count()*

*for i in x:*

*print(i)*

## Limit the Result

The limit() method takes one parameter, a number defining how many documents to return.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.find ().limit(3)*

*for i in x:*

*print(i)*

1. **Update**

Update a document by using the update\_one() method.The first parameter of the update\_one() method is a query object defining which document to update. If the query finds more than one record, only the first occurrence is updated. The second parameter is an object defining the new values of the document.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*old\_doc = { "name": "abc" }  
new\_doc = { "$set": { "name": "zzz" } }  
scol.update\_one(old\_doc, new\_doc)  
for x in scol.find():  
 print(x)*

To update all documents that meets the criteria of the query, use the update\_many() method.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*old\_doc = { "name": "abc" }  
new\_doc = { "$set": { "age":30 } }  
scol.update\_many(old\_doc, new\_doc)*

*#print(x.modified\_count, "documents updated.")  
for x in scol.find():  
 print(x)*

1. **Delete**

To delete one document, we use the delete\_one() method. The first parameter of the delete\_one() method is a query object defining which document to delete. If the query finds more than one document, only the first occurrence is deleted.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*scol.delete\_one ({name=’abc’})*

## Delete Many Documents

To delete more than one document, use the delete\_many() method. The first parameter of the delete\_many() method is a query object defining which documents to delete.

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*myquery = { "name": {"$regex": "^S"} }  
 x = scol.delete\_many(myquery)  
 print(x.deleted\_count + " documents deleted.")*

## Delete All Documents in a Collection

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

*x = scol.delete\_many({})  
 print(x.deleted\_count + " documents deleted.")*

## Delete Collection

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sdb= sample\_client ["sample\_db"]*

*scol= sdb["students"]*

scol.drop()

The drop() method returns true if the collection was dropped successfully, and false if the collection does not exist.

## Delete Database

*Eg: import pymongo  
sample\_client = pymongo.MongoClient("mongodb://localhost:27017/")*

*sample\_client.drop\_database(‘sample\_db’)*

## A Note On Unicode Strings

MongoDB stores data in BSON format. BSON strings are UTF-8 encoded so PyMongo must ensure that any strings it stores contain only valid UTF-8 data. Regular strings (<type ‘str’>) are validated and stored unaltered. Unicode strings (<type ‘unicode’>) are encoded UTF-8 first. The reason our example string is represented in the Python shell as u’Mike’ instead of ‘Mike’ is that PyMongo decodes each BSON string to a Python unicode string, not a regular str.