Object
Partners
Inc.

MongoDB: The Gateway Drug to NoSQL

Jon DeJong
Chief Software Technologist

@jondejong
https://github.com/jondejong/CJUGMongoDemo

Object **Partners** Inc.

Who are we? Why are we here?

- Flat consulting firm with 100(ish) consultants in 6 states
- Historically on the JVM (Spring/Hibernate, Groovy, GR8, Open Source)
- Changing with the world (JS, Mobile, Big Data, Microservices, DevOps)
- Engage in the local communities

What are we doing here?

This is a dev talk

(not a devops talk)

Overview of MongoDB

- What / How / Why
- Installing / Running

What are we doing here?

Java Driver

Since this is a group with a J in the name, let's start here

What are we doing here?

The GORM Plugin

MongoDB on GORM is the gateway drug to NoSQL

Big data…

MongoDB Approach

- Sharding
- Replica Sets

To SQL or to NoSQL?

Structure (and integrity)

vs.

No Structure (no [ok less] integrity)

Normalized

vs.

Denormalized

ID	FIRST_NAME	LAST_NAME	ADDRESS_ID
1	Jonny	Johnson	1
2	Jamie	Johnson	1
3	Jimmy	Jameson	2
4	Jackie	Jackson	3

ID	LINE_1	CITY	STATE	ZIP
1	5312 Park Avenue	Minneapolis	MN	55417
2	444 N Wabash Ave	Chicago	IL	60611
3	175 N State St	Chicago	IL	60601

Typical relational database

Normalized Data

- structure
- relational
- data integrity

ID	LINE_1	LAST_NAME	ADDRESS_LINE_1	CITY	STATE	ZIP
1	Jonny	Johnson	5312 Park Avenue	Minneapolis	MN	55417
2	Jamie	Johnson	5312 Park Avenue	Minneapolis	MN	55417
3	Jimmy	Jameson	444 N Wabash Ave	Chicago	IL	60611
4	Jackie	Jackson	175 N State St	Chicago	IL	60601

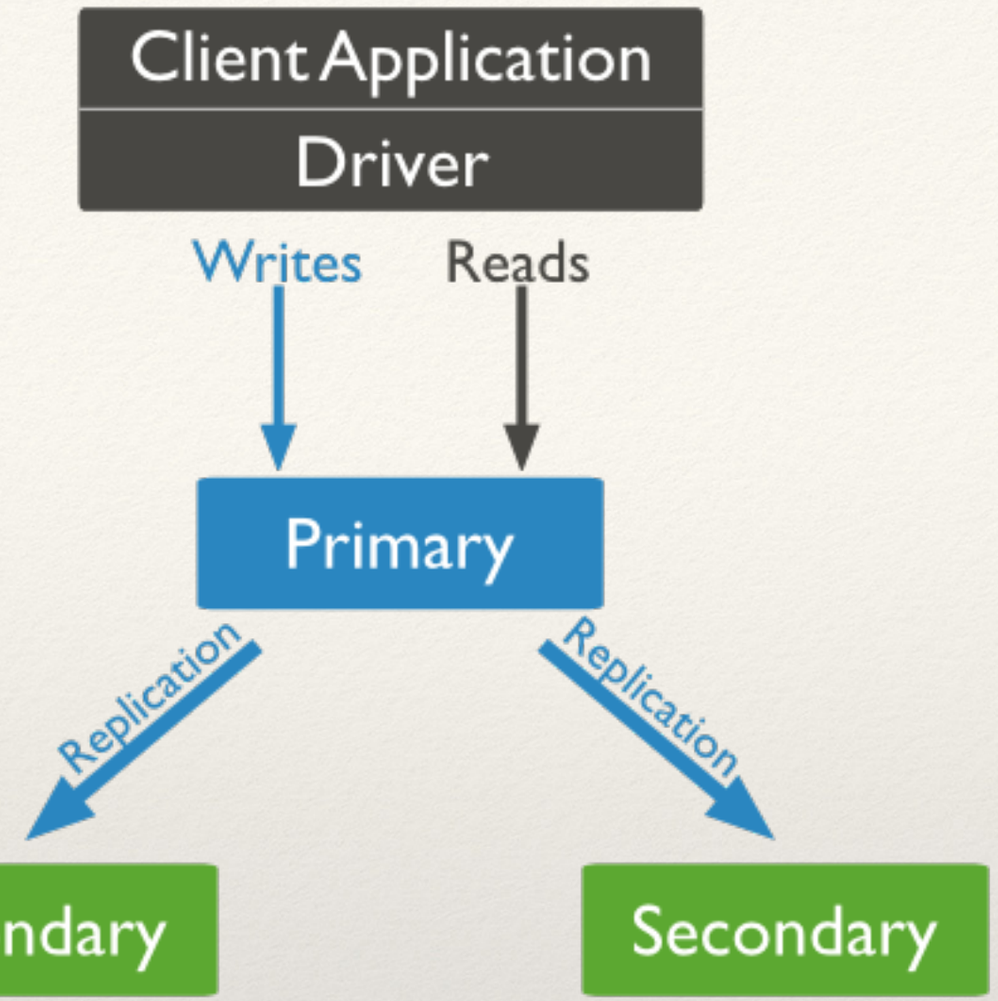
Denormalized Data Table

Denormalized Data

- No joins
- Possible data duplication
- Possible loss of data integrity

Let’s recap this debate

picking the right tool is important

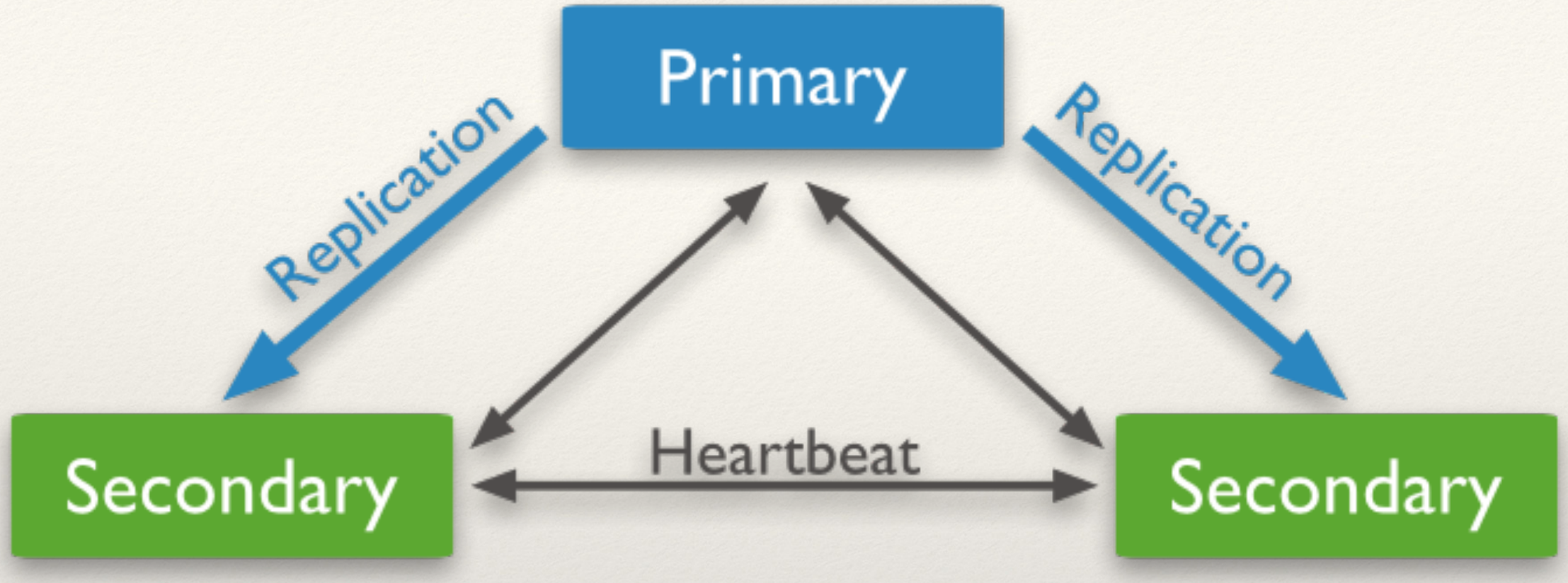


MongoDB

Replica Sets

The secondaries replicate the primary's oplog and apply the operations to their data sets. Secondaries' data sets reflect the primary's data set.

Source: http://docs.mongodb.org/manual/core/replication-introduction/

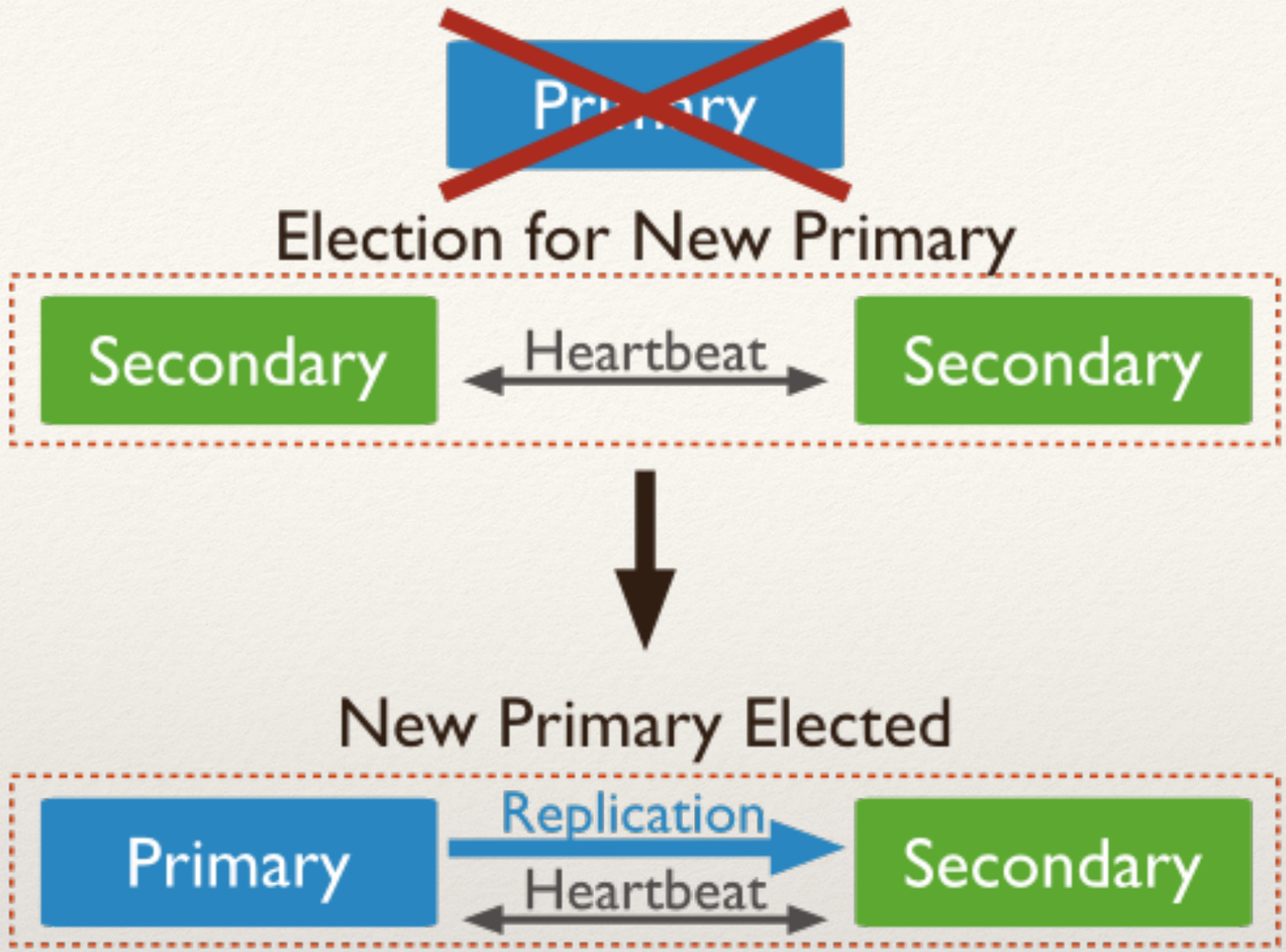


MongoDB

Replica Sets

The secondaries replicate the primary's oplog and apply the operations to their data sets. Secondaries' data sets reflect the primary's data set.

Source: http://docs.mongodb.org/manual/core/replication-introduction/

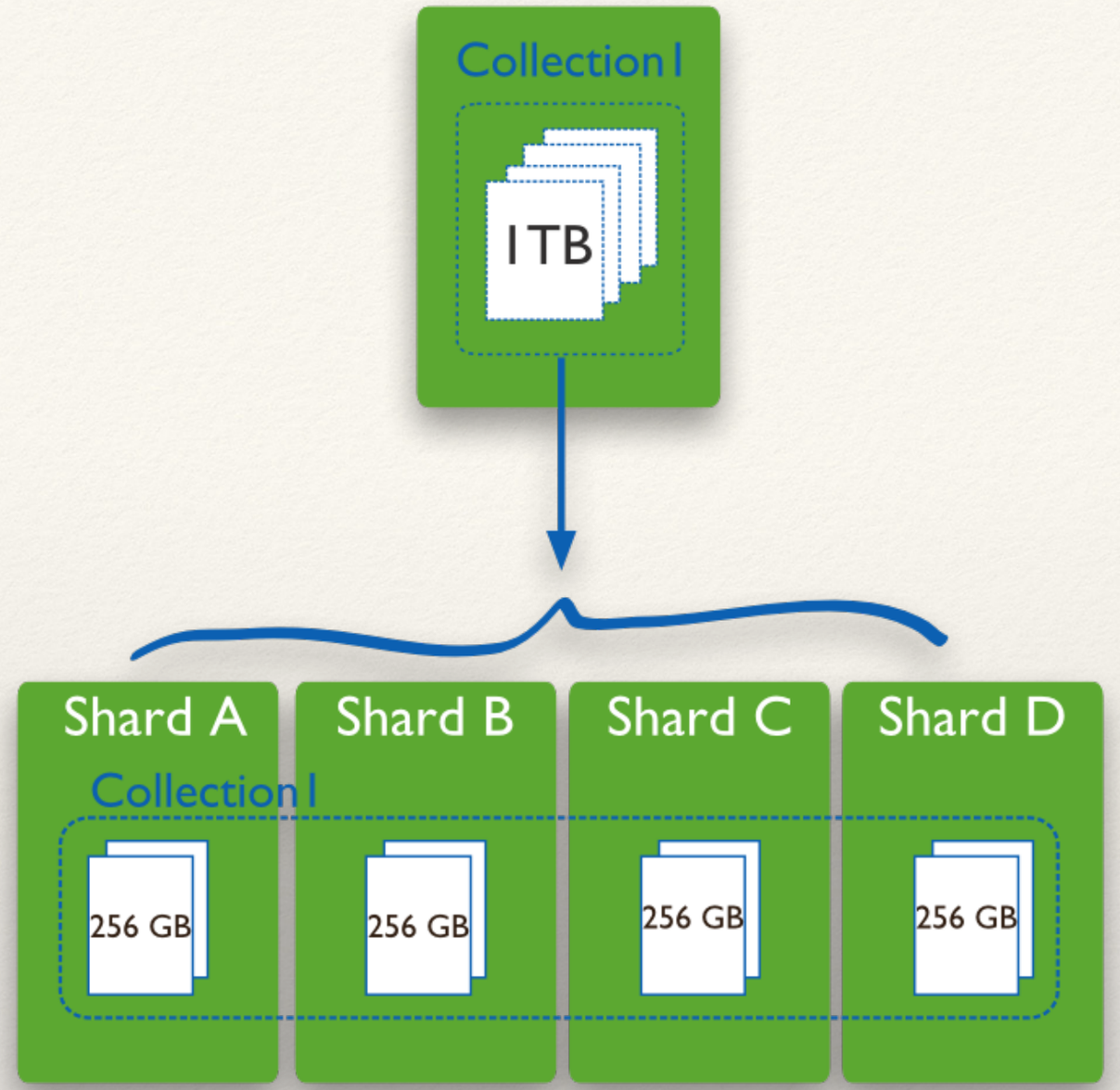


MongoDB

Replica Sets

The secondaries replicate the primary's oplog and apply the operations to their data sets. Secondaries' data sets reflect the primary's data set.

Source: http://docs.mongodb.org/manual/core/replication-introduction/

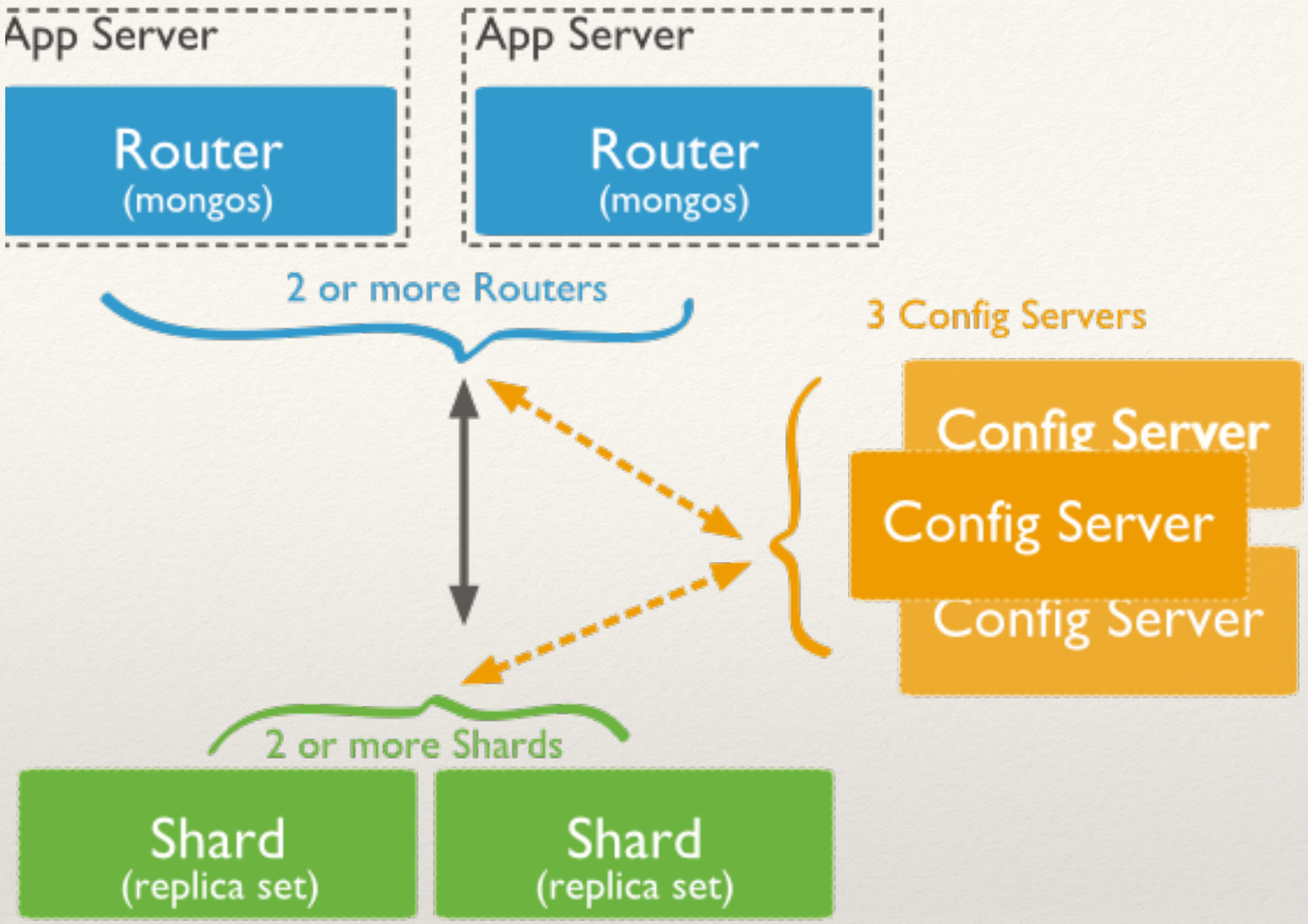


MongoDB

Sharding

Sharding, or horizontal scaling divides the data set and distributes the data over multiple servers.

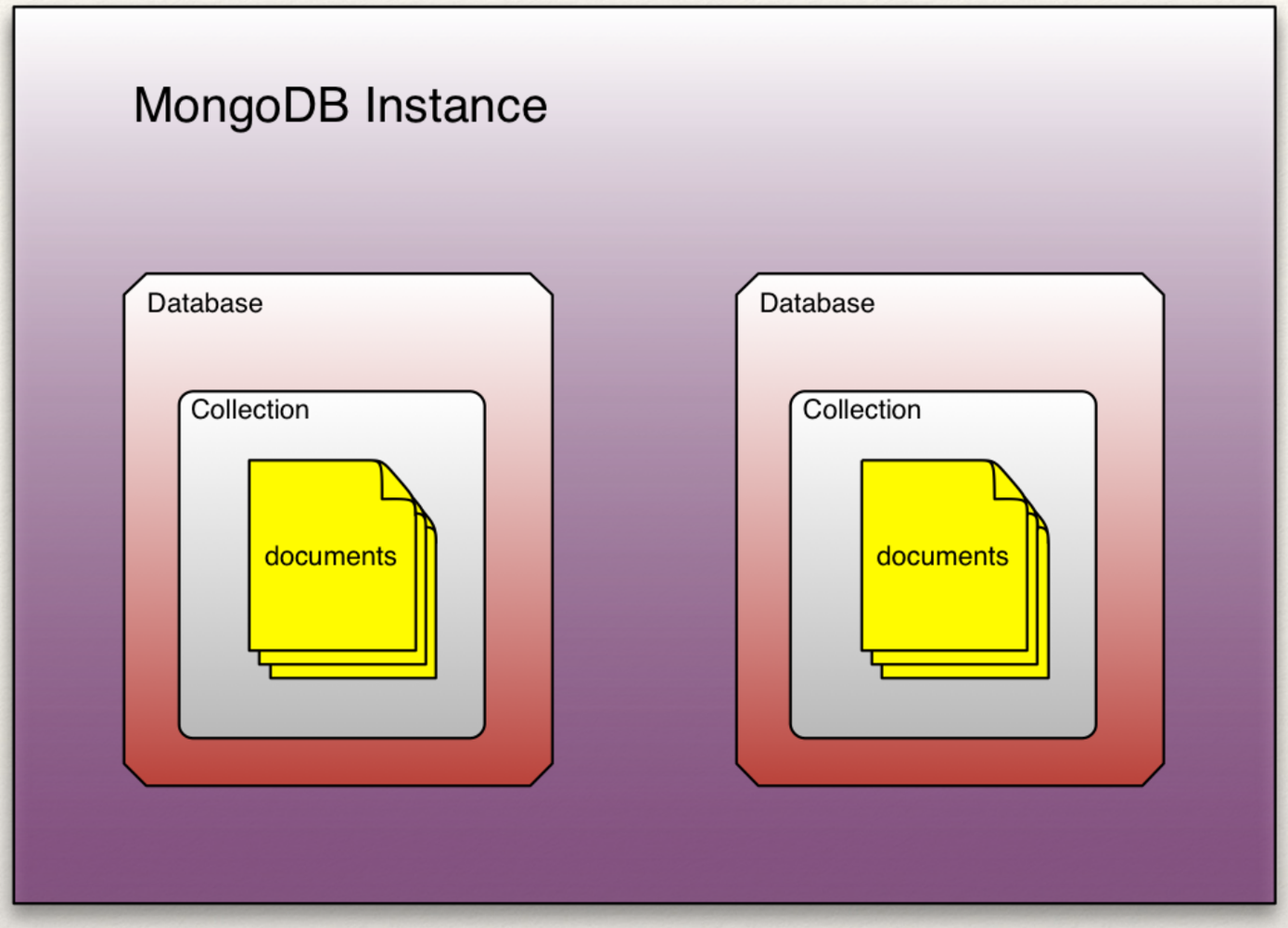
Source: http://docs.mongodb.org/manual/core/sharding-introduction/



Replica Sets Sharded

Sharding, or horizontal scaling divides the data set and distributes the data over multiple servers.

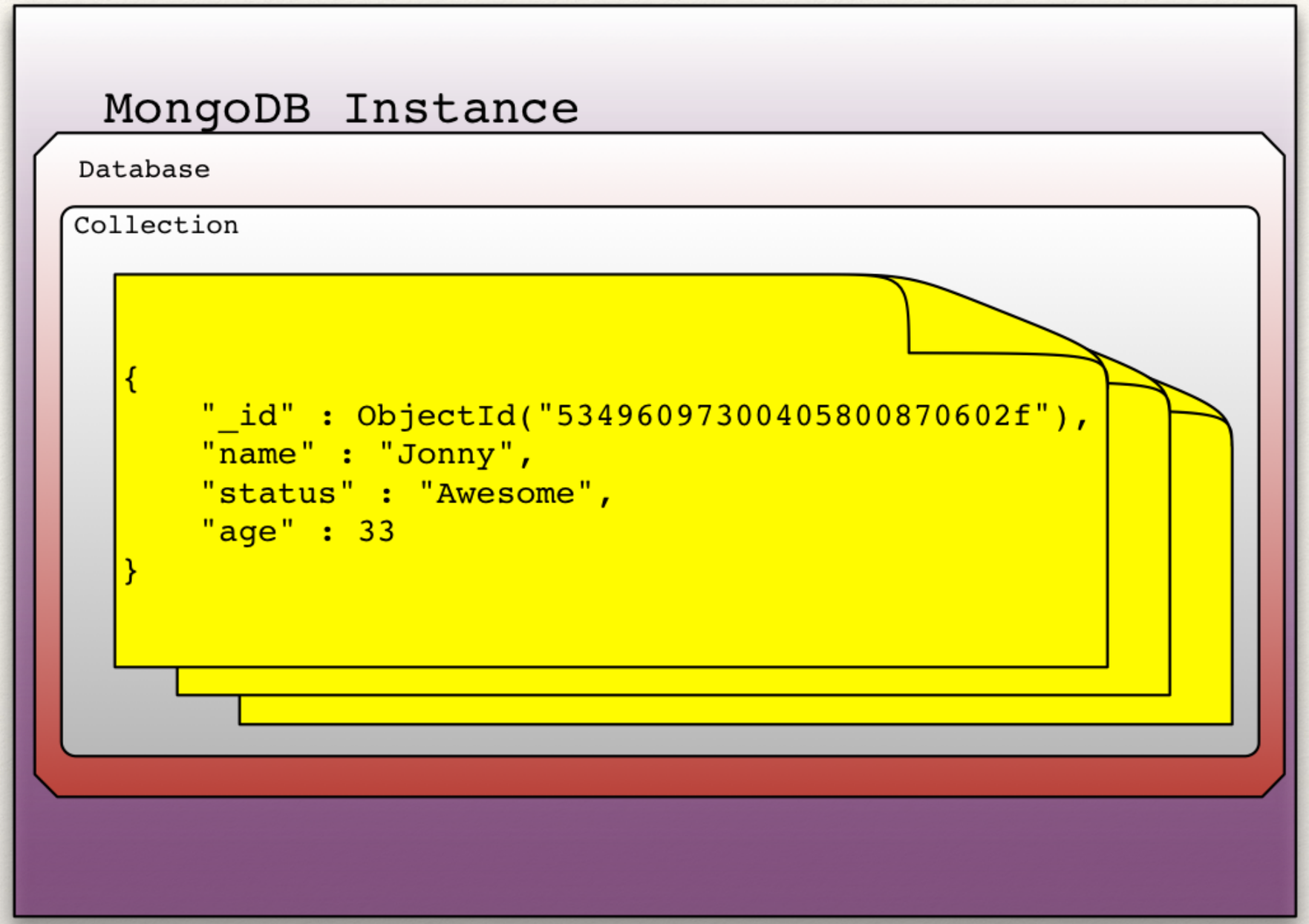
Source: http://docs.mongodb.org/manual/core/sharding-introduction/



What is a MongoDB database?

The MongoDB database

- Databases
- Collections
- Documents



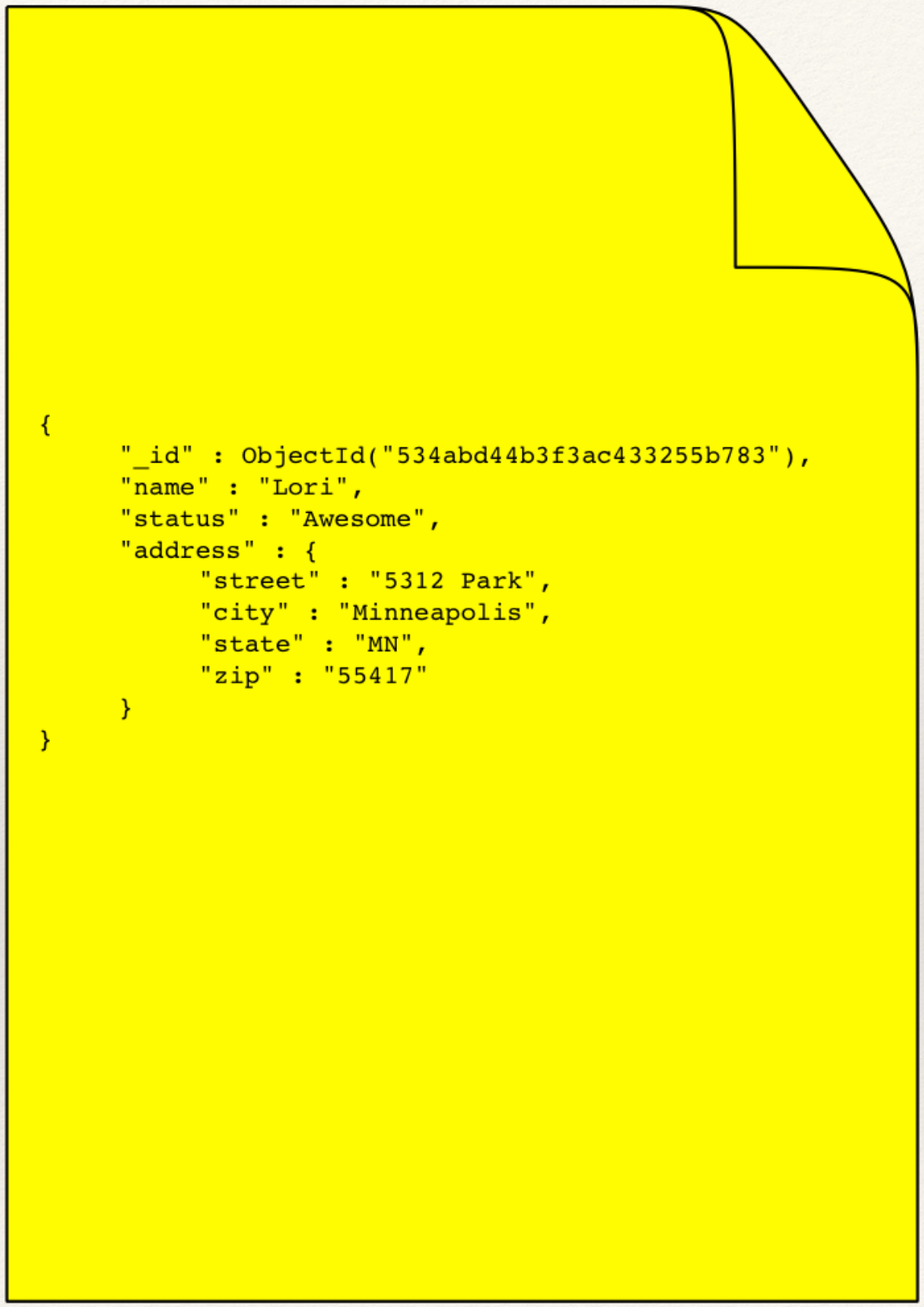
Documents

- BSON
- Unstructured

MongoDB Documents

Embedded Documents

- No structure
- Cannot be referenced directly
- One-To-One relationship modeling
- One-To-Many relationship modeling



MongoDB Documents

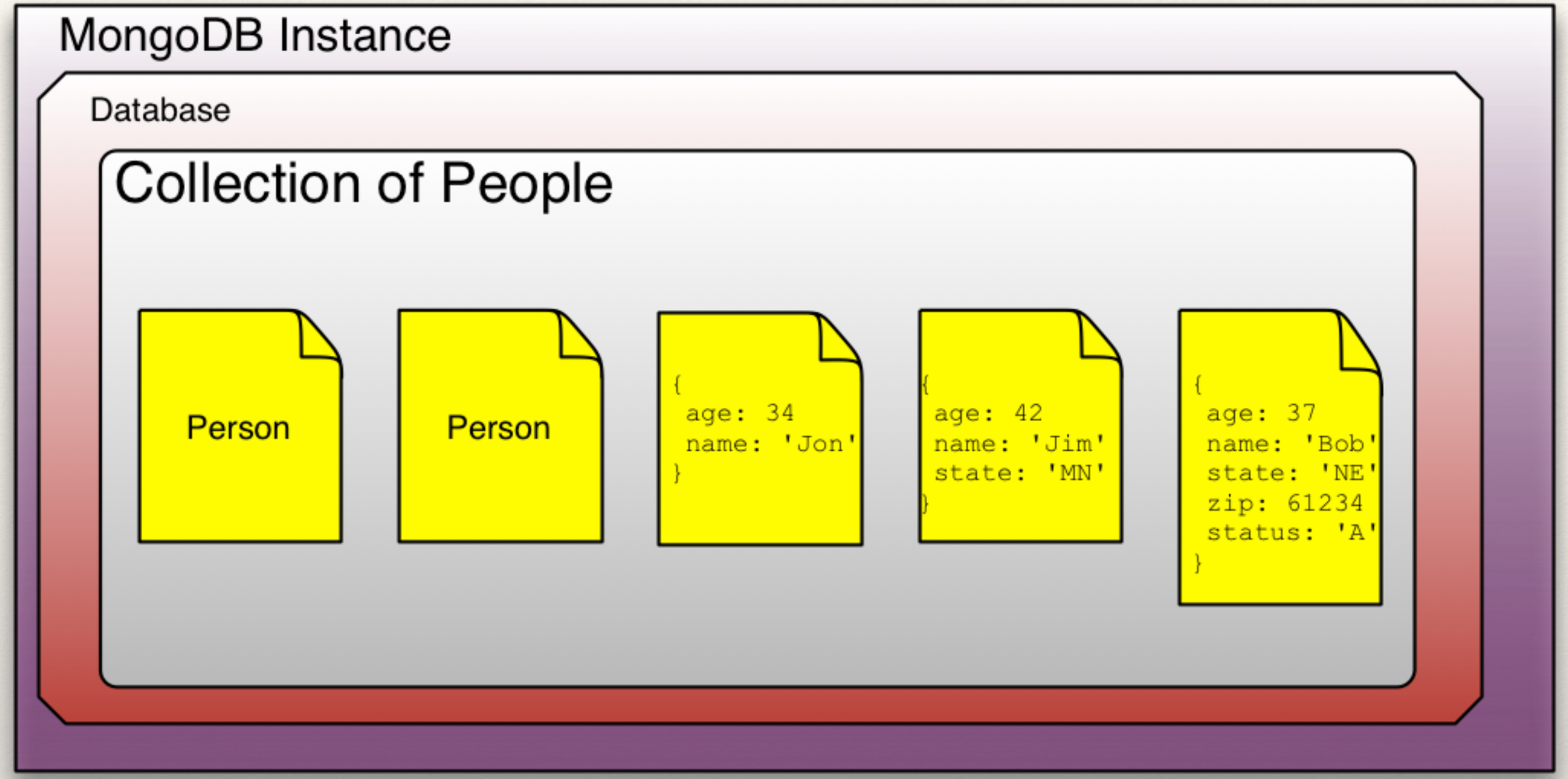
Document References

- Manual References
- DBRef's
  - ID
  - Collection
  - Database (optional)

```
{
	"_id" : ObjectId("534ac07130043542cc59b7af"),
	"address" : {
		"city" : "Minneapolis",
		"line1" : "456 My Street",
		"postalCode" : "55402",
		"state" : "MN"
	},
	"familyName" : "Thatguy",
	"version" : NumberLong(1),
	"people" : [
		ObjectId("534ac05330043542cc59b7ad"),
		ObjectId("534ac05f30043542cc59b7ae")
	],
	"extraData" : "More data about us here."
}
```

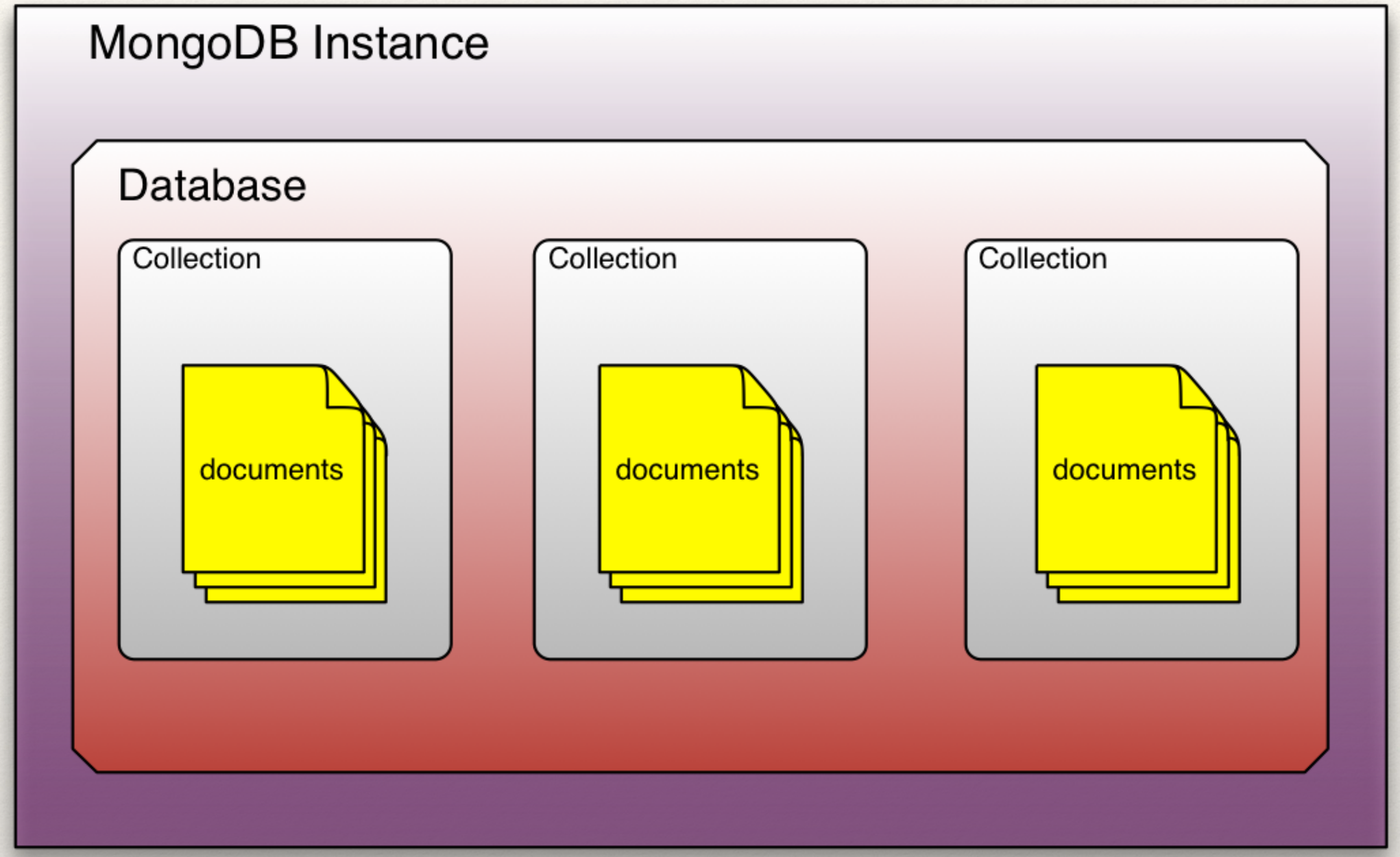
```
{
	"_id" : ObjectId("534ac05330043542cc59b7ad"),
	"age" : 42,
	"firstName" : "Jonny",
	"lastName" : "Thatguy",
	"version" : 0
}
```

```
{
	"_id" : ObjectId("534ac05f30043542cc59b7ae"),
	"age" : 41,
	"firstName" : "Jimmy",
	"lastName" : "Thatguy",
	"version" : 0
}
```



Collections

- Collection of documents
- Documents of any “type”
  - since we know there are no types



Databases

A bunch of collections
grouped together logically

Using MongoDB

- Indexes
- Write Concerns
- Read Preference

Indexing

- Collection scans are inefficient
- Indexes use a B-tree data structure
- Sorted
- Allow sorting on index key without a sort phase

Index Types

- ID
- Single Field
- Compound Field
  - Oder is important: { lastName: 1, firstName: -1 }
- Multikey
- Geospatial
- Hashed

Write Concerns

- Describes the guarantee that MongoDB provides when reporting on the success of a write operation.
- Weaker concern == faster operation
  - Not Acknowledged
  - Acknowledged
  - Journaled
  - Replicated

Read Preference

- Describes how MongoDB clients route read operations to members of a replica set.
- Mess with this and you WILL get stale data

Installing MongoDB

- Download
- Unzip
- Path

Running MongoDB

- mongod
- mongo

```
jondejong$ mongod --dbpath /Users/jondejong/CJUG/data
[initandlisten] waiting for connections on port 27017
[clientcursormon] mem (MB) res:65 virt:2977
[clientcursormon]  mapped (incl journal view):480
[clientcursormon]  connections:0
```

mongod

```
jons-mbp:~ jondejong$ mongo
MongoDB shell version: 2.6.0
connecting to: test
>
```

mongo databases

```
jons-mbp:~ jondejong$ mongo
MongoDB shell version: 2.6.0
connecting to: test
> show dbs
admin  (empty)
local  0.078GB
>
```

mongo databases

```
jons-mbp:~ jondejong$ mongo
MongoDB shell version: 2.6.0
connecting to: test
> show dbs
admin  (empty)
local  0.078GB
> db
test
>
```

mongo databases

```
jons-mbp:~ jondejong$ mongo
MongoDB shell version: 2.6.0
connecting to: test
> show dbs
admin  (empty)
local  0.078GB
> db
test
> use cjug
switched to db cjug
>
```

mongo databases

```
jons-mbp:~ jondejong$ mongo
MongoDB shell version: 2.6.0
connecting to: test
> show dbs
admin  (empty)
local  0.078GB
> db
test
> use cjug
switched to db cjug
> show dbs
admin  (empty)
local  0.078GB
>
```

mongo databases

```
jons-mbp:~ jondejong$ mongo
MongoDB shell version: 2.6.0
connecting to: test
> show dbs
admin  (empty)
local  0.078GB
> db
test
> use cjug
switched to db cjug
> show dbs
admin  (empty)
local  0.078GB
> db
cjug
>
```

mongo databases

```
> show collections
>
```

mongo collections

```
> show collections
> p = { name: "Jonny", status: "Awesome" }
{ "name" : "Jonny", "status" : "Awesome" }
>
```

mongo collections

```
> show collections
> p = { name: "Jonny", status: "Awesome" }
{ "name" : "Jonny", "status" : "Awesome" }
> db.people.insert(p)
WriteResult({ "nInserted" : 1 })
>
```

mongo collections

```
> show collections
> p = { name: "Jonny", status: "Awesome" }
{ "name" : "Jonny", "status" : "Awesome" }
> db.people.insert(p)
WriteResult({ "nInserted" : 1 })
> show collections
people
system.indexes
>
```

mongo collections

```
> show collections
> p = { name: "Jonny", status: "Awesome" }
{ "name" : "Jonny", "status" : "Awesome" }
> db.people.insert(p)
WriteResult({ "nInserted" : 1 })
> show collections
people
system.indexes
> show dbs
admin         (empty)
cjug          0.078GB
local         0.078GB
>
```

mongo collections

```
> p = { name: "Jimmy", status: "Awesome" }
{ "name" : "Jimmy", "status" : "Awesome" }
>
```

mongo find

```
> p = { name: "Jimmy", status: "Awesome" }
{ "name" : "Jimmy", "status" : "Awesome" }
> db.people.insert(p)
WriteResult({ "nInserted" : 1 })
>
```

mongo find

```
> p = { name: "Jimmy", status: "Awesome" }
{ "name" : "Jimmy", "status" : "Awesome" }
> db.people.insert(p)
WriteResult({ "nInserted" : 1 })
> db.people.find()
{ "_id" : ObjectId("5348717b09618d080193dae8"), "name" : “Jonny”,
"status" : "Awesome" }
{ "_id" : ObjectId("53488956 09618d080193dae9"), "name" : "Jimmy",
"status" : "Awesome" }
>
```

mongo find

```
> db.people.find( { name: "Jonny"} )
{ "_id" : ObjectId("5348717b09618d080193dae8"), "name" : “Jonny",
 "status" : "Awesome" }
>
```

mongo find with query

```
> db.people.update({name: "Jimmy"}, {$set: {name: "James"}}, {multi:
 true} )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

mongo update

```
> db.people.update({name: "Jimmy"}, {$set: {name: "James"}}, {multi:
 true} )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> var c = db.people.find()
```

mongo update

```
> db.people.update({name: "Jimmy"}, {$set: {name: "James"}}, {multi:
 true} )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> var c = db.people.find()
> while(c.hasNext()) { printjson(c.next()) }
{
    "_id" : ObjectId("5348717b09618d080193dae8"),
    "name" : "Jonny",
    "status" : "Awesome"
}
{
    "_id" : ObjectId("5348895609618d080193dae9"),
    "name" : "James",
    "status" : "Awesome"
}
>
```

mongo update

```
> db.people.remove({name: "Jonny"})
WriteResult({ "nRemoved" : 1 })
>
```

mongo delete

```
> db.people.remove({name: "Jonny"})
WriteResult({ "nRemoved" : 1 })
> var c = db.people.find()
```

mongo delete

```
> db.people.remove({name: "Jonny"})
WriteResult({ "nRemoved" : 1 })
> var c = db.people.find()
> while(c.hasNext()) { printjson(c.next()) }
{
    "_id" : ObjectId("5348895609618d080193dae9"),
    "name" : "James",
    "status" : "Awesome"
}
>
```

mongo delete

Ok, let's use it for real…

Java Driver

- Allows access to MongoDo
- As close to mongo as possible

Code…

GORM + MongoDB: Gateway Drug to NoSQL

Image Credit: http://www.flickr.com/photos/hans-westbeek/9191573792/

Plugin Replacement For Hibernate (almost)

- Grails Domain Objects Work like Grails Domain Objects
  - Dynamic Finders
  - Criteria Queries
  - Named Queries
- (Almost all the time)
  - No HQL queries
  - No Dirty checking methods
  - No Composite primary keys

BuildConfig.groovy

```
plugins {
    compile ":mongodb:1.3.3"
}
```

DataSource.groovy

```
grails {
    mongo {
        host = "localhost"
        port = 27017
        username = "username"
        password = "password"
        databaseName = "cjug-grails"
    }
}
// Or for replica sets…
grails {
    mongo {
        replicaSet = [ "localhost:27017", "localhost:27018"]
    }
}
```

GORM Domain Mapping

```
class Person {
    static mapWith = “mongo" / / Optional
    ObjectId id
    String firstName
    String lastName
    static mapping = {
        name index:true
        writeConcern WriteConcern.ACKNOWLEDGED
    }
}
```

Dynamic Finders

Person.findByFirstName('Jon')

Person.findByLastName('DeJong')

Code…

Geospatial Queries

- 2D, 2D Sphere
- City.findByLocationNear()
- City.findByLocationWithinBox()
- City.findByLocationWithinCircle()

The Java Driver

When all else fails, you have access to the Java driver, Groovy style

Any Questions?

@jondejong

https://github.com/jondejong/CJUGMongoDemo