Which of the following are types available in BSON?

1. Strings
2. Floating-point numbers
3. Arrays
4. Objects(Subdocuments)
5. Timestamps

MongoDB stores data in form of BSON(Binary format)

ObjectID construction:

Current Time+Identifier of the machine that’s constructing the objectID+ProcessID of the process on the computer that’s contructing the object ID+Counter that globals to the process that’s constructing object ID--- Which is unique.

findOne()

Display particular column list:

>db.people.findOne( {“name” : ”Jones”}, {“\_id” : false, “name” : true } );

>{“name” : “Jones”}

Either in find or findOne 2nd argument contains the list of attributes to be returned. And by default “\_id” is also shown, to disable that use “\_id”:false or “\_id”:0.

db.scores.find({score:{$gt:95, $lte:98}, type:”essay”})

db.scores.find({profession:{$exists:true}})

db.scores.find({name:{$type:2}}) (2 means String binary representaion)

db.scores.find({name:{$regex:”a”}})—Returns name contains “a”

db.scores.find({name:{$regex:”e$”}})--- Returns name ends with “e”

db.scores.find({name:{$regex:”^A”}})--- Returns name Starts with “A”

Write a query that retreives documents from a *users* collection where the *username* has a ‘q’ in it, and the document has an *email* field.

Cursors:

When called db.people.find(), a requester being constructed and returned in the shell.

The shell is configured to printout the cursors by iterating all of the elements that all retreive from the cursor and printing out elements.

You can hold on to the cursor or to any other value like in programming language

>cur=db.people.find(); null;

Here cur holding onto Cursor Object and cursor object has messages.

>cur.hasNext() returns true, if any other document has to visit.

>cur.next() prints next document to be visited on this cursor.

You can iterate the cursor:

>while(cur.hasNext()) printjson(cur.next());

Always establish a new cursor if you want to do any iterations of opertaions.

>cur = db.people.find(); null;

**Limit:**

>cur.limit(5); null;

Limits to only 5 documents from the cursor.

* while(cur.hasNext()) printjson(cur.next());

Now it prints only 5 documents

**Sort:**

Sort method instructs the cursor to print the documents in the order specified by the argument to the sort.

>cur.sort({name:-1});null; -- -1 Descending order(Reverse order)

null

* > while(cur.hasNext()) printjson(cur.next());

>cur.sort({name:-1}).limit(3); null;

Its important to understand that the sort and limit method modified the information that transmitted over to the database. I can apply these methods to a cursor after having retreiving the documents from DB or even checking whether cursor having hasNext().

This is because, sort and limit need to be processed inside the DB engine, donot processed ordering the documents in the memory in the client, they are processed by server side ordering and limiting.

**Skip:**

>cur.sort({name:-1}).limit(3).skip(2);null;

It will return in the reverse order and returns only 3 documents and skips the first 2 and returns the next 3 documents

Quiz: Querying, Cursors

When can you change the behavior of a cursor, by applying a sort, skip, or limit to it?

* This can be done at any point, and will reset the cursor
* This can be done at any point, and will apply to any documents the cursor hasn't yet pulled.
* This can be done at any point before the first document is called and before you've checked to see if it is empty.
* This can be done only before the cursor is created.
* This can only be done when the cursor is created.

Answer is **C**

**Counting Results:**

>db.scores.count({type:”exam”})

Returns the count.

**Quiz**: Counting Results

How would you count the documents in the *scores* collection where the type was "essay" and the score was greater than 90?

Answer: db.scores.count({type:"essay", score:{$gt:90}});

**4 Types of Updates:**

4th type: It can update multiple documents.

Db.people.update({},{title:”Dr”},{multi:true})

Multi: true ----- It does the mutliple updates. If you don’t specify multi:true, it will update to the first document which it finds.

Here where clause is empty, hence updates for all the documents.

You can specify where clause here.

**Quiz: Multi-update**

Recall the schema of the *scores* collection:

{

"\_id" : ObjectId("50844162cb4cf456b4694f8"),

"student" : 0,

"type" : "exam",

"score" : 75

}

How would you give every record whose score was less than 70 an extra 20 points?

**Query:**

db.scores.update({"score":{"$lt":70}}, {"$inc":{"score":20}}, {multi:true})

**Removing Data:**

You have essentially 3 options.

1. Each collection has a method called remove. And remove works very much like in that it takes argument which specifies what documents to remove.That argument has exactly the same semantics as the first argument to find, except that you must pass it a document.

If you pass empty document to remove then all the documents in the collection will be removed one by one.

If I specify and argument, for example,

db.people.remove({name:”Alice”})

db.people.remove({name:{$gt:”M”}})

db.people.remove({}) – Removes all documents one by one.

If you want to remove all the documents in a single pass in a somewhat more efficient manner, even if collection is very large

db.people.drop()

**Difference between removing all the documents in the collection and dropping a collection is :**

Removing all the docs from collections requires one-by-one update of internal state for each document that happens to exist in the collection.

Where as dropping a collection requires freeing up some much larger data structures inside of the database’s data files which, although still proportional to the sizeof the collection in terms of the number of the documents, is going to have constant factors that will make dropping the collection vastly faster than removing documents from the collection one by one until the collection is empty.

However, collections can have some metadata associated with them, for instance, indexes, which get discarded when the collection is dropped, but which remain in existence when a collection has all of its document removed.

**The MongoDB Java Driver:**