MongoDB

BootCamp

Part 1 - Introduction and basic queries

Part 2 - Aggregation

Akhil Sharma

Document Oriented Database

- Not Relational
- Easier Scaling (partitioning / horizontal scaling)
- Efficient Indexing
- Faster
- Row = document
- Table = collections
- No pre-defined schemas
- keys and values don't have to be of fixed types and sizes

It's Different

- Adding and removing is wayyyy simpler due to lack of schema
- Development faster, iterations easier, deployments quicket and migrations simpler
- Easier to experiment
- Easy-to-use protocol for storing large files and file metadata

Use Cases

- Big Data: If you have huge amount of data to be stored in tables, MongoDB has built-in solution for partitioning and sharding your databaseDevelopment faster, iterations easier, deployments quicket and migrations simple

- Unstable Schema: Adding a new field does not effect old documents and will be very easy.

- Distributed data: Since multiple copies of data are stored across different servers, recovery of data is instant and safe even if there is a hardware failure.

Official Drivers.

- C, C++, Rust, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go, and Erlang.
- Companies using - EA, Cisco, Shutterfly, Adobe, Ericsson, Craigslist, eBay, and Foursquare.
- Latest Version - 6.x

Basic Concepts

- Single instance of MongoDB can host multiple independent databases, each of which contains its own collections
- Every document has a special key, "_id", that is unique within a collection.
- MongoDB is distributed with a simple but powerful tool called the mongo shell. The mongo shell provides built-in support for administering MongoDB instances and manipulating data using the MongoDB query language.
- MongoDB is type-sensitive and case-sensitive
  {"count" : 5} {"count" : "5"} {"Count" : 5}
- Documents in MongoDB cannot contain duplicate keys
  {"greeting" : "Hello, world!", "greeting" : "Hello, MongoDB!"}

Naming conventions

- The empty string ("") is not a valid collection name.
- You should not create any collections with names that start with system, a prefix reserved for internal collections
- User-created collections should not contain the reserved character $ in their names
- Collection names may not contain any null character
- Database names are case-insensitive.
- Database names are limited to a maximum of 64 bytes.
- Reserved databases - admin, local, config

Data Types

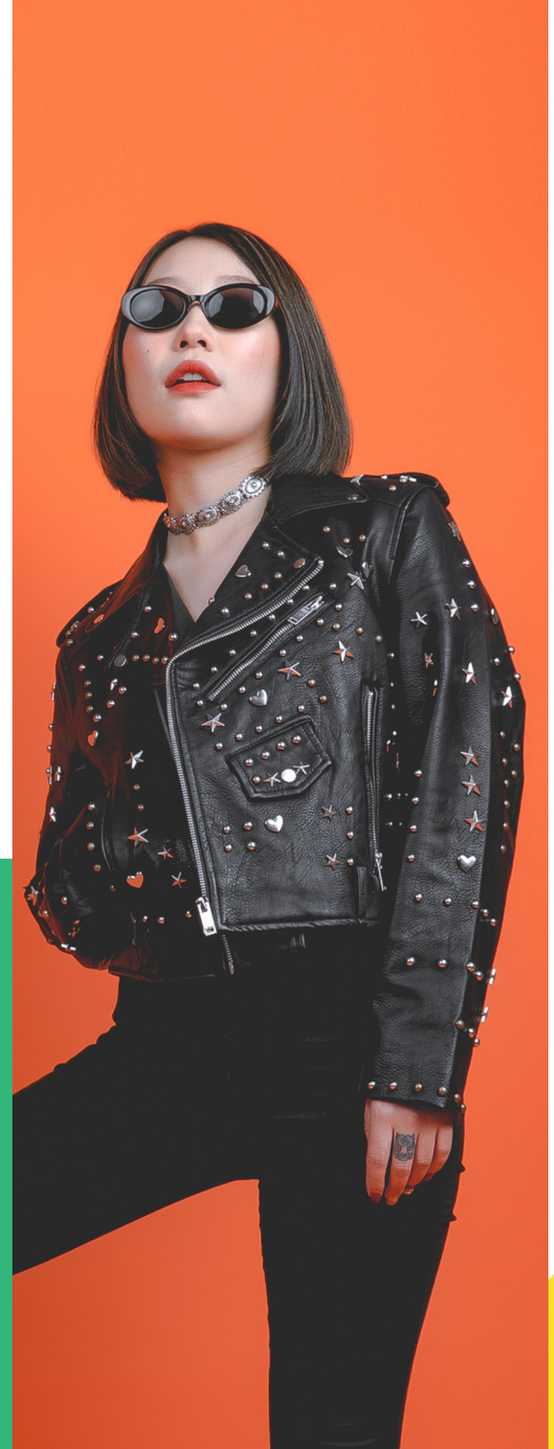
- *Null* - *{"x" : null}*
- *Boolean* - {"x" : true}
- *Number* - {"x" : 3.14} or {"x" : 3}
- *String* - {"x" : "foobar"}
- *Date* - {"x" : new Date()}
- *Regex* - {"x" : /foobar/i}
- *Arrays* - {"x" : ["a", "b", "c"]}
- *Embedded Docs* - {"x" : {"foo" : "bar"}}
- *Object ID* - {"x" : ObjectId()}
- *Code* - {"x" : function() { /* ... */ }}

One to one

```
{
    _id: "joe",
    name: "Joe Bookreader",
    address: {
                street: "123 Fake Street",
                city: "Faketon",
                state: "MA",
                zip: "12345"
              }
}
```

One to many

```
{
   "_id": "joe",
   "name": "Joe Bookreader",
   "addresses": [
                 {
                   "street": "123 Fake Street",
                   "city": "Faketon",
                   "state": "MA",
                   "zip": "12345"
                 },
                 {
                   "street": "1 Some Other Street",
                   "city": "Boston",
                   "state": "MA",
                   "zip": "12345"
                 }
               ]
 }
```



× Many to many

user document

```
{
  _id: <ObjectId1>,
  username: "123xyz"
}
```

contact document

```
{
  _id: <ObjectId2>,
  user_id: <ObjectId1>,
  phone: "123-456-7890",
  email: "xyz@example.com"
}
```

access document

```
{
  _id: <ObjectId3>,
  user_id: <ObjectId1>,
  level: 5,
  group: "dev"
}
```