MongoDB Shell Tips and Tricks

Norberto Leite

Senior Solutions Architect

mongoDB

What Is the Shell?

Embedded Javascript Interpreter	• vars • functions • data structs + types
Global Functions and Objects	• ObjectId("...") • new Date() • Object.bsonsize()
MongoDB driver Exposed	• db["collection"].find/count/update • short-hand for collections
JSON-like stuff	• Doesn't require quoted keys • Don’t copy and paste too much

MongoDB Shell: Advantages

- Debugging Queries / Syntax
- Testing
- Administration
- Scripting Glue

MongoDB Shell: Disadvantages

- Numbers in JS are a pain
  - 32/64-bit int/long → NumberInt() / NumberLong()
  - Primitive numbers are all 64-bit FP doubles
- Dates can be confusing
  - new Date("1/1/1")
  - new ISODate(…)

NOT: Date("1/1/1") → string

Speed Considerations

- Shell
  - JavaScript is slow
  - Always in "write-acknowledged" (safe mode) / GLE
  - Data Conversions
- Server
  - Applies on the server as well
  - Careful with round-tripping numbers

Easy to Use

- Tab completion on most objects
- Built-in help on most objects (.help())
- show
  - profile # 5 most recent ops of 1ms or more
  - users # List all the users of the current db
  - dbs # List all the databases
  - logs # List all available logs
- Most examples use the shell

Insert, Update, Remove

```
for ( i = 0; i < 1000; i++ ) {
    db.test.insert( {
        x: i,
        ts: new Date()
    } );
}
```

Loading Scripts

- Commandline
  - --eval switch
  - .js files
- Within the shell
  - load()

Examining Records

- Use db.collection.find() to run a query, or .findOne() to fetch a single object
- Results are printed 20 at a time, type “it” to show the next batch
- Set DBQuery.shellBatchSize to change the size of each batch
- Add .pretty() to the function to “pretty-print” the json output

```
> db.test.findOne()
{ "_id": ObjectId("51dc2e743f89441af9a0b0ff"), "x": 1 }
> db.test.find()
{ "_id": ObjectId("51dc2eba3f89441af9a0b180"), "x": 28 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b181"), "x": 29 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b182"), "x": 30 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b183"), "x": 31 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b184"), "x": 32 }
...
Type "it" for more
> it
{ "_id": ObjectId("51dc2eba3f89441af9a0b185"), "x": 33 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b186"), "x": 34 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b187"), "x": 35 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b188"), "x": 36 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b189"), "x": 37 }
...
```

Testing Queries

Use .explain( )

Testing Queries

```
> db.test.find().explain()
{
        "cursor" : "BasicCursor",
        "isMultiKey" : false,
        "n" : 100,
        "nscannedObjects" : 100,
        "nscanned" : 100,
        "nscannedObjectsAllPlans" : 100,
        "nscannedAllPlans" : 100,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 0,
        "indexBounds" : {},
        "server" : "mikes-MacBook-Pro.local:27017"
}
```

Running Commands

- db.runCommand({ ... })
  - Runs any arbitrary command against the current DB
- db.adminCommand({ ... })
  - Run commands against the admin database

db.adminCommand Definition

```
function (obj) {
    if (this._name == "admin") {
        return this.runCommand(obj);
    }
    return this.getSiblingDB("admin")
            .runCommand(obj);
}
```

You can administer replica sets

Directly from the shell

Administering Replica Sets

```
> rs.status() // Check current health of replica set

> rs.add("chicken7:27017") // Add a new member

> rs.remove("chicken7:27017") // Remove a member

> rs.config()
> rs.reconfig()   //Show or modify set configuration

> rs.stepDown()   // Force primary to become secondary

> rs.help()    // Show all available commands
```

Profiling

- setProfilingLevel(lvl, <ms>)
  - 0: none
  - 1: time-based
  - 2: all
- getProfilingLevel()
- Reading from the profile collection
  - db.system.profile.find()

Cool Functions

- printjson → tojson
- forEach on arrays, queries, and cursors

You Didn’t Hear it From Me

```
db.system.js.save({ _id:"echoFunction",
        value : function(x) { return x; }
})
db.loadServerScripts();
echoFunction(3);
```

You Didn’t Hear it From Me

```
[{x:1},{y:1}].forEach(function(x) {
    printjson(x)
})
```

```
{ "x" : 1 }
{ "y" : 1 }
```

Cool Functions

- printjson → tojson
- forEach on arrays, queries, and cursors
- Object.bsonsize
- load(file)
- run(file)

Print All Indexes

```
db.getCollectionNames().
forEach( function(x) {
    print( "Collection: " + x );
    printjson( db[x].getIndexes() );
})
```

Getting the Biggest Doc

```
var cursor  = db.coll.find();
var biggest = 0;
var doc     = {};

cursor.forEach(function (x) {
      var size = Object.bsonsize(x);
      if (size > biggest) {
     biggest = size;
     doc = x;
  }
});
```

Administration Functions

- Sharding
  - sh.status()
  - sh.enableSharding(dbname)
- Replicaset
  - rs.status()
  - cfg = rs.config(); rs.reconfig(cfg);
  - db.isMaster()
- Status
  - Object.bsonsize(document)
  - db.collectionName.stats()

.mongorc.js

- Automagically loaded on startup
  - Unless you specify --norc
- Script your command prompt
  - prompt=function() { return "Hello World"; }
- Colorize output
- Move complex aggregate() queries into functions
- Can be a directory

Emacs-like shell bindings

ctrl A	Move cursor to start of line
ctrl E	Move cursor to end of line
meta B	Move cursor left by one word
meta F	Move cursor right by one word
ctrl L	Clear screen and redisplay line
ctrl R	Reverse history search
meta <	Start of history
meta >	End of history

Want to know more?

The shell is self documented, in JavaScript \o/

…Except the native helper

- help
  - dbs # Shows all commands you can run on a database
  - connect # Connect to other nodes
- Call a function, without the brackets
  - Will show you the actual code behind it
- try.mongodb.org !

Questions?

mongoDB


Thank You
mongoDB