MongoDB

02

Presented by
Subhash EP

Subhash

mongoDB

{ name: mongo, type: DB }

Subhash

Collection Naming

- A collection is identified by its name. Collection names can be any UTF-8 string, with a few restrictions:
- The empty string ("") is not a valid collection name.
- Collection names may not contain the character \0 (the null character) because this delineates the end of a collection name.

Collection Naming

- You should not create any collections that start with system., a prefix reserved for internal collections.
- For example, the system.users collection contains the database's users, and the system.namespaces collection contains information about all of the database's collections.

Collection Naming

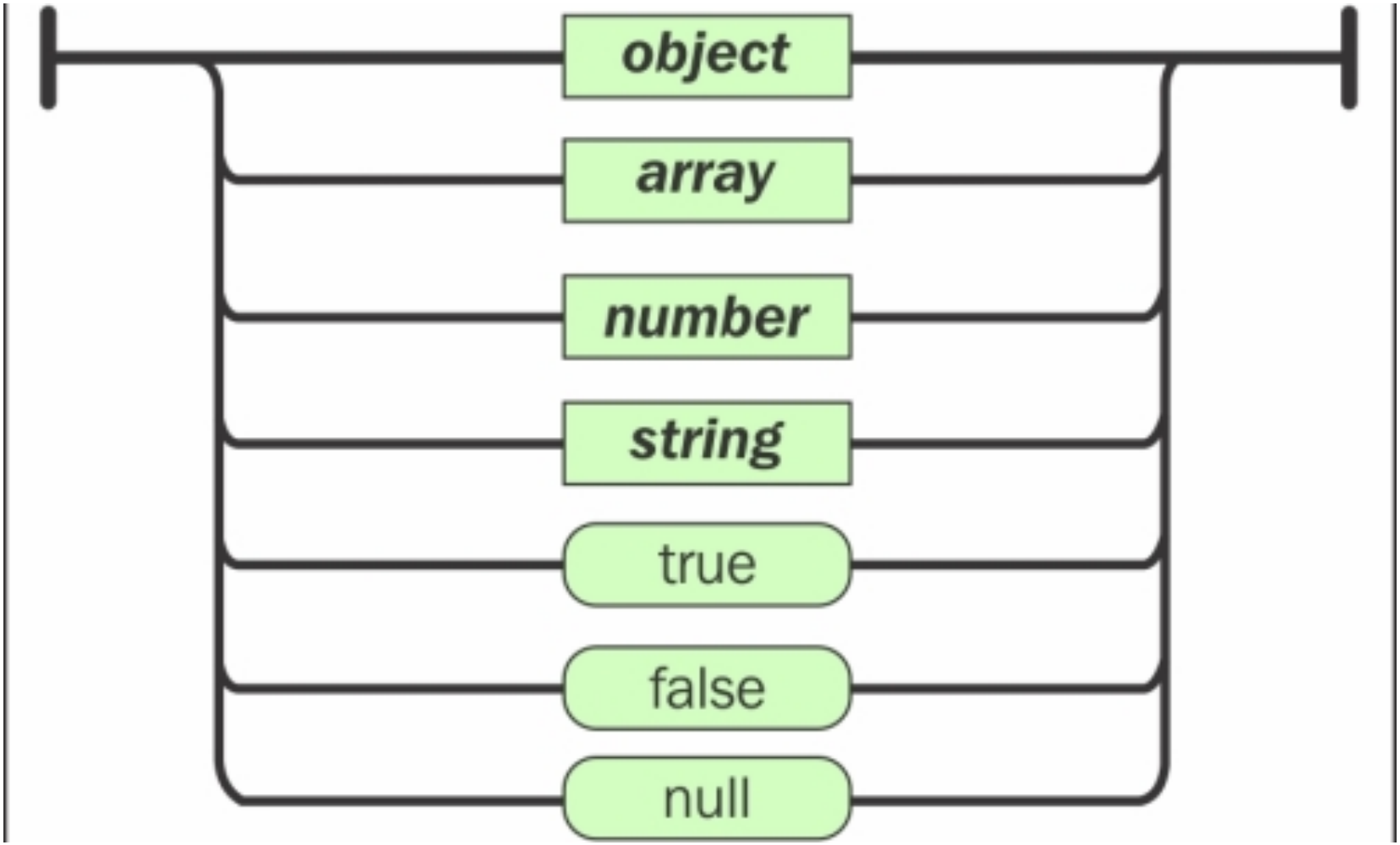
- User-created collections should not contain the reserved character $ in the name.
- The various drivers available for the database do support using $ in collection names because some system-generated collections contain it.
- You should not use $ in a name unless you are accessing one of these collections.

A sample document


```
{
  "_id": 123456,
  "firstName": "John",
  "lastName": "Clay",
  "age": 25,
  "address": {
    "streetAddress": "131 GEN. Almério de Moura Street",
    "city": "Rio de Janeiro",
    "state": "RJ",
    "postalCode": "20921060"
  },
  "phoneNumber":[
    {
      "type": "home",
      "number": "+5521 2222-3333"
    },
    {
      "type": "mobile",
      "number": "+5521 9888-7777"
    }
  ]
}
```


Subhash

JSON Value Structure



Subhash

BSON

- The documents are serialized on disk in a format known as Binary JSON (BSON),
- a binary representation of a JSON document
- maximum length for a BSON document is 16 MB.

Field Name

- Field's name in a document is a string
- _id field is reserved for a primary key
- cannot start name using the character $
- name cannot have a null character, or (.)

document primary key

- _id field is reserved for the primary key.
- By default, this field must be the first one in the document,
- Also, by definition, it is in this field that a unique index will be created.

MongoDB Shell = mongo

- a JavaScript shell
- useful for performing administrative functions,
- inspecting a running instance,
- or just playing around.

- The shell automatically attempts to connect to a MongoDB server on startup, so make sure you start mongod before starting the shell.
- The shell is a full-featured JavaScript interpreter, capable of running arbitrary JavaScript programs.

- On startup, the shell connects to the test database on a MongoDB server and
- assigns this database connection to the global variable db.
- This variable is the primary access point to your MongoDB server through the shell.

Let us Try these

- > db
- >use subhashdb
- >db

```
var post = {"title" : "My Blog Post",
        "content" : "Here's my blog post.",
        "date" : new Date()}


db.blog.insert(post)

db.blog.find()
```

- You can see that an "_id" key was added and
- that the other key/value pairs were saved as we entered them.

Read

- find and findOne can be used to query a collection.
- If we just want to see one document from a collection, we can use findOne:

```
db.blog.findOne()
```

find and findOne

- can also be passed criteria in the form of a query document.
- This will restrict the documents matched by the query.
- The shell will automatically display up to 20 documents matching a find,
  - but more can be fetched

Update

- If we would like to modify our post,
  - we can use update.
- update takes (at least) two parameters:
  - the first is the criteria to find which document to update,
  - the second is the new document

Step # 1

- The first step is to modify the variable post and add a "comments" key:

```
post.comments = []
```

Step # 2

- Then we perform the update, replacing the post titled “My Blog Post” with our new version of the document:

```
db.blog.update({title : "My Blog Post"}, post)
```

Step # 3

- Now the document has a "comments" key. If we call find again, we can see the new key:

db.blog.find()

Delete

- remove permanently deletes documents from the database.
- Called with no parameters, it removes all documents from a collection.
- It can also take a document specifying criteria for removal

Delete

- For example, this would remove the post we just created:

```
db.blog.remove({title : "My Blog Post"})
```

```
db.blog.remove({})
```

```
db.blog.drop()
```

Remove vs Drop

- Removing and Dropping a collection is mostly implementation detail.
- Removing a collection requires an one by one update of internal state that happen to exists in the collection.
- Dropping a collection requires freeing up some large data structures inside the database of data files.

Support collections

- Can use a separate collection that will keep the last used value in the sequence.
- To increment the sequence, first we should query the last used value.
- Then, use the operator $inc to increment the value.

- *See the example*

Basic Data Types

- Documents in MongoDB can be thought of as “JSON-like” in that they are conceptually similar to objects in JavaScript.
- On the other hand, JSON’s expressive capabilities are limited because the only types are null, boolean, numeric, string, array, and object.

Mongo Shell - more

- mongo some-host:30000/myDB
- mongo --nodb
- Inside mongo shell
  - conn = new Mongo("some-host:30000");
  - db = conn.getDB("myDB")

Running Scripts with the Shell

- You have used the shell interactively,
- but you can also pass the shell JavaScript files to execute.
- Simply pass in your scripts at the command line:

```
$ mongo script1.js script2.js script3.js
```

More on Mongo

- If you want to run a script using a connection
  - to a non-default host/port mongod,
  - specify the address first, then the script(s):

```
$ mongo --quiet server-1:30000/foo script1.js
script2.js script3.js
```

- Scripts have access to the db variable
  - (as well as any other globals).
- However, shell helpers such as
  - "use db" or "show collections" do not work from files

Helper	Equivalent
use foo	db.getSisterDB("foo")
show dbs	db.getMongo().getDBs()
show collections	db.getCollectionNames()

- You can also use scripts to inject variables into the shell.
- For example, we could have a script that simply initializes helper functions that you commonly use.

- *See the example*

Subhash

That's all

End of Session

©

subhash.ep@gmail.com
linkedin.com/in/subhashep