Mongo Database

The Right database to fit your data

{“author”:”kirankumar”}

The Right database to fit your Development

The Right database for Scale and Performance

WHAT DO WE LOOK FOR IN A DATABASE?

- Right structure to match my data
- Performance & Scale
- Features that enable me as a developer

NO SQL

{“author”:”kirankumar”}

NoSQL

Defining Characteristics

- Scaling out on commodity hardware
- Aggregate structure
- Schema-less attitude
- Impedance Mismatch : Relational model in-memory data structures
- Big Data : Massive data being stored and transacted
- Reduced Data Management and Tuning Requirements
- Eventually consistent / BASE (not ACID)

Understanding NO SQL

NoSQL databases are little different than conventional databases (in terms of storage, reterival, performance, scalability, security, accuracy. They are widely used in web applications (like storing website cache so that results can be fetched on search, for example google).

Conventional databases involves storing rows (tables) and then multiple tables might be joined to fetch results for required query. In NoSQL solution data might be stored based on columns or key/value pair.

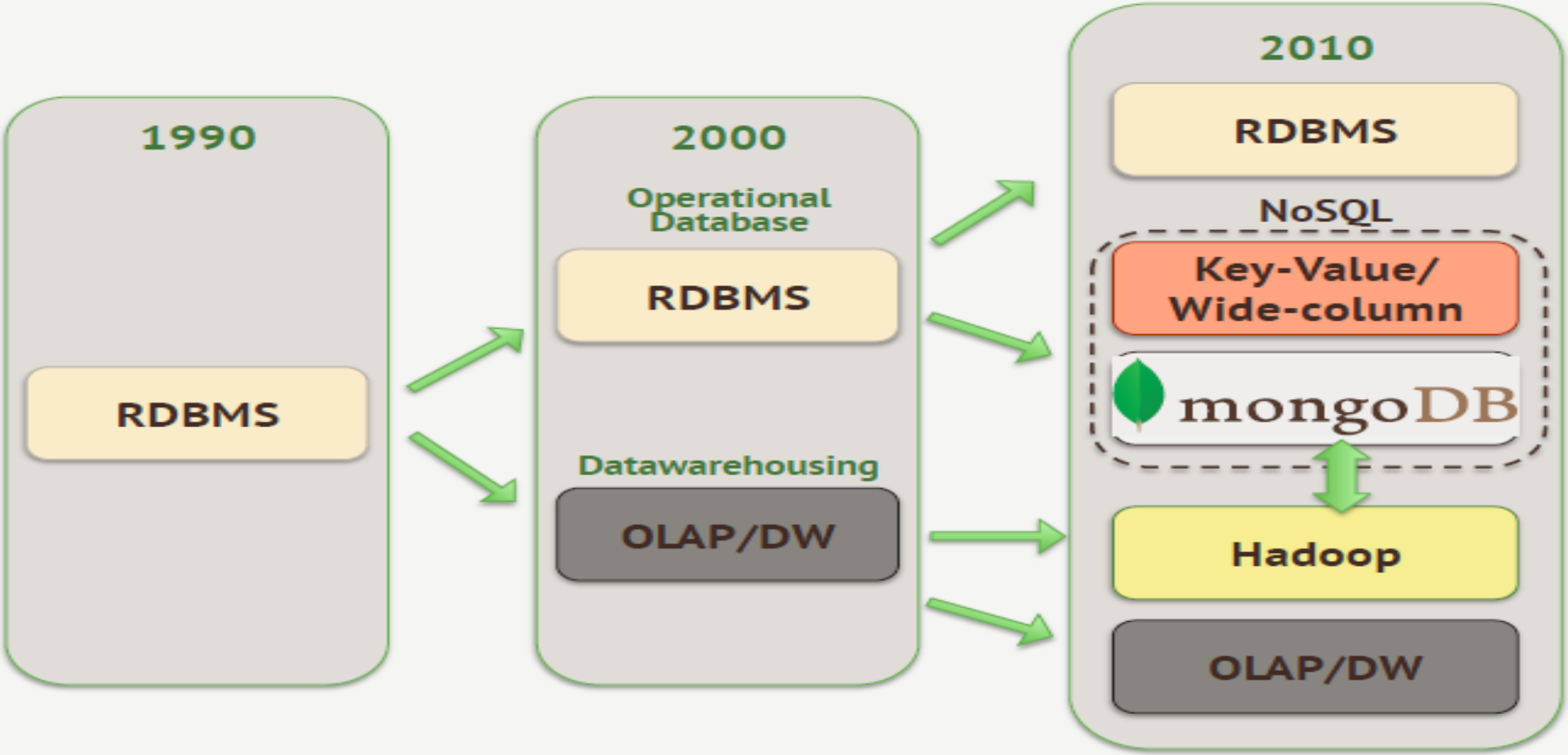
Performance is better in NoSQL databases (again depends how it's used and setup). To improve performance of conventional databases hardware or database optimization will be required.

Open source

- Mongo DB is an open source project.
- On Github
- Licensed under the AGPL
- Started and Sponsored by Mongo DB Inc(Formely Know as 10gen)

Mongo DB Evolution

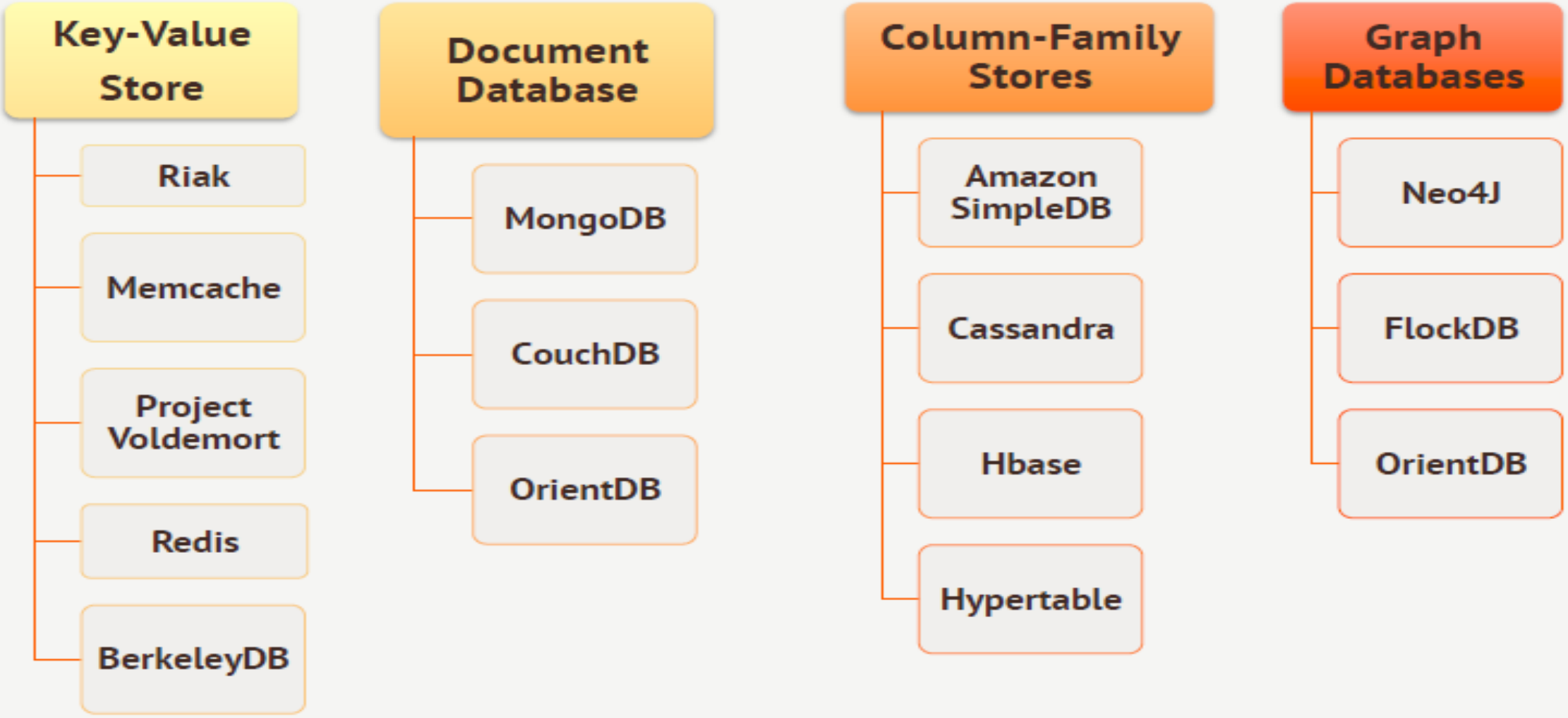
mongoDB
{ name: mongo, type: DB }



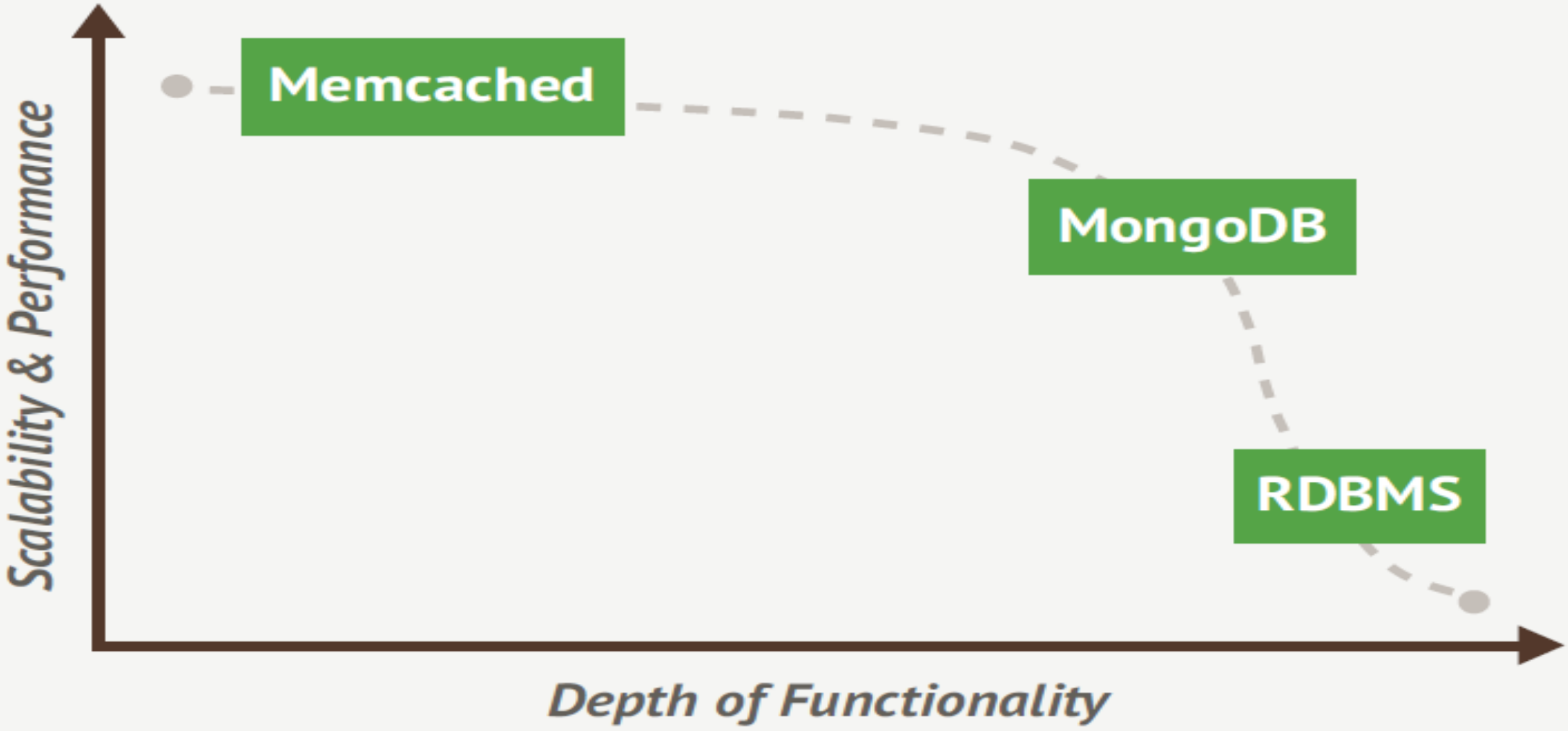
History of the Database

- 1960's – Hierarchical and Network type (IMS and CODASYL)
- 1970's – Beginnings of theory behind relational model. Codd
- 1980's – Rise of the relational model. SQL. E/R Model (Chen)
- 1990's – Access/Excel and MySQL. ODMS began to appear
- 2000;'s – Two forces; large enterprise and open source. Google and Amazon. CAP Theorem (more on that to come…)
- 2010's – Immergence of NoSQL as an industry player and viable alternative

1. NoSQL Data Model



Database Landscape



Why MongoDB

- Intrinsic support for agile development
- Super low latency access to your data
  - Very little CPU overhead
- No Additional caching layer required
- Built in Replication and Horizontal Scaling support

Overview on Mongo DB

- A ***document*** is the basic unit of data for MongoDB, roughly equivalent to a row in arelational database management system (but much more expressive).

- Similarly, a ***collection*** can be thought of as the schema-free equivalent of a table.

- A single instance of MongoDB can host multiple independent ***databases***, each of which can have its own collections and permissions.

- **MongoDB** comes with a simple but powerful JavaScript ***shell***, which is useful for the administration of MongoDB instances and data manipulation.

10gen is the company behind MongoDB.

Founded in 2007

- Dwight Merriman, Eliot Horowitz
- Doubleclick, Oracle, Marklogic, HP

$73M+ in Funding

- Flybridge, Sequoia, NEA, Union Square

Worldwide Expanding Team

- 140+ employees
- NY, Palo Alto, London, Dublin, Sydney

Set the direction & contribute code to MongoDB

Foster community & ecosystem

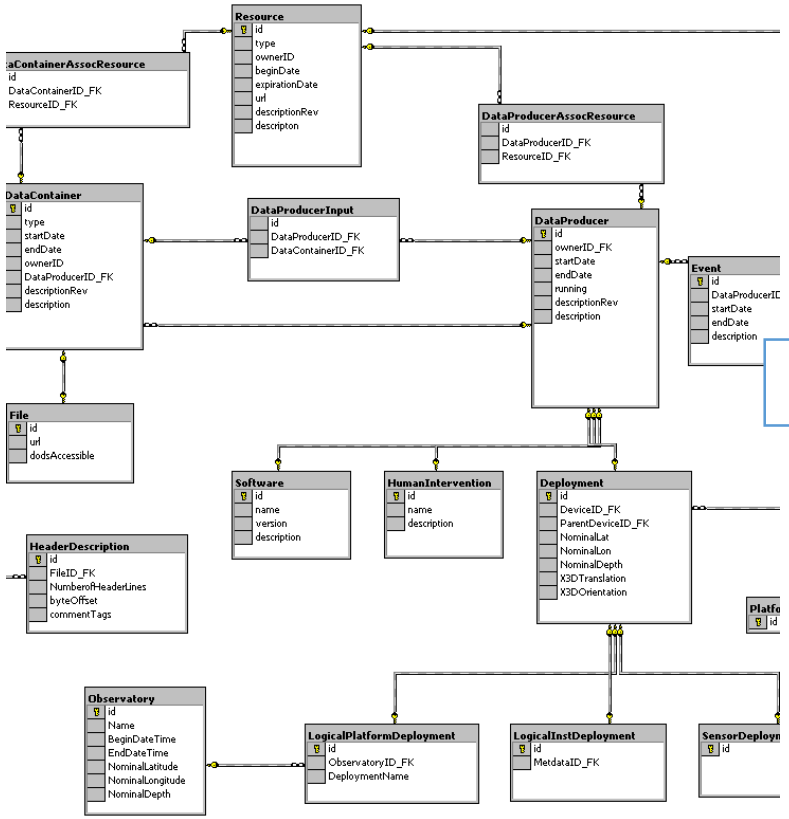
Provide commercial services

Provide MongoDB management services

Terminology

RDBMS	Mongo
Table, View	Collection
Row(s)	JSON Document
Index	Index
Join	Embedded Document
Partition	Shard
Partition Key	Shard Key

MongoDB is Easy to Use




```
{
  title: ‘MongoDB’,
  contributors: [
    { name: ‘Eliot Horowitz’,
      email: ‘eliot@10gen.com’ },
    { name: ‘Dwight Merriman’,
      email: ‘dwight@10gen.com’ }
  ],
  model: {
    relational: false,
    awesome: true
  }
}
```

High Performance

- Written in C++

- Extensive use of memory-mapped files

i.e. read-through write-through memory caching.

- Runs nearly everywhere

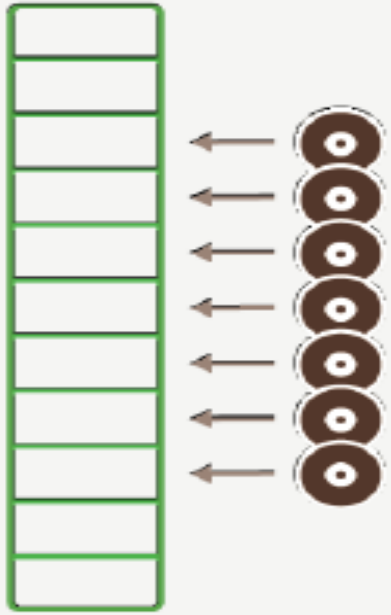
- Data serialized as BSON (fast parsing)

- Full support for primary & secondary indexes

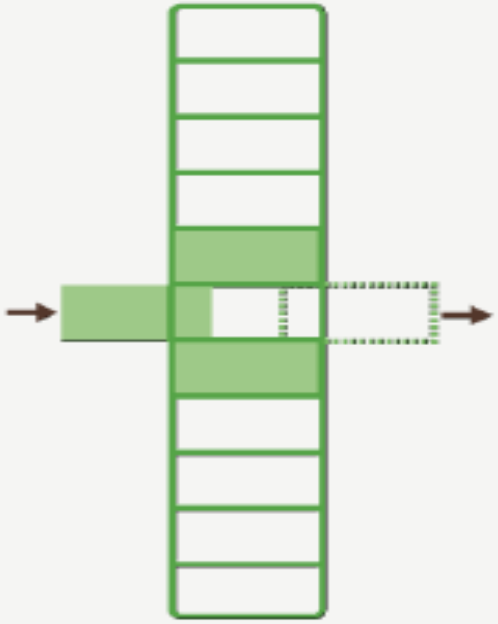
- Document model = less work

Performance

Better Data Locality

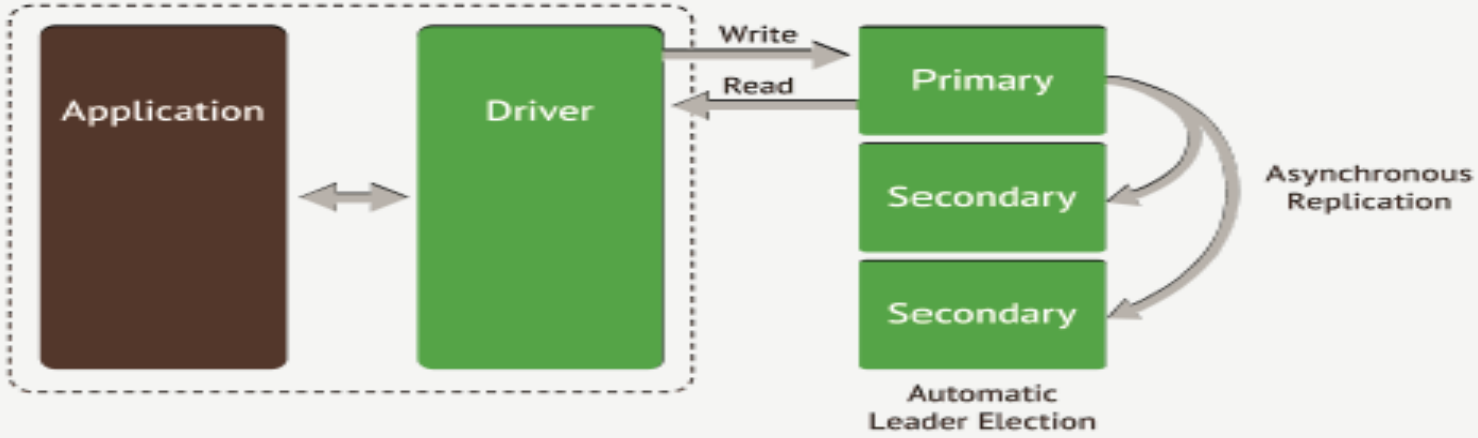


In-Memory Caching



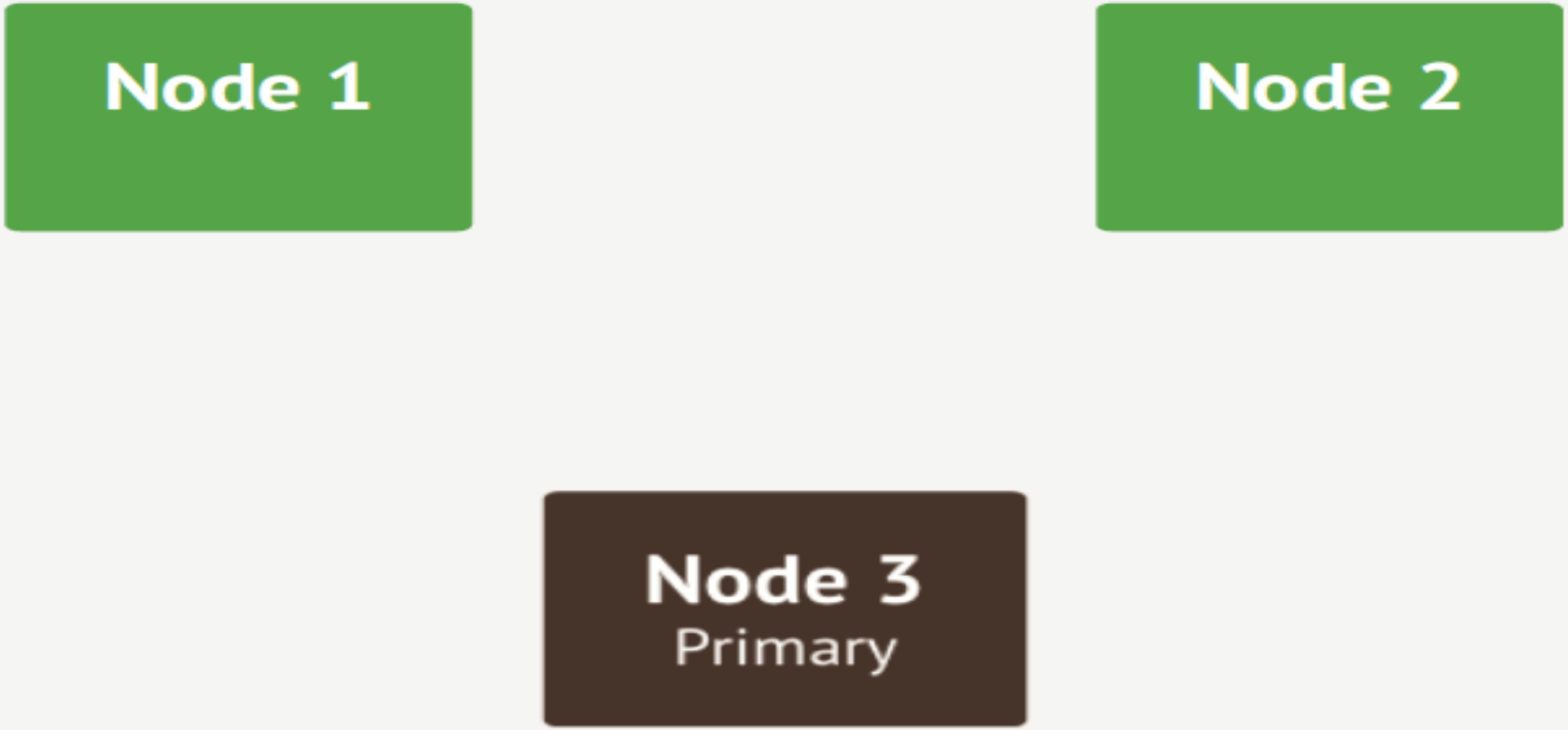
In-Place Updates

High Availability

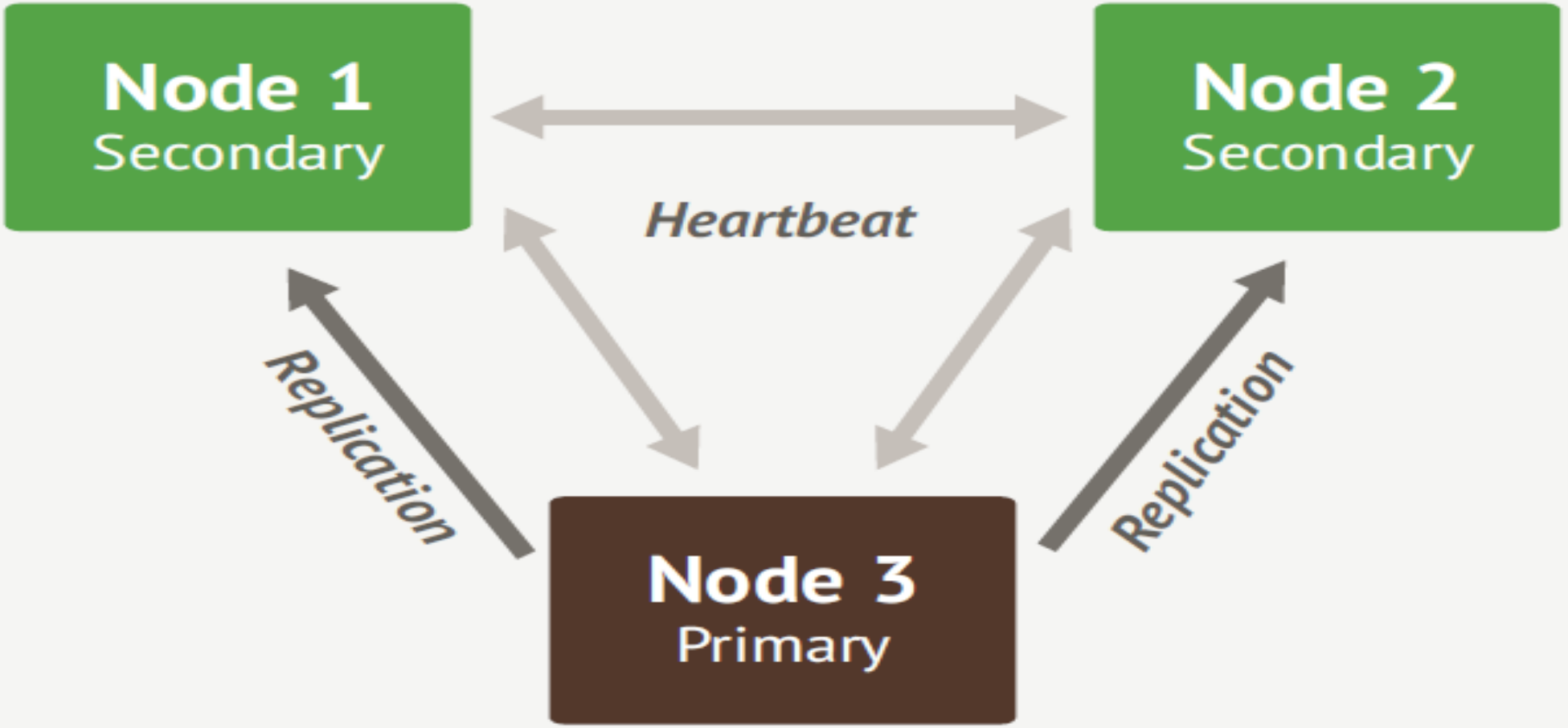


- Automated replication and failover
- Multi-data center support
- Improved operational simplicity (e.g., HW swaps)
- Data durability and consistency

Communication

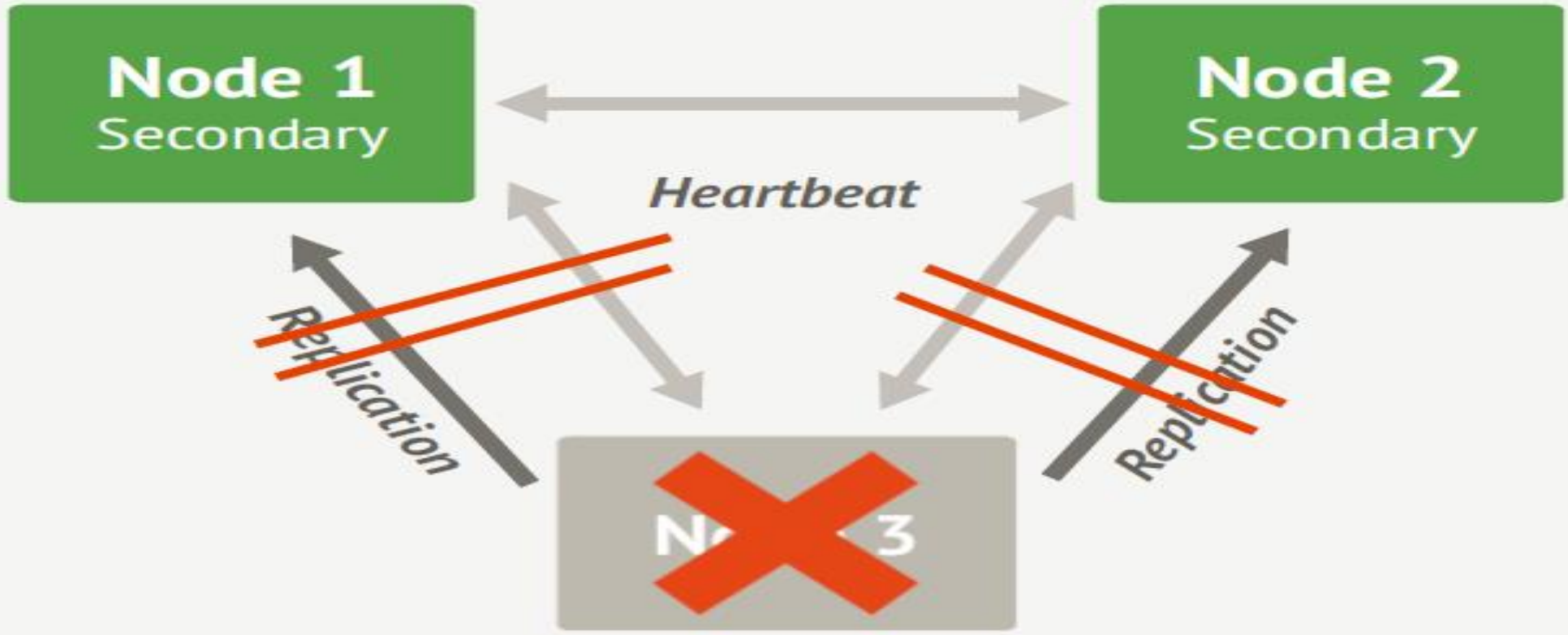


Communication



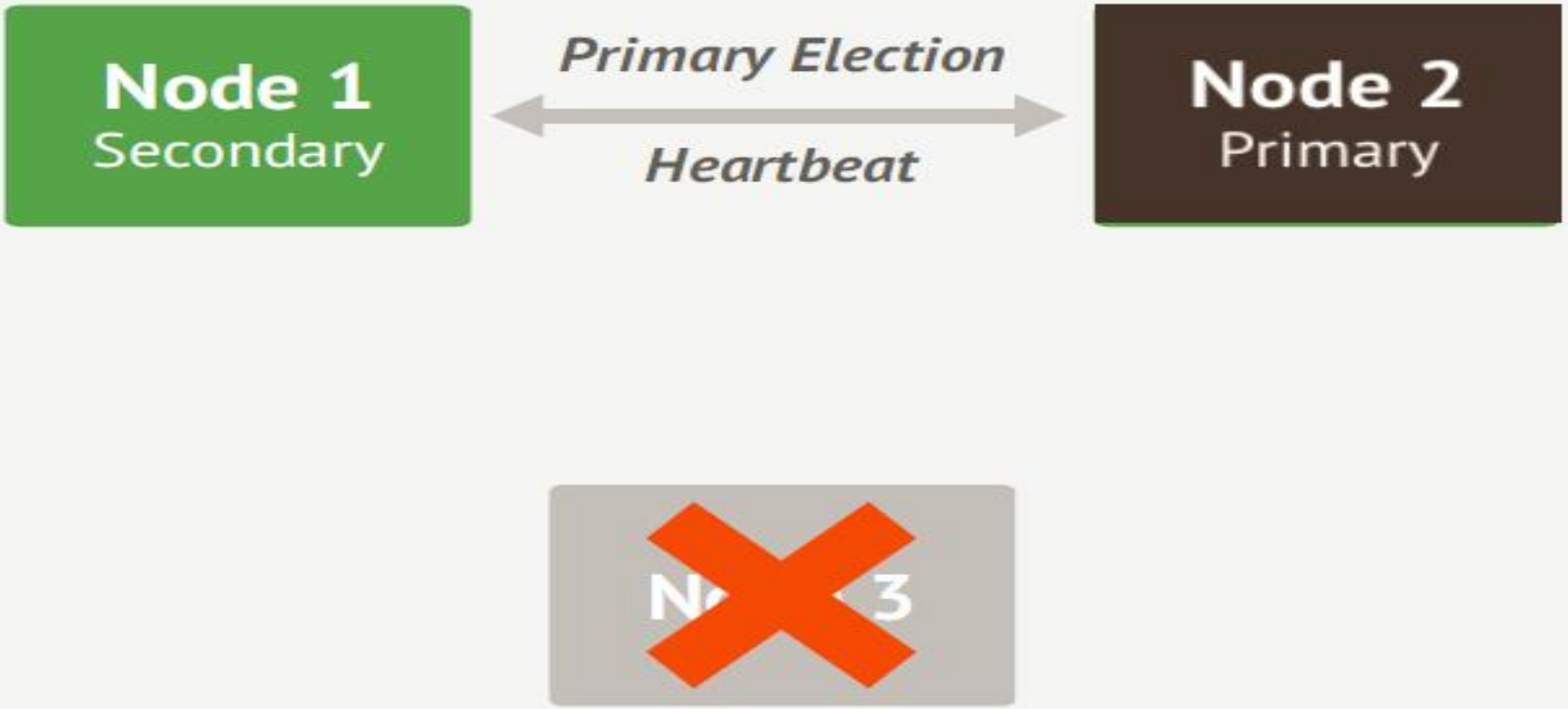
Replica Set – Communication

Replica set communication breaks down



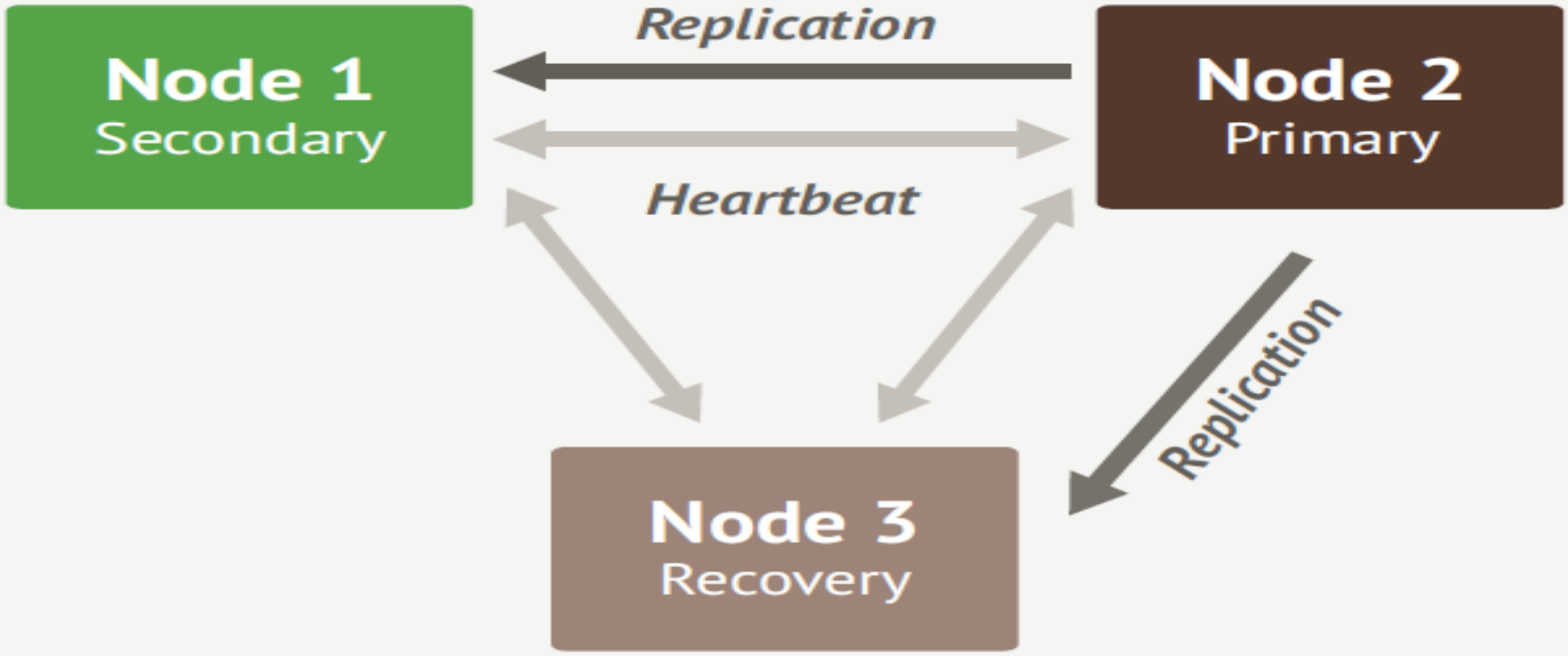
Replica set -Failure

Primary Election



Replica Set – Failure

Replica set -Recovery



Replica Set – Recovery

Document

- *At its simplest form, Mongo is a document oriented database*

- MongoDB stores all data in documents, which are JSON-style data structures composed of field-and-value pairs.

- MongoDB stores documents on disk in the BSON serialization format. BSON is a binary representation of JSON documents. BSON contains more data types than does JSON.

- ** For in-depth BSON information, see bsonspec.org.

What does a Document Look Like

```
{
 "_id" : "52a602280f2e642811ce8478",
 "ratingCode" : "PG13",
 "country" : "USA",
 "entityType" : "Rating”
}
```

Rules for a document

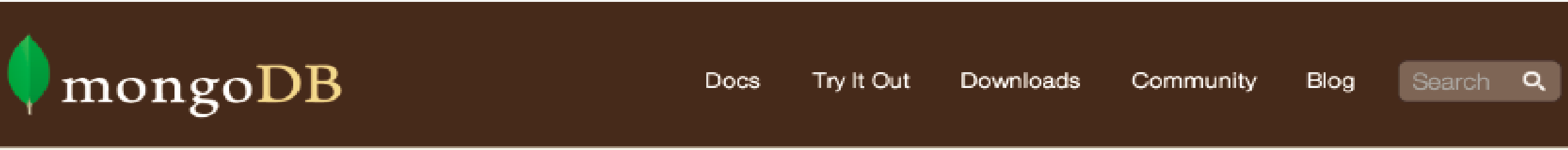
- Documents have the following rules:

- The maximum BSON document size is 16 megabytes.
- The field name _id is reserved for use as a primary key; its value must be unique in the collection.
- The field names cannot start with the $ character.
- The field names cannot contain the . character.

BSON

- JSON has powerful, but limited set of datatypes
  - Mongo extends datypes with Date, Int types, Id, …
- MongoDB stores data in BSON
- BSON is a binary representation of JSON
  - Optimized for performance and navigational abilities
  - Also compression
  - See bsonspec.org

mongodb.org/downloads



MongoDB Downloads

This table lists MongoDB distributions by platform and version. We recommend using these binary distributions, but there are also packages available for various package managers.

Production Release (2.6.0) — 4/8/2014

Release Notes, Changelog, Source: tgz | zip

Windows	Linux	Mac OS X	Solaris
64-bit	64-bit	64-bit	64-bit
Download	Download	Download	Download
64-bit zip | msi 32-bit zip | msi 64-bit legacy zip | msi	32-bit		

Mongo Install

- Windows
- http://docs.mongodb.org/manual/tutorial/install-mongodb-on-windows/

- MAC
- http://docs.mongodb.org/manual/tutorial/install-mongodb-on-os-x/

- Create Data Directory , Defaults
  - C:\data\db
  - /data/db/ (make sure have permissions)

- Or can set using -dbpath
- C:\mongodb\bin\mongod.exe --dbpath d:\test\mongodb\data

Start IT!

- Database
- mongod
-

- Shell
- mongo
- show dbs
- show collections
- db.stats()

Basic Operations

- 1_simpleinsert.txt
    - Insert
    - Find
        - Find all
        - Find One
        - Find with criteria
    - Indexes
        - Explain()

More Mongo Shell

- 2_arrays_sort.txt
    - Embedded documents
    - Limit, Sort
    - Using regex in query
    - Removing documents
    - Drop collection

Import / Export

- 3_imp_exp.txt
- Mongo provides tools for getting data in and out of the database
  - Data Can Be Exported to json files
  - Json files can then be Imported

Import / Export

- 3_imp_exp.txt
- Mongo provides tools for getting data in and out of the database
  - Data Can Be Exported to json files
  - Json files can then be Imported

Conditional Operators

- 4_cond_ops.txt
    - $lt
    - $gt
    - $gte
    - $lte
    - $or
    - Also $not, $exists, $type, $in
- (for $type refer to http://docs.mongodb.org/manual/reference/operator/query/type/#_S_type)

Admin Commands

- 5_admin.txt
    - how dbs
    - show collections
    - db.stats()
    - db.posts.stats()
    - db.posts.drop()
    - db.system.indexes.find()

Data MODELING

- Remember with NoSql redundancy is not evil
- Applications insure consistency, not the DB
- Application join data, not defined in the DB
- Datamodel is schema-less
- Datamodel is built to support queries usually

Questions to ask

- Your basic units of data (what would be a document)?
- How are these units grouped / related?
- How does Mongo let you query this data, what are the options?
- Finally, maybe most importantly, what are your applications access patterns?
  - Reads vs. writes
  - Queries
  - Updates
  - Deletions
  - How structured is it

Data Model - Normalized

- Normalized
  - Similar to relational model.
  - One collection per entity type
  - Little or no redundancy
  - Allows clean updates, familiar to many SQL users, easier to understand

Other considerations For Data Modeling

- Many or few collections
- Many Collections
    - As seen in normalized
    - Clean and little redundancy
    - May not provide best performance
    - May require frequent updates to application if new types added
- Multiple Collections
    - Middle ground, partially normalized
- Not many collections
    - One large generic collection
    - Contains many types
    - Use type field

Consideration Continued

- Document Growth – will relocate if exceeds allocated size
- Atomicity
  - Atomic at document level
  - Consideration for insertions, remove and multi-document updates
- Sharding – collections distributed across mongod instances, uses a shard key
- Indexes – index fields often queries, indexes affect write performance slightly
- Consider using TTL to automatically expire documents

Integrated Search

Text Search

- Now production-ready
- Integrated with query engine
- Integrated with aggregation framework
- Multi-language document support

Features

- Language-specific tokenization and stemming
- Stop words
- Relevance ranking
- Field boosting

Supported Languages

da – Danish	en – English
nl – Dutch	fi – Finish
fr – French	de – German
hu – Hungarian	it – Italian
no – Norwegian	pt – Portuguese
ro – Romanian	ru – Russian
es – Spanish	sv – Swedish
tr - Turkish	

Integrated within find()

```
db.articles.ensureIndex( { body: "text" } );

db.articles.insert(
    { body: "the quick brown fox jumped over the lazy dog" }
);

db.articles.find( { "$text" : { $search : "quickly" } } );
```

More Text Examples

```
// search for a single word
db.articles.find( { "$text": { "$search": "coffee" } } );

// search for any of these words
db.articles.find( { "$text": { "$search": "bake coffee cake" } } );

// search for a phrase
db.articles.find( { "$text": { "$search": "\"coffee cake\"" } } );

// excluding some terms
db.articles.find( { "$text": { "$search": "bake coffee -cake" } } );
```

Relevance

```
db.articles.find(
   { "$text": { "$search": "cake" } },
   { "score": { "$meta": "textScore" } }
).sort( { "score": { "$meta": "textScore" } } ).limit(3)
```

Forcing a Language

```
db.articles.find(
   { "$text": { "$search": "leche", $language: "es" } }
);
```

Query System Improvements

Query System Improvements

- Index Intersection
- Aggregation Framework
- New Update Operators

Index Intersection

- Simpler ad-hoc queries
- Existing indexes can be combined to optimize a query
  - Less Index Maintenance
  - Smaller Working Set
  - Lower Write Overhead
  - More Adaptive
  - Able to control potential intersections using QueryShape

Index Intersection

```
// index creation
db.beers.ensureIndex( { barrels: 1 } );
db.beers.ensureIndex( { alcohol: 1 } );

// retrieval
db.beers.find( { barrels: { "$gte": 100 }, alcohol: { "$gte": 5.5 } } );
```

New Update Operators

- $mul
- $min/$max
- $bit
- $currentDate
- New modifiers for $push

Security

Security

- Authentication with LDAP (Enterprise only)
- x.509 Certificates
- User defined roles
- Collection level security
- Auditing (Enterprise only)
- Windows Kerberos Support

State of Security in MongoDB 2.6

- Authentication
  - Who are you?
  - X.509 authentication and Kerberos
- Authorization
  - What can you do?
  - User Defined Roles, Collection-level Access Control
- Auditing
  - What have you done?
  - DDL, User Manipulation, Authorization failure

Roles and Collection-level Security

```
db.createRole({
  "role": "appUser",
  "db": "myApp",
  "privileges": [
        {
            "resource": { "db": "myApp" , "collection": "" },
            "actions": [ "find", "dbStats", "collStats" ]
        },
        {
            "resource": { "db": "myApp", "collection": "beers" },
            "actions": [ "insert"]
        }
  ]
};
```

Auditing

- Schema actions
- Replica Set actions
- Authentication & Authorization actions
- Other actions

Auditing – Dropping a collection

```
var auditEntry = {
    "atype" : "dropCollection",
    "ts" : { "$date" : "2014-04-08T16:48:34.333+1000" },
    "local" : { "ip" : "127.0.0.1", "port" : 27017 },
    "remote" : { "ip" : "127.0.0.1", "port" : 55771 },
    "users" : [
        { "user": "matias", "db": "test" }
    ],
    "param" : { "ns" : "test.beers" },
    "result" : 0
};
```

Auditing – Shutting down the server

```
var auditEntry = {
    "atype" : "shutdown",
    "ts" : { "$date" : "2014-04-08T16:54:24.373+1000" },
    "local" : { "ip" : "127.0.0.1", "port" : 27017 },
    "remote" : { "ip" : "127.0.0.1", "port" : 55771 },
    "users" : [
        { "user": "matias", "db": "admin" }
    ],
    "param" : {},
    "result" : 0
};
```

Mongo DB Pros

- **Cost**

It's free, open source. Can has more scale? Just add hardware. Licensing costs need not apply (can run on Linux)
.

- **Schema-less**

If you need to support a flexible schema, MongoDB's document storage is a big plus. It doesn't mean you don't need to think about schema at all, it just means you can very easily model data of this nature and cope with changes without headaches.

- **Quick start & fast learning**

Mongodb was quick and easy. There was no entry barrier. I can't fault how quick and easy it was to get up and running with the basics. Hacking around to pick up the more advanced stuff was also a pretty painless exercise too. Using the C# driver has been a largely very positive and intuitive experience.

- **Replica sets**

Configuring is simple, making scaling reads and failover pretty effortless. Want more redundancy or scaling of reads? Fire up another machine, add to the set and away you go.

Mongo DB Pros

- **Auto Sharding**

Again, configuring is simple. You do need to give very careful consideration to this up front when deciding on what keys you want to shard on. Once you've done that, sharding "just does it's stuff".

- **Community**

It has a good community behind it and that IMHO is very important. I don't like sitting in a cupboard on my own with the lights off. I like being a part of a bigger community - to learn from, work through issues with and to contribute back to.

- **Rapidly evolving**

MongoDB is rapidly changing and it's good to see bugs are being tracked and fixed in good time. There is also a fast flowing feature enhancement pipeline too, so you typically don't have to wait for a long time to get something.

- **Choose your consistency**

You can choose to have data replicated to a configurable number of replicas before returning if you wish to have stronger level of consistency. Depends on what value you put on certain bits of data, but the choice is yours. So you can trade off performance for consistency.

Mongo DB cons

- Data size in MongoDB is typically higher due to e.g. each document has field names stored it
- less flexibity with querying (e.g. no JOINs)
- no support for transactions - certain atomic operations are supported, at a single document level
- at the moment Map/Reduce (e.g. to do aggregations/data analysis) is OK, but not blisteringly fast. So if that's required, something like Hadoop may need to be added into the mix
- less up to date information available/fast evolving product

Design decisions with Mongo

- Agile incremental releases
- Unstructured data from multiple suppliers
- GridFS : Stores large binary objects
- Spring Data Services
- Embedding and linking documents
- Easy replication set up for AWS

Books

MongoDB: The Definitive Guide, 2nd Edition
By: Kristina Chodorow
Publisher: O'Reilly Media, Inc.
Pub. Date: May 23, 2013
Print ISBN-13: 978-1-4493-4468-9
Pages in Print Edition: 432

MongoDB in Action
By: Kyle Banker
Publisher: Manning Publications
Pub. Date: December 16, 2011
Print ISBN-10: 1-935182-87-0
Print ISBN-13: 978-1-935182-87-0
Pages in Print Edition: 312

The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing
By Eelco Plugge; Peter Membrey; Tim Hawkins
Apress, September 2010
ISBN: 9781430230519
327 pages

Books Cont.

MongoDB Applied Design Patterns
By: Rick Copeland
Publisher: O'Reilly Media, Inc.
Pub. Date: March 18, 2013
Print ISBN-13: 978-1-4493-4004-9
Pages in Print Edition: 176

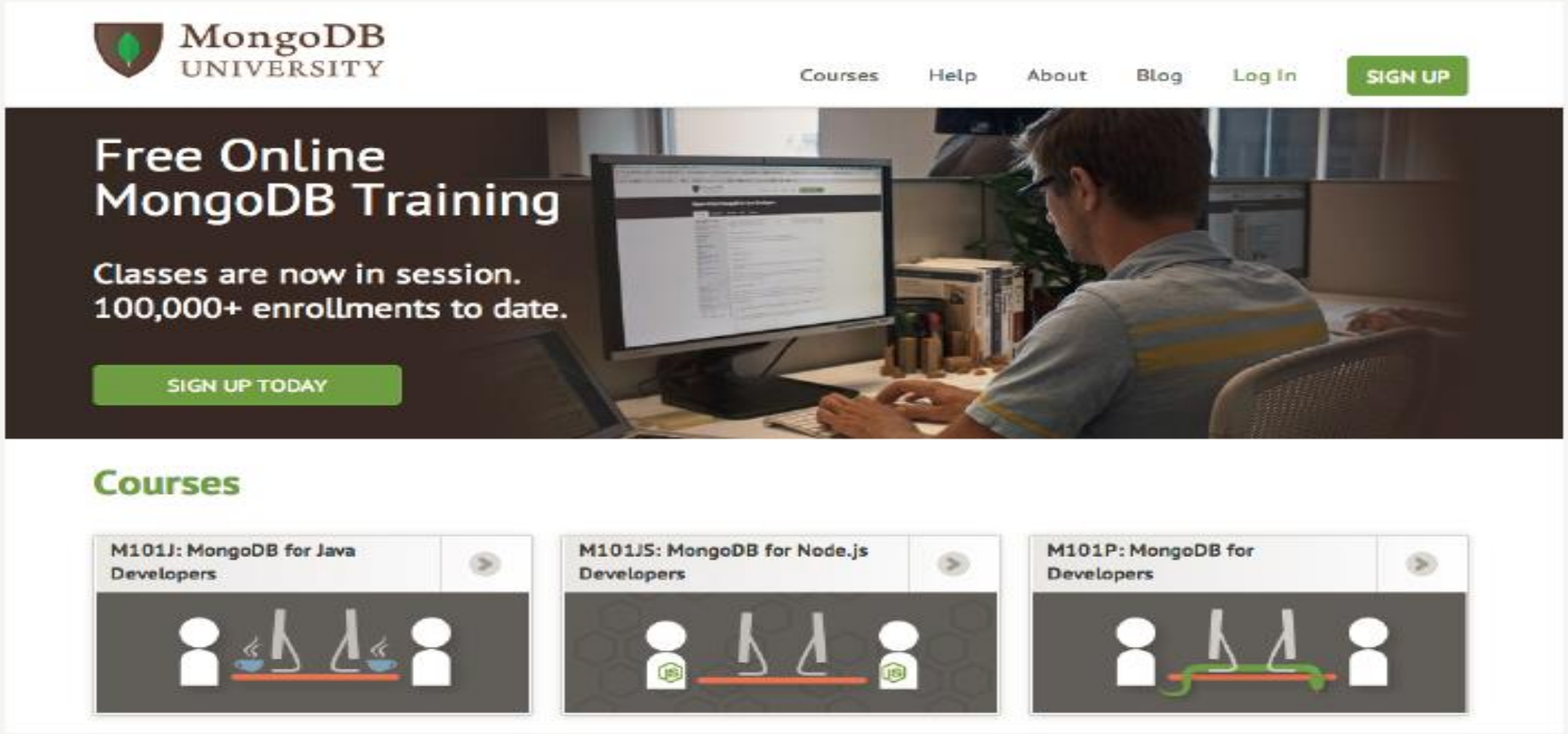
MongoDB for Web Development (rough cut!)
By: Mitch Pirtle
Publisher: Addison-Wesley Professional
Last Updated: 14-JUN-2013
Pub. Date: March 11, 2015 (Estimated)
Print ISBN-10: 0-321-70533-5
Print ISBN-13: 978-0-321-70533-4
Pages in Print Edition: 360

Instant MongoDB
By: Amol Nayak;
Publisher: Packt Publishing
Pub. Date: July 26, 2013
Print ISBN-13: 978-1-78216-970-3
Pages in Print Edition: 72

Important Sites

- http://www.mongodb.org/
- https://mongolab.com/welcome/
- https://education.mongodb.com/
- http://blog.mongodb.org/
- http://stackoverflow.com/questions/tagged/mongodb
- http://bitnami.com/stack/mean

Online Training at MongoDB University



For More Information

Resource	Location
MongoDB Downloads	mongodb.com/download
Free Online Training	education.mongodb.com
Webinars and Events	mongodb.com/events
White Papers	mongodb.com/white-papers
Case Studies	mongodb.com/customers
Presentations	mongodb.com/presentations
Documentation	docs.mongodb.org
Additional Info	info@mongodb.com

{“catchme on”:
[
{“name” : “kirankumar”,
“website” : [
“techincal”:”www.javaquestionbank.net”,
“personal” :www.matamkirankumar.com],
“twitter : ” @matamkiran11”
}
]
}

For more details
{“Linked In” : “in.linkedin.com/pub/kiran-kumar-matam/2b/239/91b/”}

☺Thank you ☺