Mongo CRUD

Ramesh S

Create

use school

db.class5.insert({“name” : ”Hari” , ”age” : 10 , ”likes” : ”Painting”})

db.class5.insert({“name” : ”Swathi” , ”age” : 11 , ”likes” : ”Karate”})

db.class5.insert({“name” : ”Isha” , ”age” : 10 , ”likes” : ”Reading”})

Create - Bulk Insert

db.class5.insert([{ “name” : ”John” , ”age” : 9, ”likes” : ”Cricket” } , { “name” : “Lekha” , ”age” : 11, ”likes” : ”Cricket” } , {“name” : ”Lahari” , ”age” : 8, ”likes” : ”Singing”}])

Create - Working with Arrays

use school

db.class5.insert({ “name” : ”Jane” , ”age” : 10 , ”likes” : [”Painting” , ”Singing”] })

db.class5.insert({ “name” : ”Hari” , ”age” : 11 , ”likes” : [”Karate” , ”Cricket”] })

db.class5.insert({ “name” : ”Ravi” , ”age” : 10, ”likes” : [”Reading” , ”Cricket”] })

Finding

db.class5.find()

db.class5.find({})

db.class5.findOne()

db.class5.find().pretty()

find() returns a cursor

findOne() returns a document/object

pretty() formats the output for better human reading

Finding

db.class5.find({“name”:”Ravi”})

Finds all students with name “Ravi”

db.class5.find({“age”:10})

Finds all students with age 10

Finding

db.class5.find({“name”:”Hari”,”age”:10})

Finds all students with name “Hari” and age 10

db.class5.find({“age”:8,”likes”:”Singing”})

Finds all students with age 8 and who like Singing

Finding - working with comparison operators

db.class5.find({"age":{$gt:10}})

Finds all students with age greater than 10

You may want to try out $lt , $gte , $lte , $eq , $ne

Finding - searching arrays

db.class5.find({“likes”:”Cricket”})

Finds all students who like “Cricket”

Did you notice that find() is polymorphic?

Finding - searching arrays based on position

db.class5.find({“likes.0”:”Cricket”})

Finds all students who like “Cricket” most

Finding

```
db.class5.find({

    $or:[{“likes”:”Cricket”},{“likes”:”Reading”}]

})
```

Finds all students who like “Cricket” or “Reading”

Finding

```
db.class5.find({

    “likes”:{$in:[”Cricket”,”Reading”]}

})
```

Finds all students who like “Cricket” or “Reading”

Finding - based on a field

use school

db.class5.insert({“name” : ”Ashok” , ”age” : 10})

db.class5.insert({“name” : ”Rahim” , ”likes” : ”Painting”})

db.class5.find({age:{$exists:true}})

db.class5.find({likes:{$exists:false}})

Finding - based on a field

db.employees.find({spouse:{$exists:true}})

db.employees.find({retired:{$exists:false}})

Finding - projection

db.class5.find({ age: 10 }, { name: 1, age: 1 })

db.class5.find({ age: 10 }, { likes:0 })

db.class5.find({ age: 10 }, { age: 0 })

db.class5.find({ age: 10 }, { _id: 0 })

db.class5.find({ age: 10 }, { _id: 0,name:1 })

db.class5.find({ age: 10 }, { name:1,age:0 })

Finding - distinct

db.class5.distinct(“age”)

db.class5.distinct(“likes”)

Update

db.xyz.insert({"name":"Ramesh"})

db.xyz.update({"name":"Ramesh"},{"job":"Trainer"})

db.xyz.update({"name":"Ramesh"},{$set:{"job":"Trainer"}})

db.xyz.update({"name":"Ramesh"},{$set:{"job":"Trainer"}},{"multi":true})

db.xyz.update({"name":"Ramesh"},{$set:{"job":"Trainer"}},{"multi":true,"upsert":true})

Update - working with arrays

```
db.class5.update(
   { “age”: 10 },
   { $push: { marks: 89 } },
   { multi:1 }
)
```

This will add 89 to marks array

Update - working with arrays

```
db.class5.update(
   { age: 10 },
   { $addToSet: { “likes”: “Swimming” } },
   {“multi”:true}
)
```

This will add “Swimming” to likes array if it is not already there

Advanced Queries

```
db.abcd.insert({'name':'Ramesh',

        experience:{Intergraph:'IT Head',Genpact:'AVP'}

})


db.abcd.find().pretty()
```

Advanced Queries

db.abcd.update({name:'Ramesh'},

{$set:{'experience.HCL':'AGM'}})

db.abcd.find().pretty()

Advanced Queries

db.abcd.update({name:'Ramesh'},

{$set:{'skills':['MongoDB','AngularJS']}})

db.abcd.find().pretty()

Advanced Queries

db.abcd.update({name:'Ramesh'},

{$push:{'skills':'Python'}})

db.abcd.find().pretty()

Advanced Queries

db.abcd.update({name:'Ramesh'},

{$pull:{skills:'AngularJS'}})

db.abcd.find().pretty()

Advanced Queries

db.abcd.update({name:'Ramesh'},

{$pop:{skills:1}})

db.abcd.find().pretty()

Delete documents

db.xyz.remove({name:'ramesh'})

Caution!

db.xyz.remove({}) removes all documents in a collection

Delete fields

db.persons.update({ },{$unset: {"city": ""}},{“multi”:1})

Rename fields

db.persons.update({ },{$rename: {“city”: “village”}},{“multi”:1})

MongoDB Supported BSON Data Types

Type	Number	Alias	Notes
Double	1	“double”	
String	2	“string”	
Object	3	“object”	
Array	4	“array”	
Binary data	5	“binData”	
Undefined	6	“undefined”	Deprecated.
ObjectId	7	“objectId”	

MongoDB Supported BSON Data Types

Type	Number	Alias	Notes
Boolean	8	“bool”	
Date	9	“date”	
Null	10	“null”	
Regular Expression	11	“regex”	
DBPointer	12	“dbPointer”	
JavaScript	13	“javascript”	
Symbol	14	“symbol”	

Query by field type

```
db.class5.find({"name":{$type:"number"}})
```

```
db.class5.find( { $where : "Array.isArray(this.likes)" } )
```

Regex

db.training.find({name:{$regex:/^h/i}})

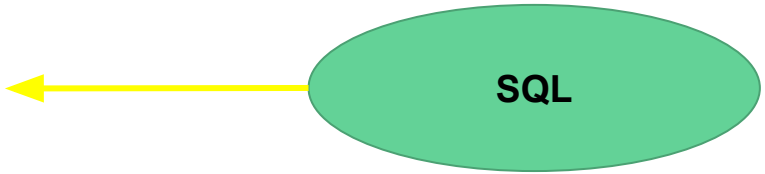
This will find all documents with names

starting with 'h' case insensitive.

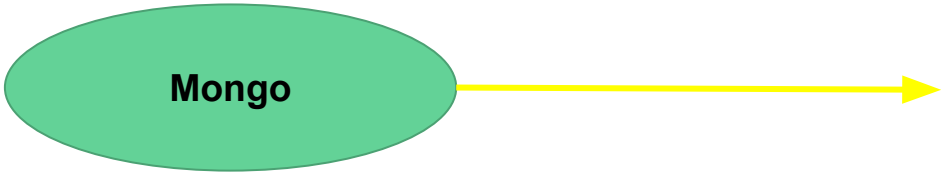
```
db.xyz.find({"name": /me/ }) # contains “me”

db.xyz.find({"name": /^R/ })  #starts with “R”

db.xyz.find({"name": /^R/i })  # starts with “R” ignore case

db.xyz.find({"name": /h$/ })  #ends with “h”
```

SQL vs Mongo Query

```
UPDATE users
SET age = age + 3
WHERE status = "A"
```




```
db.users.update(
{ status: "A" } ,
{ $inc: { age: 3 } },
{ multi: true })
```



Working with Binary Data

db.coll.insert({'abc': new BinData(0,"AQAAAAEBAAVlbl9VSwAAAAA") })

db.coll.find({abc: new BinData(0,"AQAAAAEBAAVlbl9VSwAAAA") })

Sorting

use school

db.class5.find({}).sort({“age”:1}) # sort by age asc

db.class5.find({}).sort({“age”:-1}) # sort by age desc

db.class5.find({}).sort({“age”:1,”name”:1}) # sort by age and name

Pagination

db.class5.find().sort({“age”:1}).skip(2)

db.class5.find().sort({“age”:1}).skip(2).limit(2)