MongoDB

NoSQL for SQL addicts

Vitaly Kushner
astrails.com

NoSQL

- column databases
- key value stores
- document databases

Documents

Data Types

- string
- integer
- float
- boolean
- symbol
- object id
- date
- regular expression
- code
- null
- array
- document

```
{
    name: "Vitaly Kushner",
    company: "Astrails",
    presentation: {
        title: "MongoDB",
        subtitle: "NoSQL for SQL addicts"
    }
}
```

Databases & Collections

Shell

```
➜ mongo
MongoDB shell version: 1.6.5
connecting to: test
> help()
    db.help()                    help on db methods
    db.mycoll.help()             help on collection methods
    rs.help()                    help on replica set methods
    help connect                 connecting to a db help
    help admin                   administrative help
    help misc                    misc things to know
    ...
```

INSERT INTO table (column) VALUES (“value”)

```
db.a.insert({column:“value”})
```

```
> show dbs
admin
local
test
> use intro
switched to db intro
> db.a.insert({a: "test"})
> show dbs
admin
intro
local
test
> show collections
a
system.indexes
> db.a.find()
{ "_id" : ObjectId("4d2eaab9d126330e1e0f9899"), "a" : "test" }
```

```
{ "_id" : ObjectId("4d2eaab9d126330e1e0f9899"), "a" : "test" }
```

```
> db.a.insert({_id: 1, just: "test"})
> db.a.find()
{ "_id" : ObjectId("4d2eaab9d126330e1e0f9899"), "a" : "test" }
{ "_id" : 1, "just" : "test" }
```

Object ID

- Timestamp (4 bytes)
- machine (3 bytes)
- pid (2 bytes)
- increment (3 bytes)

```
DELETE FROM table WHERE column = value;
```

```
> db.a.remove({_id: 1})
```

```
UPDATE table SET column1 = value1 WHERE column2 = value2;
```

```
db.a.update({a: "test"}, {foo: "bar"})
```

```
> db.a.find()
{ "_id" : ObjectId("4d2eadffd126330e1e0f989a"), "a" : "test" }
> db.a.update({a: "test"}, {foo: "bar"})
> db.a.find()
{ "_id" : ObjectId("4d2eadffd126330e1e0f989a"), "foo" : "bar" }
```

Modifiers

$set

```
> db.a.update({"foo": "bar"}, {$set : {bar: "baz"}})
> db.a.find()
{ "_id" : ObjectId("4d2eadffd126330e1e0f989a"), "bar" : "baz",
"foo" : "bar" }
```

$inc

```
> db.a.update({"foo": "bar"}, {$inc : {qwe: 1}})
> db.a.find()
{ "_id" : ObjectId("4d2eadffd126330e1e0f989a"), "bar" : "baz",
"foo" : "bar", "qwe" : 1 }
```

```
> db.a.update({"foo": "bar"}, {$inc : {qwe: 1}})
> db.a.find()
{ "_id" : ObjectId("4d2eadffd126330e1e0f989a"), "bar" : "baz",
"foo" : "bar", "qwe" : 2 }
```

$push

```
> db.b.insert({a: "test"})
> db.b.update({a: "test"}, {$push: {foo: "bar"}})
> db.b.find()
{ "_id" : ObjectId("4d2eb5d0d126330e1e0f989b"), "a" : "test",
"foo" : [ "bar" ] }
> db.b.update({a: "test"}, {$push: {foo: "baz"}})
> db.b.find()
{ "_id" : ObjectId("4d2eb5d0d126330e1e0f989b"), "a" : "test",
"foo" : [ "bar", "baz" ] }
```

```
> db.posts.insert({"title": "mongodb"})
> db.posts.update({"title": "mongodb"}, {$push: {comments:
{user: "Vitaly", comment: "MongoDB rocks!"}}})
> db.posts.update({"title": "mongodb"}, {$push: {comments:
{user: "Boris", comment: "Yes, it does!"}}})
> db.posts.find()
{ "_id" : ObjectId("4d2eb7bfd126330e1e0f989d"),
"comments" : [
        {
                "user" : "Vitaly",
                "comment" : "MongoDB rocks!"
        },
        {
                "user" : "Boris",
                "comment" : "Yes, it does!"
        }
], "title" : "mongodb" }
```

```
> db.posts.insert({_id: 1, title: "test",
  stats: {pageviews: 0, clicks: 0}})
> db.posts.update({_id: 1}, {$inc: {"stats.clicks": 3}})
> db.posts.find()
{ "_id" : 1, "title" : "test", "stats" : { "pageviews" : 0, "clicks" : 3 } }
```

$addToSet

```
> db.c.insert({a: "test"})
> db.c.update({a: "test"}, {$addToSet: {foo: "bar"}})
> db.c.update({a: "test"}, {$addToSet: {foo: "baz"}})
> db.c.update({a: "test"}, {$addToSet: {foo: "bar"}})
> db.c.find()
{ "_id" : ObjectId("4d2eb9e4d126330e1e0f989e"), "a" : "test",
"foo" : [ "bar", "baz" ] }
```

$pop

```
> db.c.find()
{ "_id" : ObjectId("4d2eb9e4d126330e1e0f989e"), "a" : "test",
"foo" : [ "bar", "baz" ] }
> db.c.update({a: "test"}, {$pop: {foo: 1}})
> db.c.find()
{ "_id" : ObjectId("4d2eb9e4d126330e1e0f989e"), "a" : "test",
"foo" : [ "bar" ] }
```

```
> db.posts.find()
{ "_id" : ObjectId("4d2ebf47d126330e1e0f98a0"),
"comments" : [
        {
                "user" : "Vitaly",
                "comment" : "MongoDB rocks!"
        },
        {
                "user" : "Boris",
                "comment" : "Yes, it does!"
        }
], "title" : "mongodb" }
```

```
db.posts.update({title: "mongodb"},
{$inc: {"comments.1.rating": 1}})
```

```
> db.posts.find()
{ "_id" : ObjectId("4d2ebf47d126330e1e0f98a0"),
"comments" : [
        {
                "user" : "Vitaly",
                "comment" : "MongoDB rocks!"
        },
        {
                "user" : "Boris",
                "rating" : 1,
                "comment" : "Yes, it does!"
        }
], "title" : "mongodb" }
```

upsert

```
> db.d.update({foo: 5}, {$inc: {foo: 1}, $set: {bar: "baz"}}, true)
> db.d.find()
{ "_id" : ObjectId("4d2ec0f6ac41433c1f1c9c31"), "bar" : "baz",
"foo" : 6 }
```

multi update

```
> db.e.find()
{ "_id" : 1, "a" : 2, "b" : 3 }
{ "_id" : 2, "a" : 2, "b" : 4 }
> db.e.update({a:2}, {$set: {b: 10}})
> db.e.find()
{ "_id" : 1, "a" : 2, "b" : 10 }
{ "_id" : 2, "a" : 2, "b" : 4 }
> db.e.update({a:2}, {$set: {b: 20}}, false, true)
> db.e.find()
{ "_id" : 1, "a" : 2, "b" : 20 }
{ "_id" : 2, "a" : 2, "b" : 20 }
```

Queries

partial documents

```
> db.posts.find()
{ "_id" : ObjectId("4d2ebf47d126330e1e0f98a0"),
"comments" : [
        {
                "user" : "Vitaly",
                "comment" : "MongoDB rocks!"
        },
        {
                "comment" : "Yes, it does!",
                "rating" : 1,
                "user" : "Boris"
        }
], "title" : "mongodb" }
```

partial documents

```
> db.posts.find({}, {title: 1})
{ "_id" : ObjectId("4d2ebf47d126330e1e0f98a0"), "title" :
"mongodb" }
```

partial documents

```
> db.posts.find({}, {title: 0})
{ "_id" : ObjectId("4d2ebf47d126330e1e0f98a0"),
"comments" : [
        {
                "user" : "Vitaly",
                "comment" : "MongoDB rocks!"
        },
        {
                "comment" : "Yes, it does!",
                "rating" : 1,
                "user" : "Boris"
        }
] }
```

partial documents

```
> db.posts.find({}, {"comments.rating": 1})
{ "_id" : ObjectId("4d2ebf47d126330e1e0f98a0"),
"comments" : [ { }, { "rating" : 1 } ] }
```

conditionals

- $lt - less then
- $gt - greater then
- $lte - less then or equal
- $gte - greater then or equal

conditionals

```
db.posts.find({"stats.clicks": {$gte: 0, $lte: 10}})
```

Indexes

```
db.posts.ensureIndex({"date" : -1, "title": 1})
db.posts.ensureIndex({"comments.date" : -1})
db.users.ensureIndex({"email": 1}, {"unique": true})
```

Geospatial

```
db.users.ensureIndex({"location" : "2d"})
db.users.find({location: {$near: [10, 20]}})
db.users.find({location: {$within: {$box: [[10, 20], [30,40]]}}})
db.users.find({location: {$within: {$center: [[10, 20], 35]}}})
```

There is more

- counts
- limits
- sorting
- offsets
- grouping

Map Reduce

Replication

Master Slave

```
# on the master
db1> mongod --master

# on the slave
db2> mongod --slave --source db1:10000
```

Replica Set

```
# on the primary
db1> mongod --replSet myset/db2

# on the secondary
db2> mongod --replSet myset/db1

# in a mongo shell:
> db.runCommand({"replSetInitiate", {_id: “myset”, ...
```

Replication is Asynchronous!

Sharding

Sharding

```
# start config server
db1> .mongod

# start mongoS
db2> mongos --configdb db1

# start a shard
db3> ./mongod

# in a mongo shell connected to mongoS:
> db.runCommand({addshard : "db3"})
> db.runCommand({enablesharding : "intro"})
> db.runCommand({shardcollection : "intro.a", "key" : {_id: 1}})
```

MongoDB

Q & A

Vitaly Kushner
astrails.com