MongoDB Course Notes

Zvi Mints

———————————————————————————————

Insert

```
db.<collection>insert (
          [
          { firstName: “Zvi”, lastName: “Mints”, empId: 1 },
          { firstName: “Zvi”, lastName: “Mints”, empId: 1 }
          ]
                                              )
```

Because we insert multiple documents, instead of **WriteConcern** object being returned,
we receive a **BulkWriteResult** object

———————————————————————————————————

forEach

db.<collection>.find().forEach(**JSFUNCTION**)

For Example:

JSFUNCTION = function(myDoc) { print(“user: “ + myDoc.lastName); }

———————————————————————————————————

Find

And

```
db.<collection>.find(
          {  $and : [   { Dept: “Eng” },
                                  { Title : “Mgr” }
                           ]
       }
)
```

Or

```
db.<collection>.find(
{
$or : [   { Dept: “Eng” },
          { Title : “Mgr” }
           ]
 }
)
```

$gt Example

```
db.<collection>.find({ age : { $gt : 30 } })
```

And & Or

```
db.<collection>.find(
{
 $and : [
{ $or:  [{ Dept: “Eng” },{ Title : “Mgr” }]  },
{ $or:  [{ Dept: “Ru” },{ qty : { $lt : “20”}  }]  }
           ]
 }
```

)

Find First

```
db.<collection>findOne()
```

Find Last

```
db.<collection>.find().sort({ _id: -1}).limit(1).pretty()
```

Find Last That Match Something

```
db.<collection>.find({status: “matched”}).sort({_id: -1}).limit(1).pretty()
```

From MySQL:

- SELECT firstname,lastname
FROM <collection>
WHERE DOE > 2018
ORDER BY lastname

```
db.<collection>.find(
                        {DOE: {$gt : {2018} } },    // Query
                        { _id:0, firstname: 1, lastname: 1} // Projection
                        ).sort({lastname:1}
```

Conclusion

db.<collection>.find({1} , {2})

s.t 1 is **QUERY** and 2 is **PROJECTION**

- 2 can be empty and get **db.<c>.find({ dept: “Eng” })** or
  db.<c>.find({ dept: “Eng”, title : “Manager” })

if we want to use PROJECTION **db.<c>.find({}, {_id:0 })**

Check Indexes

```
db.getCollectionNames().forEach(
function(collection) {
        indexes = db[collection].getIndexKeys();
        print("Indexes for " + collection + ":");
        printjson(indexes)
});
```

Index

```
db.<collection>.createIndex({ ColumnName :  1 }, {background: true, unique: true} ) // 1 Means Sorted
db.<collection>.dropIndex( { ColumnName : 1 })
```

For simple find query like db.<collection>.find({ x : 9992 }).explain() we can get :

”stage" : "COLLSCAN"

Collection Scan -> All over the collection one by one "stage" : "FETCH",

FETCH Means We find in Index but we need to go to the **hardware** to find the _id

"stage" : "IXSCAN" // With **Index Scan**

But for:

db.testCollection.find({x : 19}, {x:1,_id:0}).explain()

"stage" : "PROJECTION_DEFAULT",

stage" : "IXSCAN",

With **NO** Fetch! Which is the **BEST CASE SCENARIO**