About Mongo-DB

An easy to use NoSQL Database -- A document database ( means records in MongoDB is document which is a data structure composed of key/fields and value pairs ) -- designed for ease of development and scaling – MongoDB documents are similar to JSON object.

* **SQL vs MongoDB Components**

|  |  |
| --- | --- |
| **SQL** | **MongoDB** |
| Database | Database |
| Tables | Collection |
| Rows | Documents |

* **Advantages of using MongoDB over SQL**
* Documents are native datatype for many programming languages like python dictionary.
* Horizontal Scalability.
* Data in MongoDB has a flexible schema, so can dynamically modify schema as per the application performance requirement without downtime.
* Because of embedded documents it avoids the concept of expensive joins and so improve the performance.
* **Mongo v/s MongoD(Mongo Daemon)**
* Mongo is a command line shell that connects to MongoD instance for performing any operation.
* MongoD is basically the host process – it acts just like a complier u can say – it helps mongo commands to perform any operations like CRUD.

Pymongo

The standard MongoDB driver library for Python – easy to use and offers an intuitive API for accessing databases, collections, and documents – highly compatible with python as the object retrieves from mongoDB with pymongo are in JSON format which is native to python datatypes like dictionary and lists , so are easy to handle and perform operations.

* **Insert Documents into collection**
* from pymongo import MongoClient
* try:
* conn = MongoClient('localhost', 27017)
* print("Connected successfully!!!")
* except:
* print("Could not connect to MongoDB")
* MySampleDB = conn['SampleDB']
* MySmplTable\_Crsr = MySampleDB['SampleCollection']
* Dict\_single\_record = {"Name":'ronaldo443',"Age":45}
* Dict\_many\_records= [{"Name":'shubham22',"Age":49 },{"Name":'ranumandal043',"Age":35}]
* # To insert single record
* MySmplTable\_Crsr.insert\_one(Dict\_single\_record)
* # To insert multiple records
* MySmplTable\_Crsr.insert\_many(Dict\_many\_records)
* **Find Operation :**
* To find and return only one field

from pymongo import MongoClient

try:

    conn = MongoClient('localhost', 27017)

except:

    print("Could not connect to MongoDB")

MySampleDB = conn['SampleDB']

MySmplTable\_Crsr = MySampleDB['SampleCollection']

# To find and return only one field

res = MySmplTable\_Crsr.find\_one()

print(res)

* To find and return all occurrence fields

res = MySmplTable\_Crsr.find() # -- this provides an cursor object

for r in res:

    print(r)

* To return only selected fields and hide others

# To return only selected fields and hide other

res = MySmplTable\_Crsr.find({},{"\_id":0 ,"Name":1,"Age":1}) # 0 to hide and 1 to display

for r in res:

    print(r)

* **MongoDB Query Object :**

Help to enhance the functionality of find() function – first argument of find() is query object – help filtering the results -- used to limit the search.

* **Simple query**

from pymongo import MongoClient

try:

    conn = MongoClient('localhost', 27017)

except:

    print("Could not connect to MongoDB")

MySampleDB = conn['SampleDB']

MySmplTable\_Crsr = MySampleDB['SampleCollection']

# simple query

qry\_obj = {"Age":49}

res = MySmplTable\_Crsr.find(qry\_obj)

for r in res:

    print(r)

* **Advanced Query (Using modifiers)**

# advanced query with modifiers like : $lt , $lte ,$gt ,$gte

qry\_lessthan = {"Age":{"$lt":46}} #less than ,$gt - greater than

res = MySmplTable\_Crsr.find(qry\_lessthan)

for r in res:

    print(r)

* **Filter using RegEx**

# filtering result with Regular Expression

qry\_RegEX = {"Name":{"$regex":'^r'}} #less than ,$gt - greater than

res = MySmplTable\_Crsr.find(qry\_RegEX)

for r in res:

    print(r)

* **MongoDB sort() :**

sort the result in ascending or descending order -- takes one parameter for "fieldname" and one parameter for "direction" (ascending is the default direction).

* **Ascending order**

from pymongo import MongoClient

try:

    conn = MongoClient('localhost', 27017)

except:

    print("Could not connect to MongoDB")

MySampleDB = conn['SampleDB']

MySmplTable\_Crsr = MySampleDB['SampleCollection']

# To sort in Asc order by field "Age"

res = MySmplTable\_Crsr.find().sort("Age")

for r in res:

    print(r)

* **Descending order**

# To sort in desc order by field "Age"

res = MySmplTable\_Crsr.find().sort("Age",-1)

for r in res:

    print(r)

* **Document(Row/Record) Delete :**

First parameter of the delete\_one() method is a query object defining which document to delete.

from pymongo import MongoClient

try:

    conn = MongoClient('localhost', 27017)

except:

    print("Could not connect to MongoDB")

MySampleDB = conn['SampleDB']

MySmplTable\_Crsr = MySampleDB['SampleCollection']

# To delete single record ...

Delete\_qry = {"Age":35}

res = MySmplTable\_Crsr.delete\_one(Delete\_qry)

# To delete multiple records ...

Delete\_qry = {"Age":{"$lt":49}}

res = MySmplTable\_Crsr.delete\_many(Delete\_qry)

print( "Total documents deleted == ",res.deleted\_count) # To get deleted records count

# To Delete all the records in a collection ...

res = MySmplTable\_Crsr.delete\_many({})

print( "Total documents deleted == ",res.deleted\_count) # To get deleted records count

* **Drop Collection(Table) :**

from pymongo import MongoClient

try:

    conn = MongoClient('localhost', 27017)

except:

    print("Could not connect to MongoDB")

MySampleDB = conn['SampleDB']

MySmplTable\_Crsr = MySampleDB['SampleCollection']

MySmplTable\_Crsr.drop() # droping table(collection) "SampleCollection"

* **Update Collection**

First parameter of the update\_one() and update\_many() method is a query object defining which document to update (Where condition).

**NOTE**: IF field does not exists in the record, then instead of updating it will simply insert the new defined field in the record.

* **MongoDB Limit**

from pymongo import MongoClient

try:

    conn = MongoClient('localhost', 27017)

except:

    print("Could not connect to MongoDB")

MySampleDB = conn['SampleDB']

MySmplTable\_Crsr = MySampleDB['SampleCollection']

# To update on basis of single condition

condition\_qry = {"Age":49}

Update\_qry = {"$set":{ "Address": "Philipines 123" }} # Changed/new value

MySmplTable\_Crsr.update\_many(condition\_qry ,Update\_qry)

# To update on basis of multiple condition

condition\_qry\_mtiple = {"Age":45,"Name":'ronaldo443'}

Update\_qry = {"$set":{ "Address": "Columbia 123" }} # Changed/new value

MySmplTable\_Crsr.update\_many(condition\_qry\_mtiple ,Update\_qry)

limit the result in MongoDB, we use the limit() method -- takes one parameter, a number defining how many documents to return.

from pymongo import MongoClient

try:

    conn = MongoClient('localhost', 27017)

except:

    print("Could not connect to MongoDB")

MySampleDB = conn['SampleDB']

MySmplTable\_Crsr = MySampleDB['SampleCollection']

for res in MySmplTable\_Crsr.find().limit(2):

    print(res)