#mongodbmadrid

MongoDB Essentials Training

Norberto

Eng|SA, MongoDB

norberto@mongodb.com

mongoDB

Welcome!!!

mongoDB

Instructor

- Norberto Leite
- Senior Solutions Architect
- @nleite
- norberto@mongodb.com

I HAVE THE CAPE.
I MAKE THE FUCKING
WOOSH NOISES

Setting Expectations

- By the end of the next 3 days you should:
  - Be confortable with MongoDB shell
  - Fully understand concepts like Replica Sets and shards
  - Completely able to configure, monitor and administer MongoDB cluster
  - Schema design expertise
  - Driver understanding
  - Debug and analyze queries and log
  - Troubleshoot and understand where to look

Agenda Day 1

- Introduction to MongoDB
  - JSON / BSON
  - Environments
  - Install
- CRUD and MongoDB Shell
- Indexing and Schema Design

Agenda Day 2

- Replication
- Sharding
- Monitoring
- Performance Troubleshooting

Agenda Day 3

- Drivers
- Backups and Recovery
- Extra topics
  - Aggregation Framework
  - MapReduce
  - Geospatial Indexes
  - GridFS

This is not a watch and learn course!

YOU WILL DO LOTS OF HANDS ON!

What is MongoDB?

mongoDB

MongoDB is a _____ database

- Document
- Open source
- High performance
- Horizontally scalable
- Full featured

Document Database

- Not for .PDF & .DOC files
- A document is essentially an associative array
- Document = JSON object
- Document = PHP Array
- Document = Python Dict
- Document = Ruby Hash
- etc

Open Source

- MongoDB is an open source project
- On GitHub
- Licensed under the AGPL
- Started & sponsored by MongoDB Inc (formerly 10gen)
- Commercial licenses available
- Contributions welcome

High Performance

- Written in C++
- Extensive use of memory-mapped files
  i.e. read-through write-through memory caching.
- Runs nearly everywhere
- Data serialized as BSON (fast parsing)
- Full support for primary & secondary indexes
- Document model = less work

Shard 1
Shard 2
Shard 3
Shard N
Horizontally Scalable

Database Landscape

Performance

Better Data Locality

In-Memory Caching

In-Place Updates

Full Featured

- Ad Hoc queries
- Real time aggregation
- Rich query capabilities
- Strongly consistent
- Geospatial features
- Support for most programming languages
- Flexible schema

Document Database

mongoDB

Terminology

RDBMS		MongoDB
Table, View	➜	Collection
Row	➜	Document
Index	➜	Index
Join	➜	Embedded Document
Foreign Key	➜	Reference
Partition	➜	Shard

In a MongoDB based app

We start building our app

and let the schema evolve

MongoDB ERD

Article
•Name
•Slug
•Publish date
•Text
•Author
Comment[]
•Comment
•Date
•Author
Tag[]
•Value
Category[]
•Value
User
•Name
•Email address

Data Model

Relational

PERSON

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rome

NO RELATION

CAR

Car_ID	Model	Year	Value	Pers_ID
101	Bently	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

MongoDB

```
{
  first_name: 'Paul',
  surname: 'Miller',
  city: 'London',
  location:
[45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  }
}
```

Instaltion

mongoDB

Storage Layer

mongoDB

Directory Layout

```
drwxr-xr-x     136          Nov 19 10:12        journal
-rw-------  16777216       Oct 25 14:58        test.0
-rw-------  134217728          Mar 13  2012          test.1
-rw-------  268435456          Mar 13  2012          test.2
-rw-------  536870912          May 11  2012          test.3
-rw-------  1073741824      May 11  2012        test.4
-rw-------  2146435072      Nov 19 10:14        test.5
-rw-------  16777216       Nov 19 10:13        test.ns
```

Directory Layout

- Aggressive pre-allocation (always 1 spare file)
- There is one namespace file per db which can hold 18000 entries per default
- A namespace is a collection or an index

Internal Structure

mongoDB

Internal File Format

Extent Structure

Extents and Records

Extent

mongoDB

To Sum Up: Internal File Format

- Files on disk are broken into extents which contain the documents
- A collection has **one or more extents**
- Extent grow exponentially **up to 2GB**
- Namespace entries in the *ns* file point to the **first extent** for that collection

What's Memory Mapping?

Memory Mapped Files

- All data files are **memory mapped** to Virtual Memory by the OS
- MongoDB **just reads / writes to RAM** in the filesystem cache
- **OS takes care of the rest!**
- Virtual process size = total files size + overhead (connections, heap)
- If journal is on, the virtual size will be roughly doubled

Virtual Address Space

Working With MongoDB

mongoDB

Running MongoDB

```
$ tar –z xvf mongodb-osx-x86_64-2.4.x.tgz
$ cd mongodb-osx-i386-2.4.4/bin
$ mkdir –p /data/db
$ ./mongod
```

Mongo Shell

```
MacBook-Air-:~ $ mongo
MongoDB shell version: 2.4.4
connecting to: test
> db.test.insert({text: 'Welcome to MongoDB'})
> db.test.find().pretty()
{
        "_id" : ObjectId("51c34130fbd5d7261b4cdb55"),
        "text" : "Welcome to MongoDB"
}
```

CRUD

mongoDB

Switch to Your DB

```
>db
test
> use blog
 switching to db blog

> db.users.insert( user )
```

Insert the Record

```
> db.users.insert(user)
```

No collection creation necessary

Find One Record

```
> db.users.findOne()
{
    "_id" : ObjectId("50804d0bd94ccab2da652599"),
    "username" : ”erlichson",
    "first_name" : ”Andrew",
    "last_name" : ”Erlichson"
}
```

_id

- _id is the primary key in MongoDB
- Automatically indexed
- Automatically created as an ObjectId if not provided
- Any unique immutable value could be used

ObjectId

- ObjectId is a special 12 byte value
- Guaranteed to be unique across your cluster
- ObjectId("50804d0bd94ccab2da652599")

```
|----ts-----||---mac---||-pid-||----inc-----|
     4          3        2          3
```

Creating a Blog Post

```
> db.article.insert({
                title: ‘Hello World’,
                body:  ‘This is my first blog post’,
                date: new Date(‘2013-06-20’),
                username: ‘erlichson’,
                tags: [‘adventure’, ‘mongodb’],
                comments: [ ]
})
```

Finding the Post

```
> db.article.find().pretty()
{
    "_id" : ObjectId("51c3bafafbd5d7261b4cdb5a"),
    "title" : "Hello World",
    "body" : "This is my first blog post",
    "date" : ISODate("2013-06-20T00:00:00Z"),
    "username" : "erlichson",
    "tags" : [
        "adventure",
        "mongodb"
    ],
    "comments" : [ ]
}
```

Querying An Array

```
> db.article.find({tags:'adventure'}).pretty()
{
    "_id" : ObjectId("51c3bcddfbd5d7261b4cdb5b"),
    "title" : "Hello World",
    "body" : "This is my first blog post",
    "date" : ISODate("2013-06-20T00:00:00Z"),
    "username" : "erlichson",
    "tags" : [
        "adventure",
        "mongodb"
    ],
    "comments" : [ ]
}
```

Using Update to Add a Comment

```
> db.article.update({_id:
    new ObjectId("51c3bcddfbd5d7261b4cdb5b")},
    {$push:{comments:
    {name: 'Steve Blank', comment: 'Awesome Post'}}})
>
```

Post with Comment Attached

```
> db.article.findOne({_id: new ObjectId("51c3bcddfbd5d7261b4cdb5b")})
{
        "_id" : ObjectId("51c3bcddfbd5d7261b4cdb5b"),
        "body" : "This is my first blog post",
        "comments" : [
                {
                        "name" : "Steve Blank",
                        "comment" : "Awesome Post"
                }
        ],
        "date" : ISODate("2013-06-20T00:00:00Z"),
        "tags" : [
                "adventure",
                "mongodb"
        ],
        "title" : "Hello World",
        "username" : "erlichson"
}
```

Let’s do MongoDB!

MongoDB Shell

mongoDB

What Is the Shell?

Embedded Javascript Interpreter

- vars
- functions
- data structs + types

Global Functions and Objects

- ObjectId("...")
- new Date()
- Object.bsonsize()

MongoDB driver Exposed

- db["collection"].find/count/update
- short-hand for collections

JSON-like stuff

- Doesn't require quoted keys
- Don't copy and paste too much

MongoDB Shell: Advantages

- Debugging Queries / Syntax
- Testing
- Administration
- Scripting Glue

MongoDB Shell: Disadvantages

- Numbers in JS are a pain
  - 32/64-bit int/long → NumberInt() / NumberLong()
  - Primitive numbers are all 64-bit FP doubles
- Dates can be confusing
  - new Date("1/1/1")
  - new ISODate(...)

NOT: Date("1/1/1") → string

Speed Considerations

- Shell
  - JavaScript is slow
  - Always in "write-acknowledged" (safe mode) / GLE
  - Data Conversions
- Server
  - Applies on the server as well
  - Careful with round-tripping numbers

Easy to Use

- Tab completion on most objects
- Built-in help on most objects (.help())
- show
  - profile # 5 most recent ops of 1ms or more
  - users # List all the users of the current db
  - dbs # List all the databases
  - logs # List all available logs
- Most examples use the shell

Loading Scripts

- Commandline
  - --eval switch
  - .js files
- Within the shell
  - load()

Examining Records

- Use db.collection.find() to run a query, or .findOne() to fetch a single object
- Results are printed 20 at a time, type "it" to show the next batch
- Set DBQuery.shellBatchSize to change the size of each batch
- Add .pretty() to the function to "pretty-print" the json output

```
> db.test.findOne()
{ "_id": ObjectId("51dc2e743f89441af9a0b0ff"), "x": 1 }
> db.test.find()
{ "_id": ObjectId("51dc2eba3f89441af9a0b180"), "x": 28 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b181"), "x": 29 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b182"), "x": 30 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b183"), "x": 31 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b184"), "x": 32 }
...
Type "it" for more
> it
{ "_id": ObjectId("51dc2eba3f89441af9a0b185"), "x": 33 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b186"), "x": 34 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b187"), "x": 35 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b188"), "x": 36 }
{ "_id": ObjectId("51dc2eba3f89441af9a0b189"), "x": 37 }
...
```

Exercises

Schema Design

mongoDB

Documents

Provide flexibility and performance

Normalized Data

De-Normalized (embedded) Data

Relational Schema Design

Focus on data storage

Document Schema Design

Focus on data use

Schema Design Considerations

- How do we manipulate the data?
  - Dynamic Ad-Hoc Queries
  - Atomic Updates
  - Map Reduce
- What are the access patterns of the application?
  - Read/Write Ratio
  - Types of Queries / Updates
  - Data life-cycle and growth rate

Data Manipulation

Query Selectors

- **Scalar**: $ne, $mod, $exists, $type, $lt, $lte, $gt, $gte
- **Vector**: $in, $nin, $all, $size

Atomic Update Operators

- **Scalar**: $inc, $set, $unset
- **Vector**: $push, $pop, $pull, $pushAll, $pullAll, $addToSet

Data Access

Flexible Schemas

Ability to embed complex data structures

Secondary Indexes

Multi-Key Indexes

Aggregation Framework

- $project, $match, $limit, $skip, $sort, $group, $unwind

No Joins

Library Management Application

Patrons

Books

Authors

Publishers

An Example

One to One Relations

Modeling Patrons

```
patron = {
    _id: "joe",
    name: "Joe Bookreader”
}

address = {
    patron_id = "joe",
    street: "123 Fake St. ",
    city: "Faketon",
    state: "MA",
    zip: 12345
}
```

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

One to One Relations

Mostly the same as the relational approach

Generally good idea to embed “contains” relationships

Document model provides a holistic representation of objects

An Example

One To Many Relations

Modeling Patrons

```
patron = {
  _id: "joe",
  name: "Joe Bookreader",
  join_date: ISODate("2011-10-15"),
  addresses: [
    {street: "1 Vernon St.", city: "Newton", state: "MA", ...},
    {street: "52 Main St.", city: "Boston", state: "MA", ...}
  ]
}
```

Publishers and Books

- Publishers put out many books
- Books have one publisher

Book

MongoDB: The Definitive Guide,
By Kristina Chodorow and Mike Dirolf
Published: 9/24/2010
Pages: 216
Language: English

Publisher: O'Reilly Media, CA

Modeling Books – Embedded Publisher

```
book = {
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    publisher: {
        name: "O'Reilly Media",
        founded: "1980",
        location: "CA"
    }
}
```

Modeling Books & Publisher Relationship

```
publisher = {
    name: "O'Reilly Media",
    founded: "1980",
    location: "CA"
}

book = {
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English"
}
```

Publisher _id as a Foreign Key

```
publisher = {
    _id: "oreilly",
    name: "O'Reilly Media",
    founded: "1980",
    location: "CA"
}

book = {
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    publisher_id: "oreilly"
}
```

Book _id as a Foreign Key

```
publisher = {
    name: "O'Reilly Media",
    founded: "1980",
    location: "CA"
    books: [ "123456789", ... ]
}

book = {
    _id: "123456789",
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English"
}
```

Where Do You Put the Foreign Key?

Array of books inside of publisher

- Makes sense when many means a handful of items
- Useful when items have bound on potential growth

Reference to single publisher on books

- Useful when items have unbounded growth (unlimited # of books)

SQL doesn't give you a choice, no arrays

Another Example

One to Many Relations

Books and Patrons

Book can be checked out by one Patron at a time

Patrons can check out many books (but not 1000's)

Modeling Checkouts

```
patron = {
    _id: "joe",
    name: "Joe Bookreader",
    join_date: ISODate("2011-10-15"),
    address: { ... }
}

book = {
    _id: "123456789",
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    ...
}
```

Modeling Checkouts

```
patron = {
    _id: "joe",
    name: "Joe Bookreader",
    join_date: ISODate("2011-10-15"),
    address: { ... },
    checked_out: [
        { _id: "123456789", checked_out: "2012-10-15" },
        { _id: "987654321", checked_out: "2012-09-12" },
        ...
    ]
}
```

Denormalization

Provides data locality

Modeling Checkouts: Denormalized

```
patron = {
    _id: "joe",
    name: "Joe Bookreader",
    join_date: ISODate("2011-10-15"),
    address: { ... },
    checked_out: [
        { _id: "123456789",
          title: "MongoDB: The Definitive Guide",
          authors: [ "Kristina Chodorow", "Mike Dirolf" ],
          checked_out: ISODate("2012-10-15")
        },
        { _id: "987654321"
          title: "MongoDB: The Scaling Adventure",
                                    ...
        }, ...
    ]
}
```

Referencing vs. Embedding

- Embedding is a bit like pre-joined data
- Document-level ops are easy for server to handle
- Embed when the 'many' objects always appear with (i.e. viewed in the context of) their parent
- Reference when you need more flexibility

An Example

Single Table Inheritance

Single Table Inheritance

```
book = {
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    kind: "loanable",
    locations: [ ... ],
    pages: 216,
    language: "English",
    publisher: {
        name: "O'Reilly Media",
        founded: "1980",
        location: "CA"
    }
}
```

An Example

Many to Many Relations

Relational Approach

Books and Authors

```
book = {
    title: "MongoDB: The Definitive Guide",
    authors = [
        { _id: "kchodorow", name: "K-Awesome" },
        { _id: "mdirolf", name: "Batman Mike" },
    ]
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English"
}

author = {
    _id: "kchodorow",
    name: "Kristina Chodorow",
    hometown: "New York"
}
```

Relation stored on both sides

```
book = {
    _id: 123456789,
    title: "MongoDB: The Definitive Guide",
    authors = [ "kchodorow", "mdirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English"
}

author = {
    _id: "kchodorow",
    name: "Kristina Chodorow",
    hometown: "Cincinnati",
    books: [ 123456789, ... ]
}
```

An Example

Trees

Parent Links

```
book = {
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    category: "MongoDB"
}

category = { _id: MongoDB, parent: "Databases" }
category = { _id: Databases, parent: "Programming" }
```

Child Links

```
book = {
    _id: 123456789,
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English"
}

category = { _id: MongoDB, children: [ 123456789, ... ] }
category = { _id: Databases, children: ["MongoDB", "Postgres"}
category = { _id: Programming, children: ["DB", "Languages"] }
```

Modeling Trees

- Parent Links
  - Each node is stored as a document
  - Contains the id of the parent
- Child Links
  - Each node contains the id's of the children
  - Can support graphs (multiple parents / child)

Array of Ancestors

```
book = {
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    categories: ["Programming", "Databases", "MongoDB” ]
}

book = {
    title: "MySQL: The Definitive Guide",
    authors: [ "Michael Kofler" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    parent: "MongoDB",
    ancestors: [ "Programming", "Databases", "MongoDB"]
}
```

An Example

Queues

Book Document

```
book = {
    _id: 123456789,
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    available: 3
}

db.books.findAndModify({
    query: { _id: 123456789, available: { "$gt": 0 } },
    update: { $inc: { available: -1 } }
})
```

Questions?

mongoDB