MongoDB

INTRO

What is No SQL ?

Why & When No SQL ?

Play With MongoDB

What is No SQL ?

Catch-all term for databases that generally aren't relational and don't have a query language like SQL

Schema Less or dynamic schema

NoSQL is a term coined by **Carlo Strozzi** and repurposed by **Eric Evans**

No SQL Types

There have been various approaches to classify NoSQL databases

- **Column** : Cassandra
- **Document** : MongoDB , CouchDB
- **Key–value** : FoundationDB
- **Graph** : Neo4J

Why No SQL ?

Big Data is one of the key forces driving the growth and popularity of NoSQL for business.

A Big Data project is normally typified by:

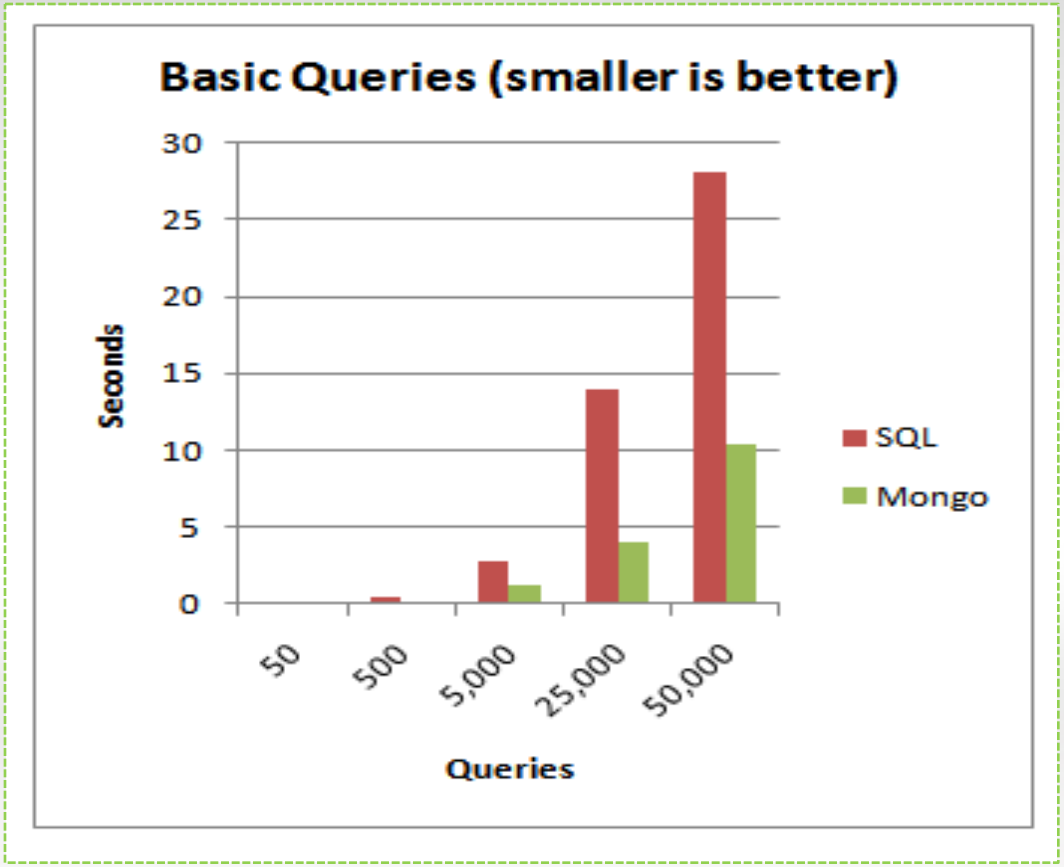
High data velocity: lots of data coming in very quickly, possibly from different locations.

Data variety: storage of data that is structured, semi-structured and unstructured.

Data volume: data that involves many terabytes or petabytes in size.

Data complexity: data that is stored and managed in different locations or data centers.

Why No SQL ?


Basic Queries (smaller is better)
30
25
20
15
10
5
0
Seconds
50
500
5,000
25,000
50,000
Queries
SQL
Mongo

When No SQL ?

Your relational database will not scale to your traffic at an acceptable cost.

In a NoSQL database, there is no fixed schema and no joins. NoSQL can take advantage of "scaling out". Scaling out refers to spreading the load over many commodity systems.

You have local data transactions which do not have to be very durable. e.g. "liking" items on websites.

Object-oriented programming that is easy to use and flexible

When Not No SQL ?

A lot of transactions

a lot of relations in my business

What Is MongoDB ?

Open-source NoSQL database that provides high performance , high availability , and automatic scaling .

Document-oriented

Great for unstructured data, especially when you have a lot of it

Document Database (BSON)

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

```
{
  name: "sue",                          ← field: value
  age: 26,                              ← field: value
  status: "A",                          ← field: value
  groups: [ "news", "sports" ]          ← field: value
}
```

The advantages of using documents

Documents correspond to native data types in many programming languages.

Embedded documents and arrays reduce need for expensive joins.

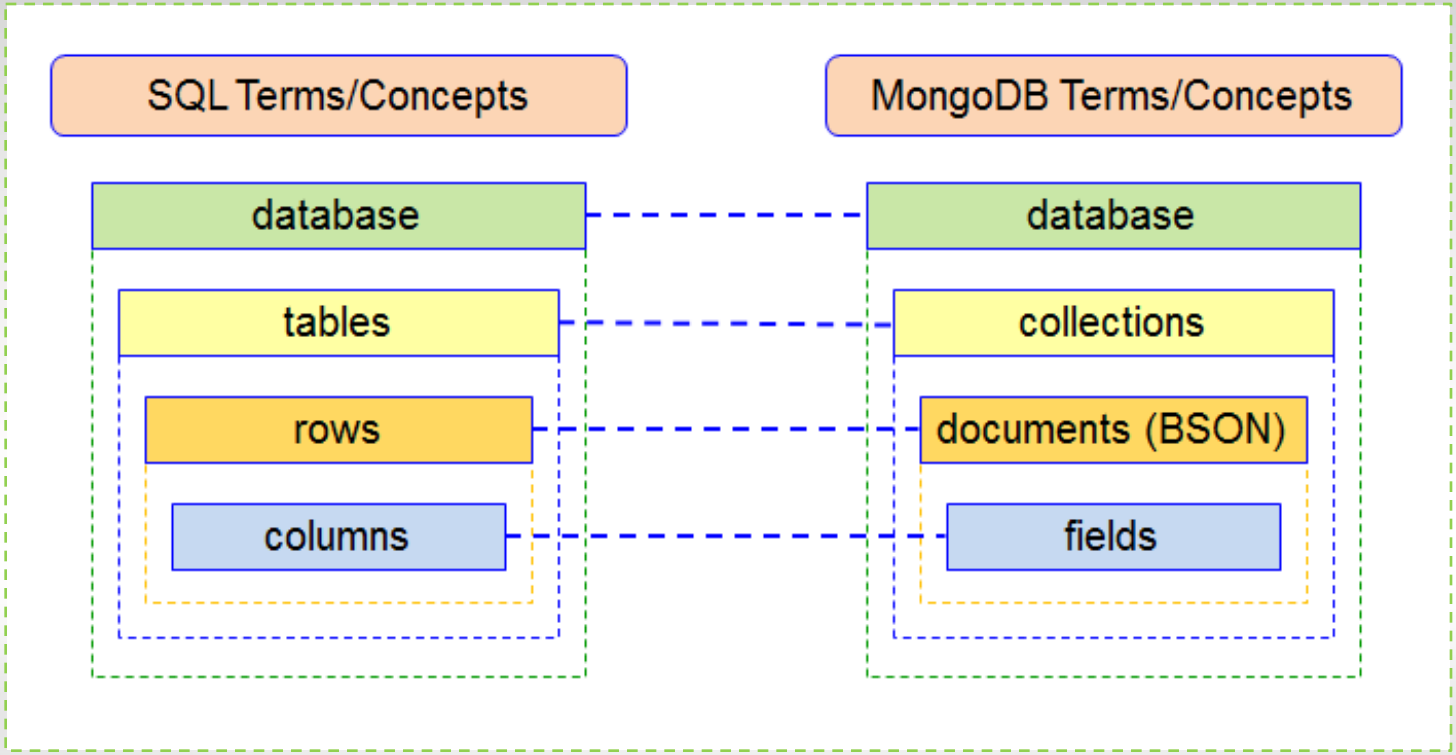
Why MongoDB ?

High Performance（Embedded data models）

High Availability（Automatic Failover）

Horizontal Scalability（Sharding）

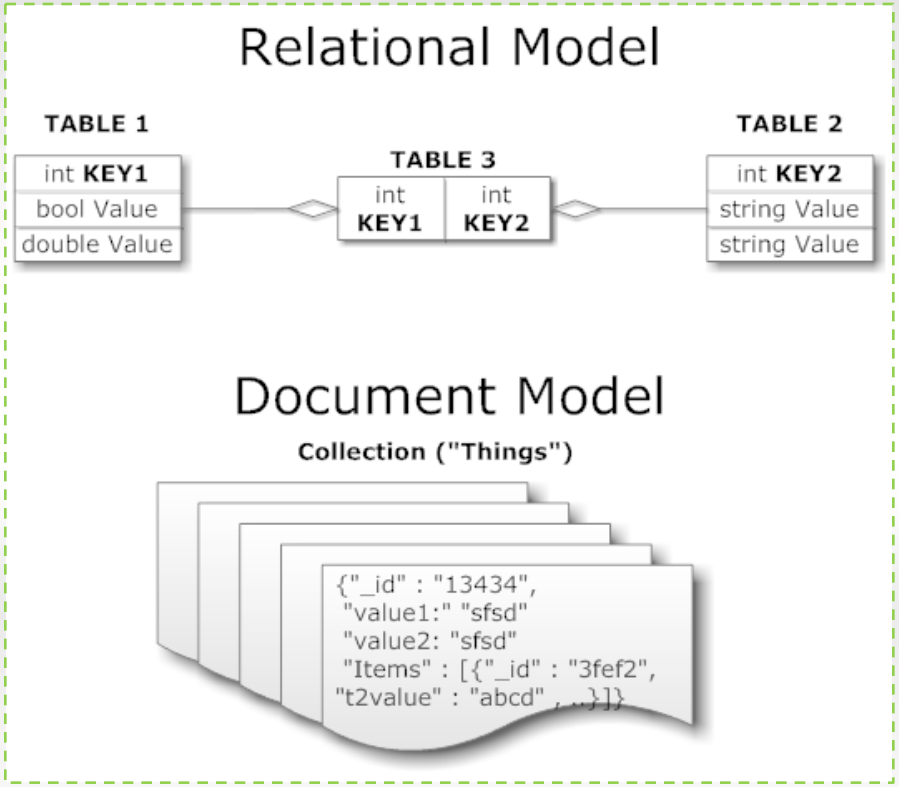
MongoDB comparison to SQL



The main difference? SQL is *relational* and MongoDB is *document-oriented*.

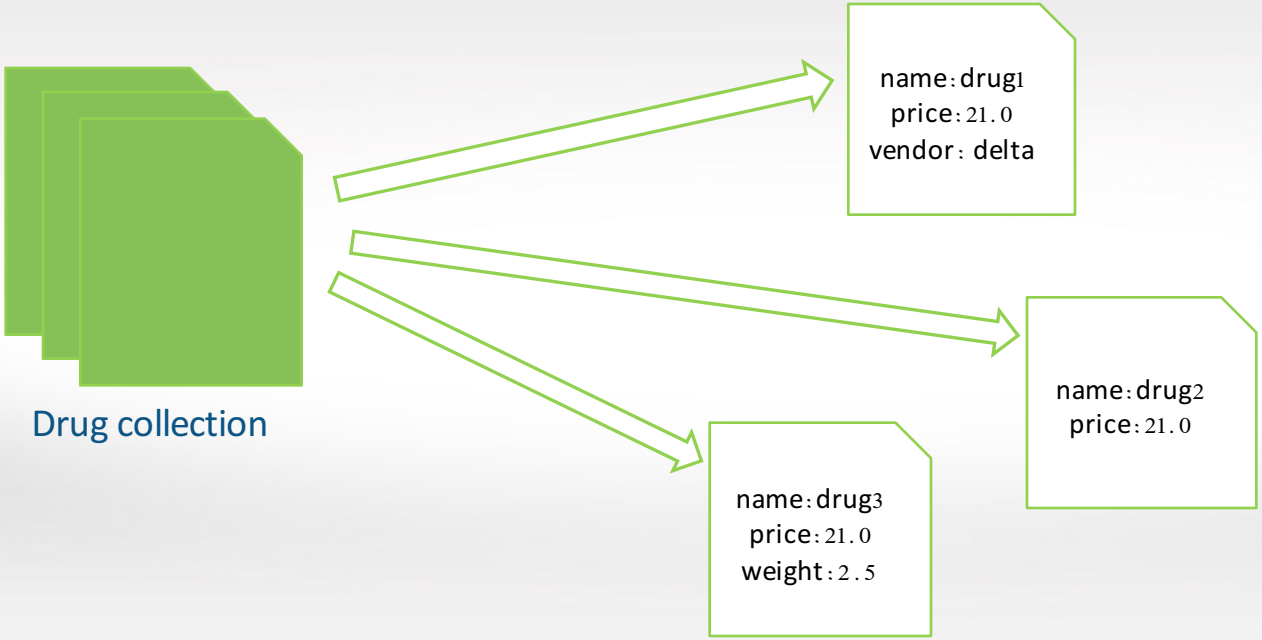
Relational vs. Document-oriented

Relational database management systems save data in rows within tables. MongoDB saves data as documents within collections.



Collections

Collections are simply groups of documents. Since documents exist independently, they can have different fields.



This is referred to as a “dynamic schema”

BSON Documents

MongoDB stores data records as BSON documents. BSON is a binary representation of JSON documents, though it contains more data types than JSON

JSON

Strings
Numbers
Booleans
Arrays
Objects
Null

BSON

JSON Data Types +
ObjectID
Date
binData
maxKey
minKey

BSON Documents

```
var mydoc = {
        __id : ObjectId("5099803df3f4948bd2f98391"),
        name : { first : "Alan", last : "Turing" },
        birth : new Date('Jun 23, 1912'),
        death : new Date('Jun 07, 1954'),
        contribs : [ "Turing machine", "Turing test", "Turingery" ],
        views : NumberLong(1250000)
    }
```

The maximum BSON document size is 16 megabytes.

Installation

Current Mongo Version IS 3.4

Create a /etc/yum.repos.d/mongodb-org-3.4.repo file so that you can install MongoDB directly, using yum.

```
[mongodb-org-3.4]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.4/x86_64/
gpgcheck=1
 enabled=1
 gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc
```

Then write this command in your terminal

sudo yum install -y mongodb-org

configuration file

/etc/mongod.conf

Installation – mongodb-org

mongodb-org : A metapackage that will automatically install the four component packages

mongodb-org-server "mongod daemon"

mongodb-org-mongos "mongos daemon"

mongodb-org-shell "mongo daemon"

mongodb-org-tools

mongodb-org-server "mongod"

mongod is the primary daemon process for the MongoDB system. It handles data requests, manages data access, and performs background management operations.

Listen on **port** 27017

Open your terminal and write "**mongod**" with some options such as

--port

--dbpath

--maxConns

mongodb–org–shell “mongo”

mongo is an interactive JavaScript shell interface to MongoDB, which provides a powerful interface for systems administrators as well as a way for developers to test queries and operations directly with the database. mongo also provides a fully functional JavaScript environment for use with a MongoDB. This document addresses the basic invocation of the mongo shell and an overview of its usage.

Open your terminal and write “**mongo**” with some options such as

––port

––host

––username ‹username›

––password ‹password›

Files : ~/.dbshell,~/.mongorc.js

Play with mongo shell

Ensure that MongoDB is running before attempting to start the mongo shell.

First run "mongod" from your terminal ➔ mongo server

Second run "mongo" from another terminal ➔ mongo shell

To display the database you are using, type **db**:

›db ➔ will display "test" which is default database

To list the available databases, use the helper **show dbs**

›**show dbs**

To switch databases, issue the use ‹db› helper ➔ create it if not exists

Using shell

MongoDB comes with helper methods to make it easy to interact with the database.

db

show dbs

use ‹database name›

help

Documents are always stored in collections within a database.

Collection Operations

MongoDB stores documents in collections. Collections are analogous to tables in relational databases.

Implicitly creation

```
db.myNewCollection2.insert({x:1})
db.myNewCollection3.createIndex({y:1})
```

explicitly creation

```
db.createCollection(name, options)
```

db.createCollection

```
db.createCollection(<name>,{
                    capped: <boolean>,
                    autoIndexId: <boolean>,
                    size: <number>,
                    max: <number>,
                    validator: <document>,
                } )
```

Create Collection With Validation

```
db.createCollection( "contacts",
   {
      validator: { $or:
         [
            { phone: { $type: "string" } },
            { email: { $regex: /@mongodb\.com$/ } },
            { status: { $in: [ "Unknown", "Incomplete" ] } }
         ]
      }
   }
)
```

MongoDB CRUD Operations

Insert Operation

Read Operation

Delete Operation

Update Operation

Insert Document

db.collection.insert()

db.collection.insertOne()

db.collection.insertMany()

```
db.users.insert (
   {
     name: "sue",
     age: 26,
     status: "A"
   }
)
```

collection

field: value

field: value

field: value

document

Read Operation

```
db.collection.find()
```

```
db.collection.find().pretty()
```

```
db.collection.find({query})
```

Update Operation

db.collection.update()

db.collection.updateOne()

db.collection.updateMany()

db.collection.replaceOne()

```
db.users.update(                    ← collection
   { age: { $gt: 18 } },            ← update criteria
   { $set: { status: "A" } },       ← update action
   { multi: true }                  ← update option
)
```

Delete Operation

db.collection.remove()

db.collection.deleteOne()

db.collection.deleteMany()

```
db.users.remove(          ← collection
    { status: "D" }       ← remove criteria
)
```

bulkWrite

Performs multiple write operations with controls for order of execution.

db.collection.bulkWrite()

```
db.collection.bulkWrite(
   [
      { insertOne : <document> },
      { updateOne : <document> },
      { updateMany : <document> },
      { replaceOne : <document> },
      { deleteOne : <document> },
      { deleteMany : <document> }
   ],
   { ordered : false }
)
```

Validations

MongoDB will only enforce a few rules, which means we'll need to make sure data is valid client-side before saving it.

No other document shares same __id

No syntax errors

Document is less than 16mb

Security

Add Database Users with Roles

Manage Users

Manage Roles

Add User

open mongo shell and follow these steps :

use admin

db.createUser({user:"admin",pwd:"123456",roles:[{role:"admin",db:"admin"}]})

close mongo shell and mongod server

run mongod server with --auth option ➔ mongod --auth

run mongo shell with these options :

mongo –u admin -p "123456" --authenticationDatabase "admin"

Manage User

```
db.updateUser("username",{roles:[]})
```

```
db.grandRolesToUser("username",[roles])
```

```
db.revokeRolesFromUser("username",[roles])
```

```
db.dropUser("username")
```

Create Role

```
db.createRole(
        {
                role:"customRole",
                privileges:[
                        {resource:{db:"facebook",collection:"posts"},actions:["find"]}
                ],
                roles:[]
})
```

Demo

Code Time

Thank You

E–mail Address :

ahmedcs2012@gmail.com