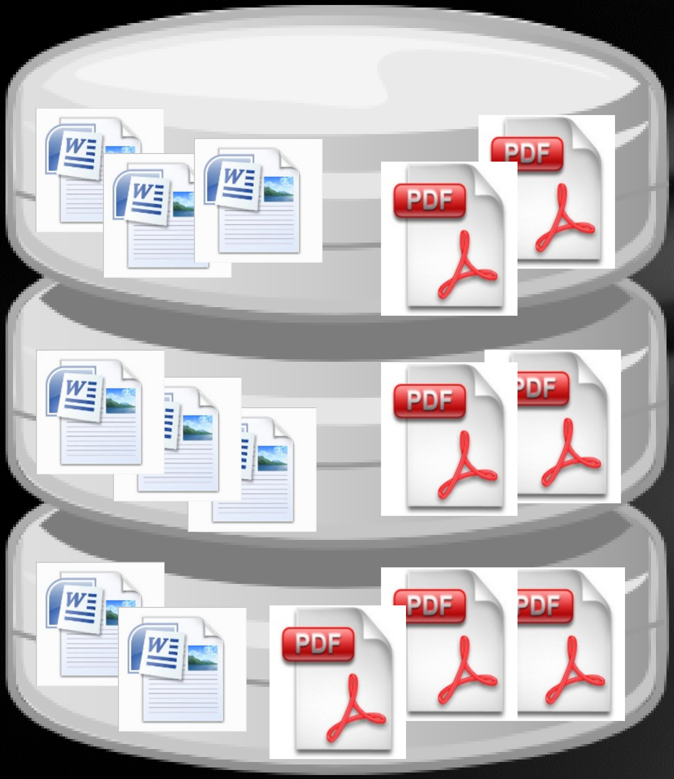
MongoDB

Document Oriented Database

What is Document Oriented Database?



What is Document Oriented Database?

Not Really...

What is Document Oriented Database?

- The central concept of a document-oriented database is the notion of a *Document*
- Documents encapsulate and encode data (or information) in some standard format(s):
  - JSON
  - XML
  - BSON
  - YAML
  - binary forms (like PDF and MS Word).
- Document is similar to row or record in relation DB, but more flexible.
- Retrieve documents based on their contents.
- Different DB offers variety ways to organize documents:
  - Collections
  - Tags
  - Non-visible Metadata
  - Directory hierarchies

MongoDB

"MongoDB wasn't designed in a lab. We built MongoDB from our own experiences building large scale, high availability, robust systems. We didn't start from scratch, we really tried to figure out what was broken, and tackle that. So the way I think about MongoDB is that if you take MySql, and change the data model from relational to document based, you get a lot of great features: embedded docs for speed, manageability, agile development with schema-less databases, easier horizontal scalability because joins aren't as important. There are lots of things that work great in relational databases: indexes, dynamic queries and updates to name a few, and we haven't changed much there. For example, the way you design your indexes in MongoDB should be exactly the way you do it in MySql or Oracle, you just have the option of indexing an embedded field."

MongoDB

"MongoDB wasn't designed in a lab. We built MongoDB from our own experiences building large scale, high availability, robust systems. We didn't start from scratch, we really tried to figure out what was broken, and tackle that. So the way I think about MongoDB is that if you take MySql, and change the data model from relational to document based, you get a lot of great features: embedded docs for speed, manageability, agile development with schema-less databases, easier horizontal scalability because joins aren't as important. There are lots of things that work great in relational databases: indexes, dynamic queries and updates to name a few, and we haven't changed much there. For example, the way you design your indexes in MongoDB should be exactly the way you do it in MySql or Oracle, you just have the option of indexing an embedded field."

"MongoDB wasn't designed in a lab. We built MongoDB from our own experiences building large scale, high availability, robust systems .... So the way I think about MongoDB is that if you take MySql, and change the data model from relational to document based, you get a lot of great features…"

– Eliot Horowitz, 10gen CTO and Co-founder

MongoDB Overview

- Schema-free document database
- Written in C++
- open-source project that mainly driven by 10gen Inc.
- 10gen Inc. also offers professional services around MongoDB.
- Has driver to all most every popular language programming.

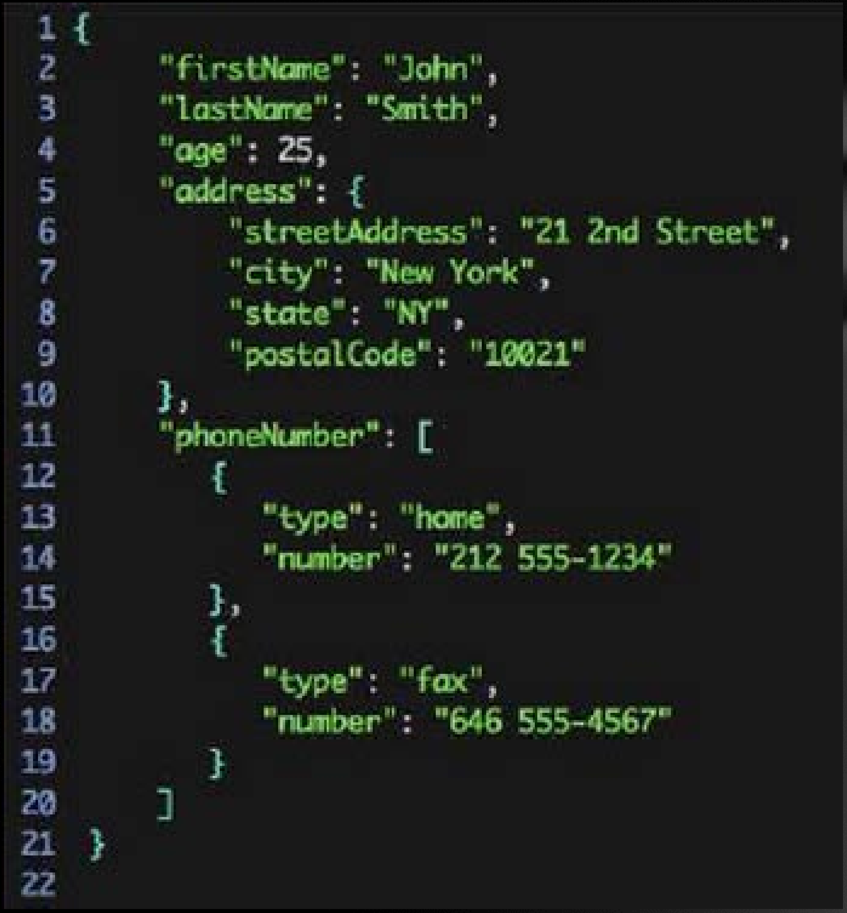
Why Choose MongoDB?

- Semi-structured Data
- Full Index Support
- Built-In Replication & Cluster Management
- Distributed Storage (Sharding)
- Easy to Query
- Fast In-Place Updates
- GridFS File Storage
- Capped collections

MongoDB in many ways "feels" like an RDMS. It's easy to learn and quick to implement.

Semi-structured Data

MongoDB is NOT a key/value store. Store complex documents as arrays, hash tables, integers, objects and every thing else supported by JSON:



```
{
        "firstName": "John",
        "lastName": "Smith",
        "age": 25,
        "address": {
            "streetAddress": "21 2nd Street",
            "city": "New York",
            "state": "NY",
            "postalCode": "10021"
        },
        "phoneNumber": [
            {
               "type": "home",
               "number": "212 555-1234"
            },
            {
               "type": "fax",
               "number": "646 555-4567"
            }
        ]
 }
```

Full Index Support

- An index is a data structure that collects information about the values of the specified fields in the documents of a collection.
- This data structure is used by Mongo's query optimizer to quickly sort through and order the documents in a collection
- Once a collection is indexed on a key, random access on query expressions which match the specified key are fast.
- Without the index, MongoDB has to go through each document checking the value of specified key in the query
- An index is automatically created for the _id field. This index is special and cannot be deleted (except capped collections)
- You can even index on a key inside of an embedded document

Full Index Support

- MongoDB also supports multi-key "compound" indexes.
- When a document's stored value for a index key field is an array, MongoDB indexes each element of the array
- MongoDB support "sparse index“. it is an index that only includes documents with the indexed field.
- MongoDB supports unique indexes, which guarantee that no documents are inserted whose values for the indexed keys match those of an existing document.
- Indexes make retrieval by a key, including ordered sequential retrieval, very fast. Updates by key are faster too as MongoDB can find the document to update very quickly.
- However, keep in mind that each index created adds a certain amount of overhead for inserts and deletes.

Built-In Replication & Cluster Management

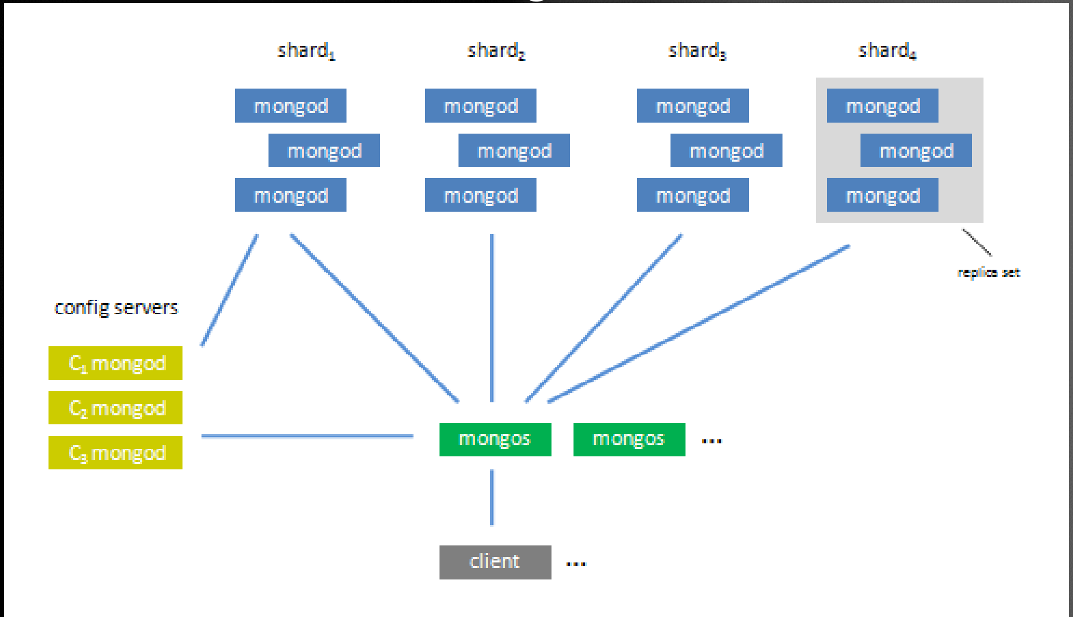
- Data redundancy
- Fault tolerant (automatic failover AND recovery)
- Consistency (wait-for-propagate or write-and-forget)
- Distribute read load
- Simplified maintenance
- Servers in the cluster managed by an elected leader

Distributed Storage (Sharding)

- Base on define shard key.
- It’s enabling horizontal scaling across multiple nodes.
- Application connects to the sharded cluster through a mongos process, which routes operations to the appropriate shard(s).
- Sharding is performed on a per-collection basis.
- Small collections need not be sharded.
- Auto-Balances as shard servers are added or removed
- Failover handled through replica sets.
- Map Reduce queries are run in parallel across shards.

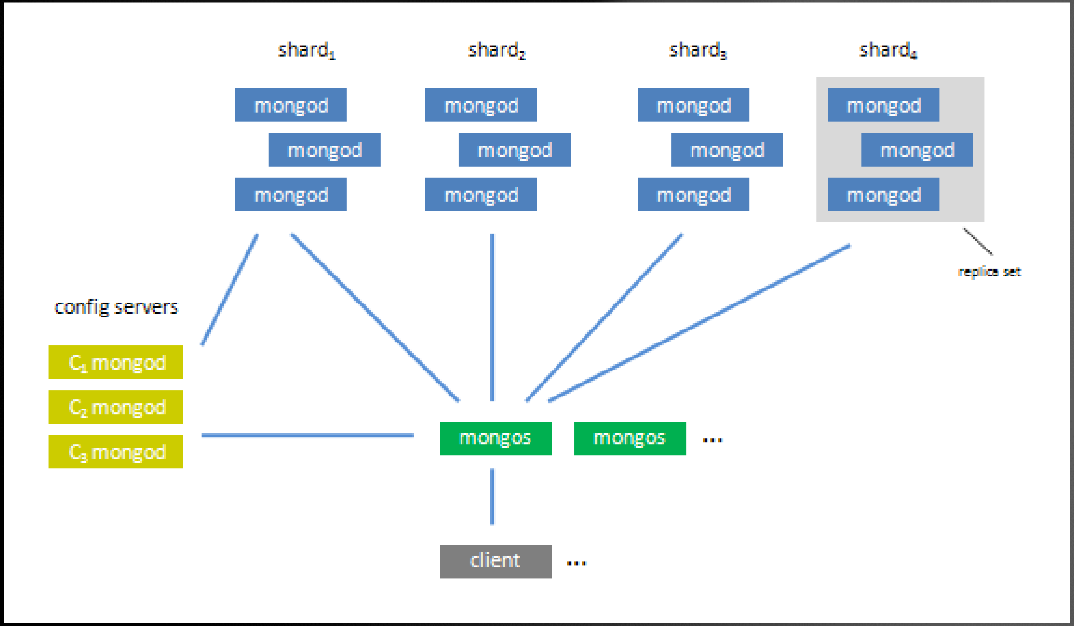
Simple Logical Architecture

1. One or more shards, each shard holds a portion of the total data (managed automatically). Reads and writes are automatically routed to the appropriate shard(s). Each shard is backed by a replica set – which just holds the data for that shard.
2. A replica set is one or more servers, each holding copies of the same data. At any given time one is primary and the rest are secondaries. If the primary goes down one of the secondaries takes over automatically as primary. All writes and consistent reads go to the primary, and all eventually consistent reads are distributed amongst all the secondaries.

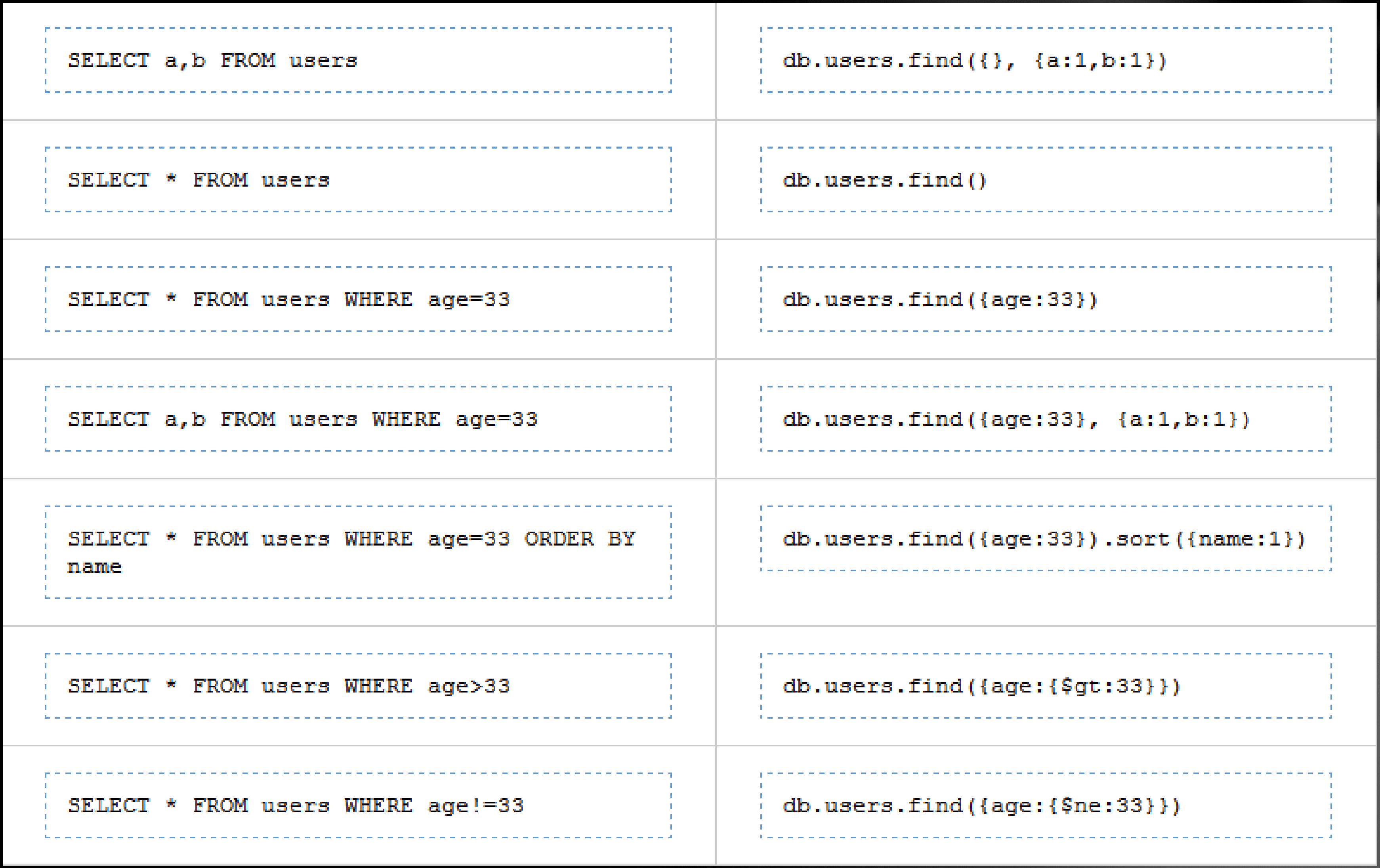


Simple Logical Architecture

3. Multiple config servers, each one holds a copy of the meta data indicating which data lives on which shard.
4. One or more routers, each one acts as a server for one or more clients. Clients issue queries/updates to a router and the router routes them to the appropriate shard while consulting the config servers.
5. One or more clients, each one is (part of) the user's application and issues commands to a router via the mongo client library (driver) for its language.



Easy to Query



`SELECT a,b FROM users`	`db.users.find({}, {a:1,b:1})`
`SELECT * FROM users`	`db.users.find()`
`SELECT * FROM users WHERE age=33`	`db.users.find({age:33})`
`SELECT a,b FROM users WHERE age=33`	`db.users.find({age:33}, {a:1,b:1})`
`SELECT * FROM users WHERE age=33 ORDER BY name`	`db.users.find({age:33}).sort({name:1})`
`SELECT * FROM users WHERE age>33`	`db.users.find({age:{$gt:33}})`
`SELECT * FROM users WHERE age!=33`	`db.users.find({age:{$ne:33}})`

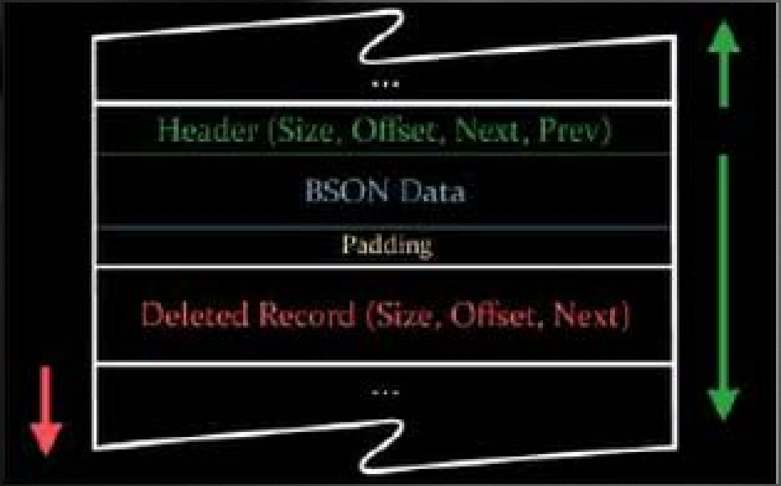
Fast In-Place Updates

MongoDB stores documents in padded memory slots. Typical RDMS updates on VARCHAR columns:

- Mark the row and index as deleted (without freeing the space)
- Append the new updated row
- Append the new index and possibly rebuild the tree

Most updates are small and don't drastically change the size of the row:

- Last log in date
- UUID replace/Password update
- Session cookie
- Counters (failed log in attempts, visits)


...
Header (Size, Offset, Next, Prev)
BSON Data
Padding
Deleted Record (Size, Offset, Next)
...

MongoDB can apply most updates over the existing row, keeping the index and data structure relatively untouched – and do so VERY FAST.

GridFS File Storage

Efficiently store binary files in MongoDB:

- Videos
- Pictures
- Translations
- Configuration files

Data is distributed in 4 or 16MB chunk sand stored redundantly in your MongoDB network.

- No serialization / fast reads
- Command line and PHP extension access

Capped collections

Fixed-size round robin tables with extremely fast reads and writes.

Perfect for:

- Logging
- Messaging
- Job Queues
- Caching

Features:

- Automatically “ages out” old data
- Canal so query, delete and update out of FIFO order
- FIFO reads/writes are nearly as fast as cat> file; tail–f/file
- Tailable cursor stays open as reads rows as they are added
- Persistent, fault-tolerant, distributed
- Atomic pop items off the stack

Getting Started

- You can install MondoDB in two main ways:
  - ❑Run it like regular exe program.
  - ❑Install it like service in your OS.

Install MongoDB

1. Download MongoDB. www.mongodb.com/downloads
2. Extract it.
3. Create the data folder. Usually /data/db –or- C:\data\db
4. Run mongod.exe

That's it! So Simple!

Install MongoDB as Service

MongoDB can also be installed as a service on Windows.

To do this we just need to run the next command from the CMD:

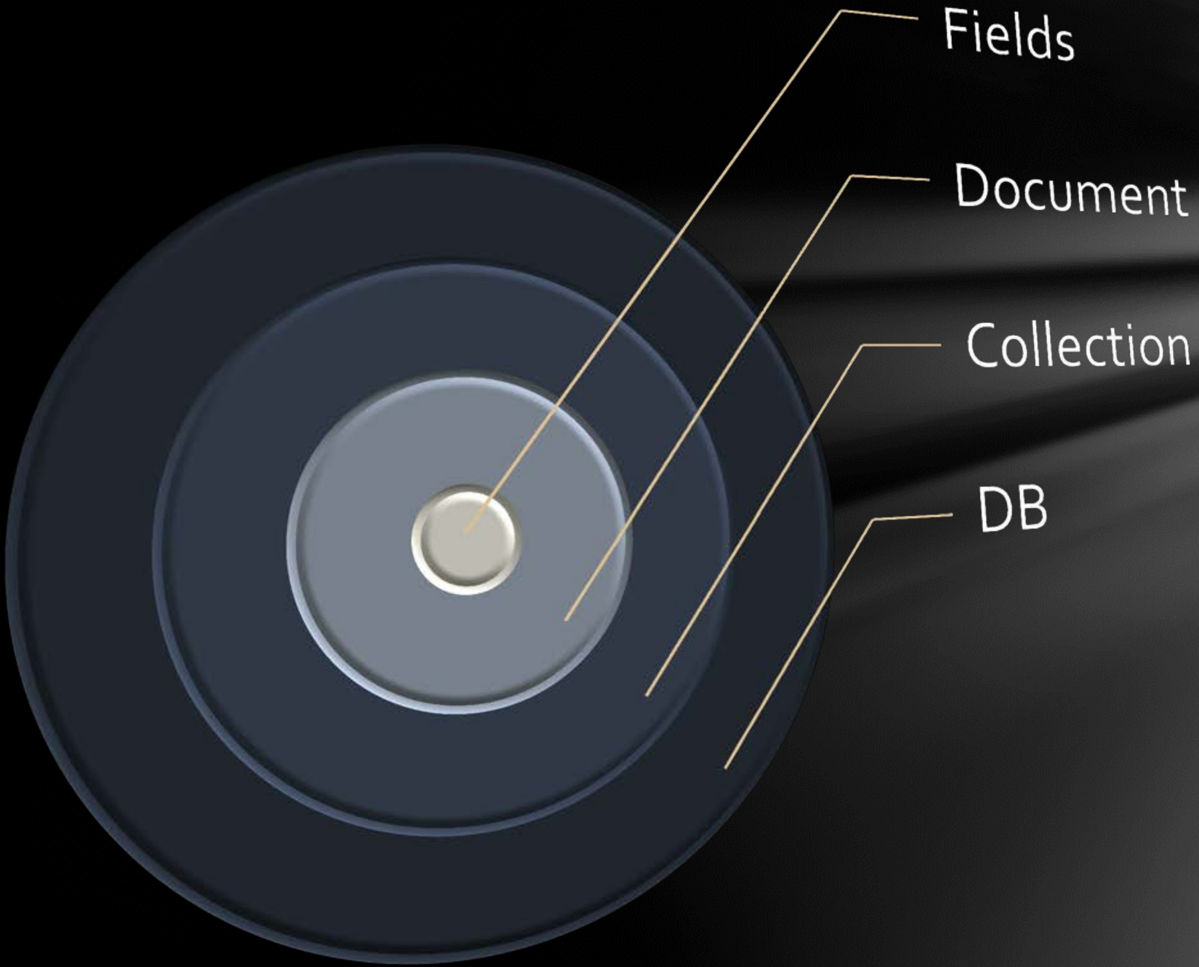
```
C:\mongodb-windows-32bit-1.6.0\bin\mongod.exe

--dbpath "\"C:\Documents and Settings\Username\My Documents\db\"" --
install
```

Using the shell

- MongoDB comes with a JavaScript shell.
- The shell is very useful for performing administrative functions, inspecting a running instance, or just playing around
- The shell is also a stand-alone MongoDB client.
- Each operation that we can do with the shell we can do with the Java Driver and vice versa.
- To start the shell you just need to run the mongo.exe

MongoDB Data Model



Fields Data Types

- Object Ids
- Regular Expressions
- Dates/Times
- Database References
- Embedded Documents
- Arrays
- Primitive Data Types

Implementation

DB Connction

```
Mongo m = new Mongo();
// or
Mongo m = new Mongo( "localhost" );
// or
Mongo m = new Mongo( "localhost" , 27017 );

//getting DB
DB db = m.getDB( "mydb" );
```

Authentication

When you create a connection to the DB you can use authentication and actually use this DB as the most popular RDMS (like Oracle and MSSQL).

Authentication is optional.

```
//example:
boolean auth = db.authenticate(myUserName, myPassword);
```

Collections

- Each DB in Mongo DB composed of collections.
- Each Collection contains documents (recommended with the same structure).
- Once you have this collection object, you can do things like insert data, query for data, etc.

```
//getting  a collction
DBCollection coll = db.getCollection("testCollection");
```

Primary Key

- Almost every MongoDB document has an _id field as its first attribute.
- The _id value is the primary key in MongoDB, it can be of any type except arrays.
- If a user tries to insert a document without providing an _id field, the database will automatically generate an _object id_ and store it the _id field.
- All the MongoDB Drivers use ObjectId as the default _id value.

Insert A Document

```
//Creating a document
BasicDBObject doc = new BasicDBObject();
 doc.put("name", "MongoDB");
 doc.put("type", "database");
 doc.put("count", 1);

//Creating embedded document
BasicDBObject info = new BasicDBObject();
info.put("x", 203);
info.put("y", 102);
doc.put("info", info);

//Insert to the collection
coll.insert(doc);
```

Update A Document

```
//update single document
basicDBObject query = new basicDBObject ( "_id" , order_id);
basicDBObject fildesToUpdate = new basicDBObject ( "state" , "cart" );
Coll.update(query, fildesToUpdate);

//update multiple documents.
basicDBObject query = QueryBuilder.start( "subtotal" ).graterthen(2500);
basicDBObject fildesToUpdate = new basicDBObject ( "state" , "cart" );
Coll.update(query, fildesToUpdate, false, true);
```

Remove A Document

Removing document is very simple. You just need to use the remove method

```
//example
//remove all the objects in the collection.
coll( "name" ).remove();
//remove specific object from the collection
coll( "name" ).remove(query);
```

Finding A Document

The basic way to get document from collection is:

```
//findone()
DBObject myDoc = coll.findOne();
//or find()
//return all the documents in the collection
DBCursor cur = coll.find();
//return the documents that much to the query
cur = coll.find(query);
```

DBCursor

- The find function return class type of DBCursor.
- The DBCursor is an iterator to the query result list.
- You can use the DBCursor as a regular java iterator and run all over the results in very simple way.

```
//Example
DBCursor cur = coll.find();
while (cur.hasNext())
{
		System.out.println(cur.next());
}
```

Getting A Document with a query

Until now we talk about how to get result from the DB, but how we can filter them?

MondoDB let you build more rich queries comparatively to other NoSQL DB.

The class QueryBuilder use us to build more complex queries.

QueryBuilder Example

```
//Get all the capital cities with 10000000 or more citizens
DBObject o;
o = QueryBuilder.
        start("population").greaterThan(10000000).and("capital").is(true)
        .get();
DBCursor answer = db.getCollection("Cities").find(o);
```

Query Embedded Document

As recommended earlier you can store embedded document.

It can be possible to query be the content of the embedded document.

```
//example
 DBObject query =
              QueryBuilder.start("coordinates.latitudeType").is("S").and
              ("coordinates.longitudeType").is("W").get();


 DBCursor answer = db.getCollection("Countries").find(query);
```

Aggregate

Choosing MongoDB meant giving up on SQL's aggregate functions (AVG(), MAX(), MIN(), SUM(), etc.).

You need to implement this function in your client side.

MongoDB give you just one aggregation function called count() that retrieve the number of documents that match to some query.

```
// Aggregate example

Coll.count(query);
```

Group By

```
BasicDBObject key = new BasicDBObject();
BasicDBObject initial = new BasicDBObject();
initial.put("totalPopulation", 0);
DBObject cond1 = QueryBuilder.start("_id").notEquals("ee").get();
String reduceString = "function(obj,prev)
            { prev.totalPopulation += obj.population; }";
BasicDBList cur = (BasicDBList) db.getCollection("Countries")
                        .group(key, cond1, initial, reduceString);
```

Order By

- To do order by in mongoDB you need to use the sort() method.
- The method get an ObjectDB with the fields to order by and -1 to order descending or 1 to order ascending.

```
//order by example
db.getCollection("Cities").find(o).sort(new basicDBObject(name, -1));
```

Indexes

```
//example of creating index
 BasicDBObject index = new BasicDBObject("country1", -1);
        index.put("country2", -1);
        index.put("lenght", 1);
        db.getCollection("Borders").ensureIndex(index, "borderNieg", true);
```

Conclusion

Pros

- Scalable, high performance database with familiar RDMS functionality
- Semi-structured (hash tables, lists, dates, ...)
- Full, range and nested Indexes
- Replication and distributed storage
- Query language and Map/Reduce
- GridFS file storage (NFS replacement)
- BSON Serialization
- Capped Collections
- A lot of drivers to program language.
- Simple to install

Cons

- Map/Reduce is single process (soon to be resolved)
- Low Availability
- Give up on aggregation function like SUM(), MAX(), MIN(), etc.
- Complex syntax for group by function

Comparison To Similar DB

CouchDB

Best used: For accumulating, occasionally changing data, on which pre-defined queries are to be run. Places where versioning is important.

For example: CRM, CMS systems. Master-master replication is an especially interesting feature, allowing easy multi-site deployments.

MongoDB

Best used: If you need dynamic queries. If you prefer to define indexes, not map/reduce functions. If you need good performance on a big DB. If you wanted CouchDB, but your data changes too much, filling up disks.

For example: For most things that you would do with MySQL or PostgreSQL, but having predefined columns really holds you back.

Redis

Best used: For rapidly changing data with a foreseeable database size (should fit mostly in memory).

For example: Stock prices. Analytics. Real-time data collection. Real-time communication.

Cassandra

Best used: When you write more than you read (logging). If every component of the system must be in Java.

For example: Banking, financial industry (though not necessarily for financial transactions, but these industries are much bigger than that.) Writes are faster than reads, so one natural niche is real time data analysis.

Common Usage

Content Management System	MongoDB, CouchDB
Real-time Analytics	MongoDB, Cassandra
Page/Query Cache	Redis, Voldemort
Logging and Archiving Event	MongoDB, Cassandra
Messaging	Redis, Cassandra
Job Queue	MongoDB, Redis
Social Networking	Neo4J, Cassandra
Data Mining & Warehousing	Hbase, InfiniDB
Binary Storage (Files, Images,…)	MongoDB
Sessions	Redis, MongoDB

MongoDB Sample Users

Company	Implementation
Disney	Using MongoDB as a common object repository to persist state information. Disney Central Services Storage: Leveraging Knowledge and skillsets - MongoSF
SAP	Uses MongoDB as a core component of SAP's platform-as-a-service (PaaS) offering.
The New York Times	Using MongoDB in a form-building application for photo submissions. Mongo's dynamic schema gives producers the ability to define any combination of custom form fields.
Open Dining Network	Open Dining Network is a restaurant data and food ordering platform that provides a Restful API to take web and mobile orders. MongoDB is used to manage all restaurant, customer, and order information

Bibliography

- *NoSQL Databases by Christof Strauch*
- *MongoDB – the definition guide by Kristina Chodorow & Michael Dirolf*
- MongoDB Official Site - *www.mongodb.com*
- *http://www.slideshare.net*
- *Compare Sites:*
  - *http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis*
  - *http://nosql.mypopescu.com/post/298557551/couchdb-vs-mongodb*

Questions…

Thank you for listening