#BigData #MongoDB

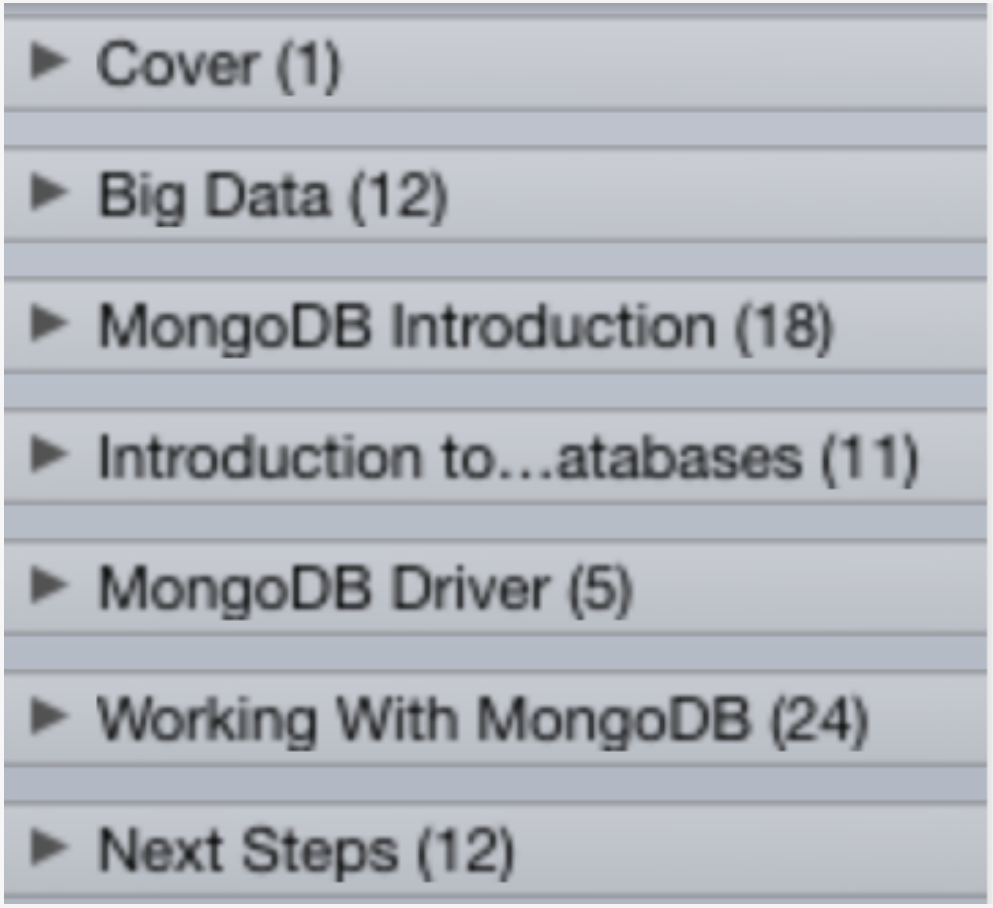
Big Data introduction using MongoDB

Prasoon Kumar

MongoDB Consultant

mongoDB

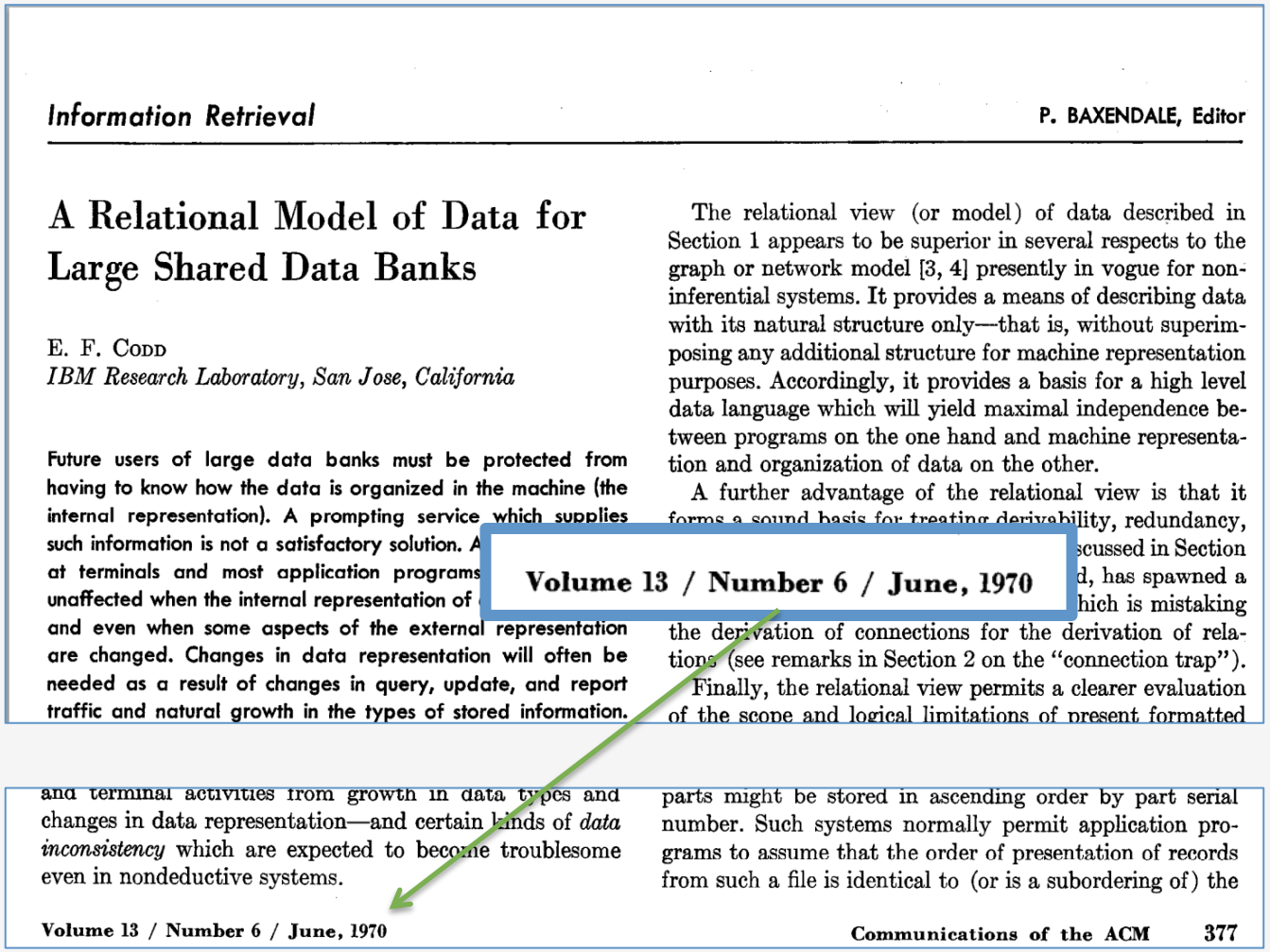
Agenda


Cover (1)
Big Data (12)
MongoDB Introduction (18)
Introduction to…atabases (11)
MongoDB Driver (5)
Working With MongoDB (24)
Next Steps (12)

Remember??

Who Are These Guys?

The relational model : 1970



Information Retrieval P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. A at terminals and most application programs unaffected when the internal representation of and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, scussed in Section d, has spawned a hich is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted

Volume 13 / Number 6 / June, 1970

and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

Traditional Data Approaches

- **Filter – Store – Distribute**
  - Encyclopedias
  - Newspapers
  - Libraries
  - Banking
- Why?
  - Storage caps
  - Bandwidth caps

What Happened in 2006?

- North Korea tested a Nuclear Device
- Italy defeated France in the World Cup
- Pluto is demoted to a Dwarf planet

What Happened in 2006?

- North Korea tested a Nuclear Device
- Italy defeated France in the World Cup
- Pluto is demoted to a Dwarf planet
- Twitter launched
- Google bought YouTube for 1.65bn
- **Amazon launched S3**
  - **The first Amazon Web Service (AWS)**

The AWS Disruption

- Cost per GB/Month for Stable Storage
  - ~$5GB down to .15 cent per GB
- Unlimited Storage
- Purchased in GB chunks
- Pay only for what you use

Store Everything Is A Challenge

- SQL Databases
  - depend on a pre-filter
  - Assume monolithic memory
  - Assume single disk farm
  - Hard to partition
  - Based on 1970's storage assumptions

This was a problem for Google

2010 Search Index Size:
100,000,000 GB

New data added per day
100,000+ GB

Databases they could use
0

Source: http://googleblog.blogspot.com/2010/06/our-new-search-index-caffeine.html

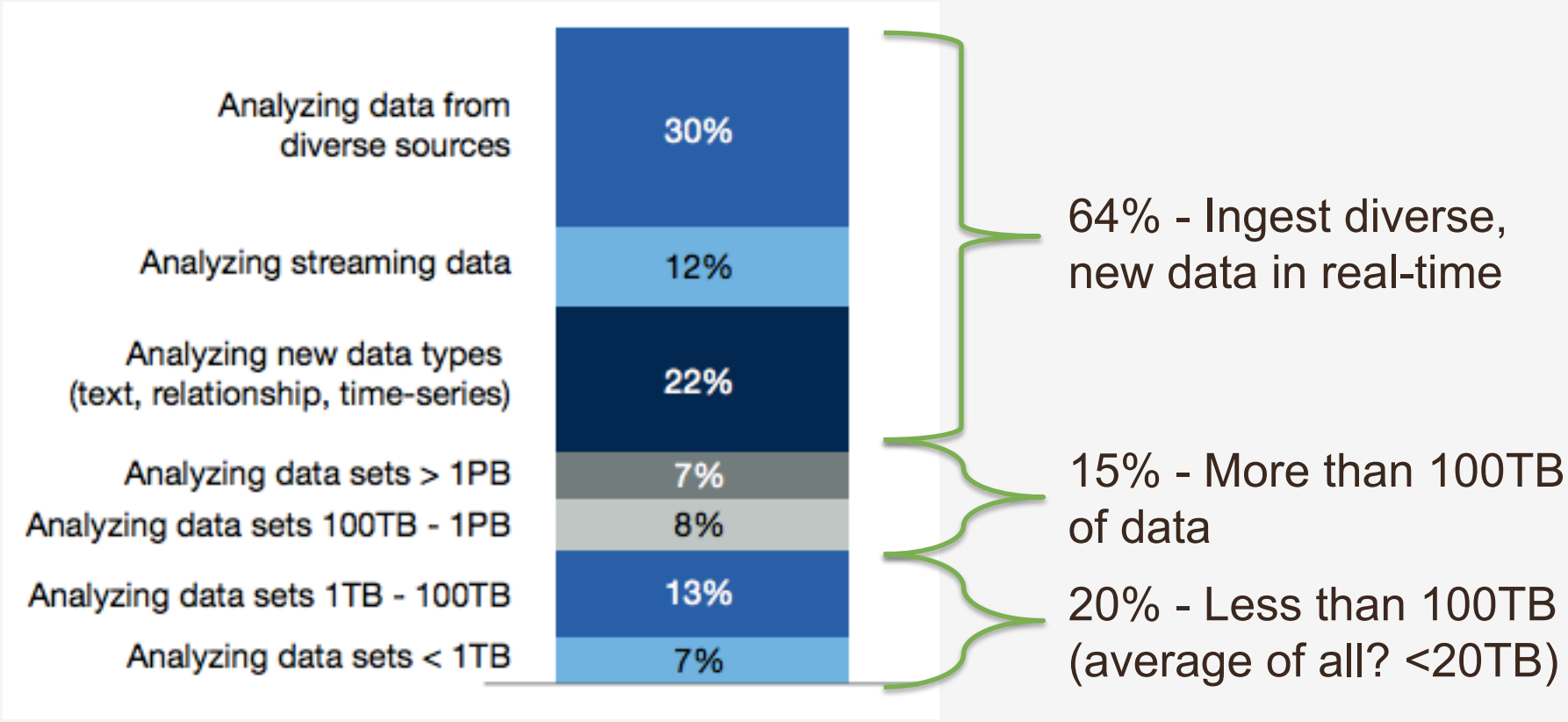
Why NoSQL

Velocity

Variety

Volume

Understanding Big Data – It's Not Very "Big"



from Big Data Executive Summary – 50+ top executives from Government and F500 firms

Unlock Your Big Data

What is MongoDB?

mongoDB

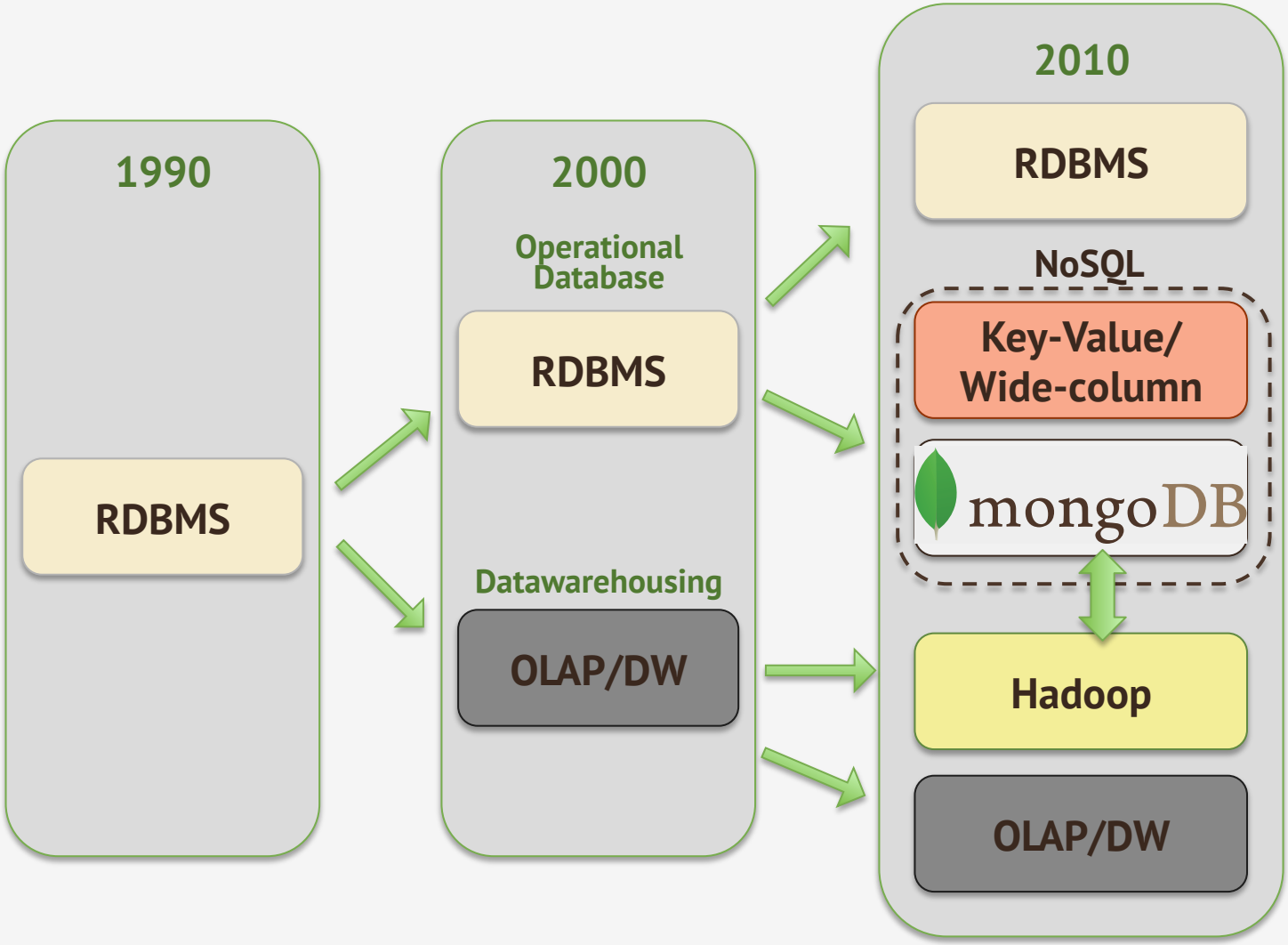
MongoDB is a _____ database

1. Document
2. Open source
3. High performance
4. Horizontally scalable
5. Full featured

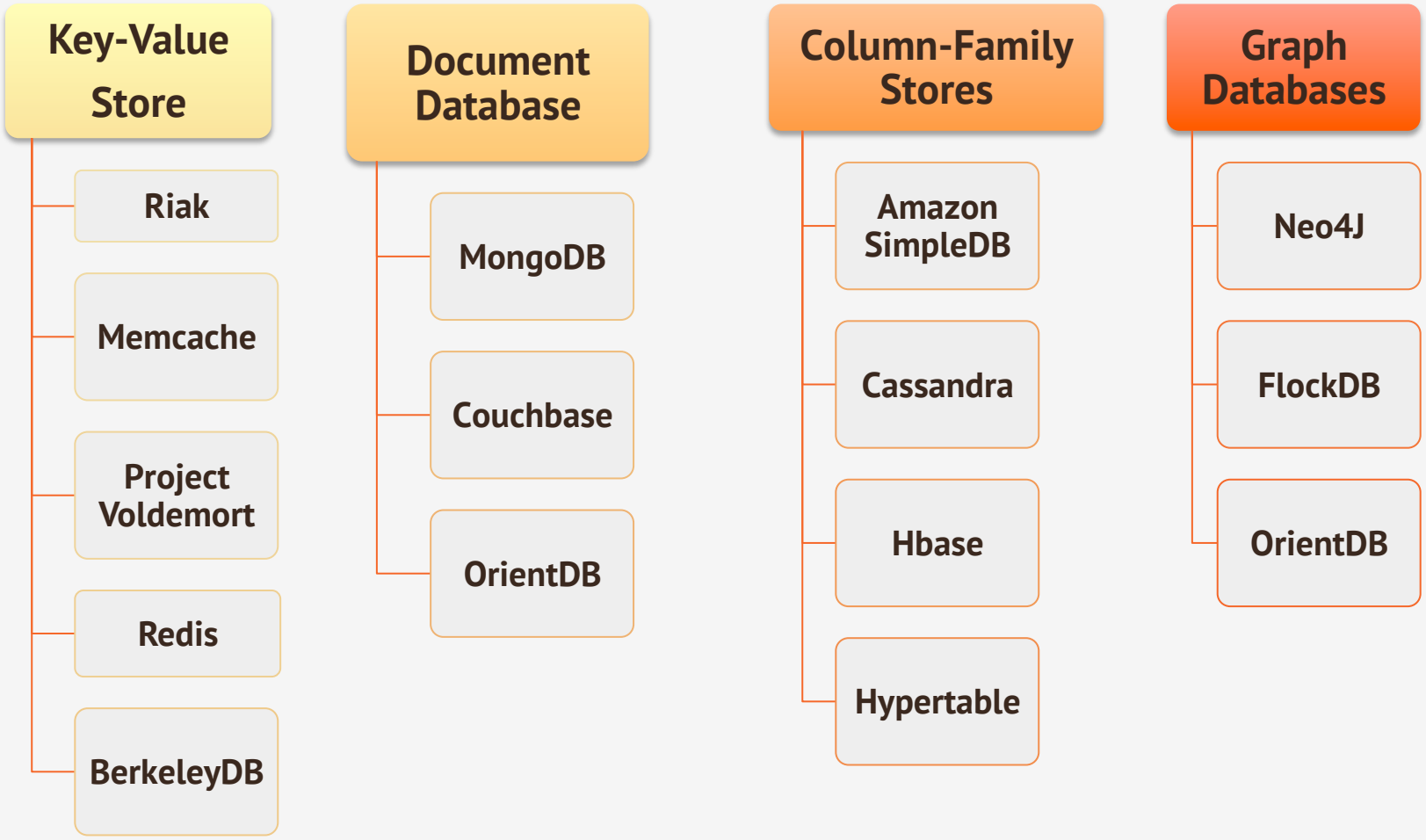
1. Document Database

- 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

1. Database Evolution



1. NoSQL Data Model



2. Open Source

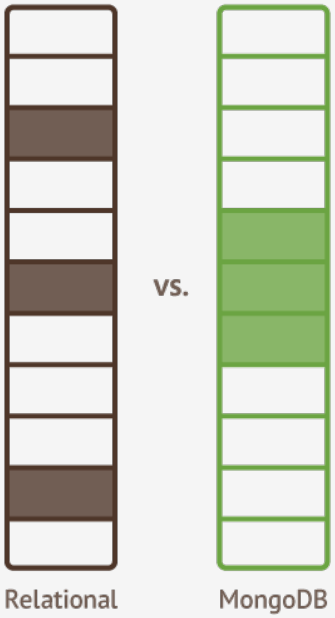
- MongoDB is an open source project
- On GitHub
- Licensed under the AGPL
- Started & sponsored by MongoDB Inc (formerly known as 10gen)
- Commercial licenses available
- Contributions welcome

2. Global Community

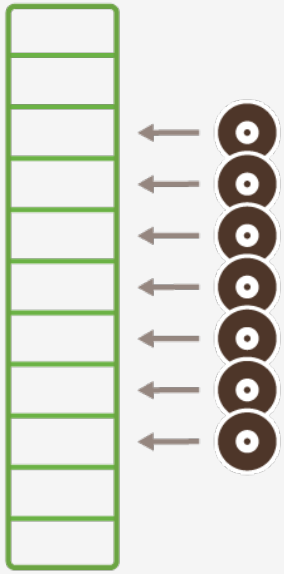
3. High Performance

- 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

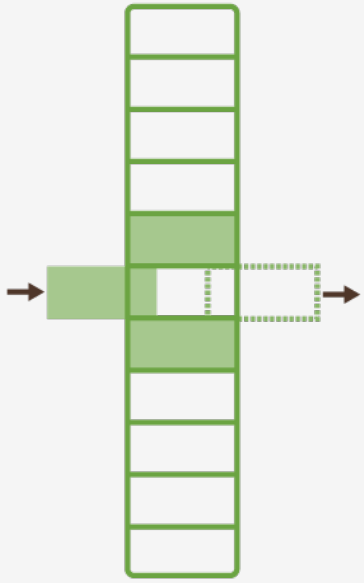
3. Performance



Better Data Locality

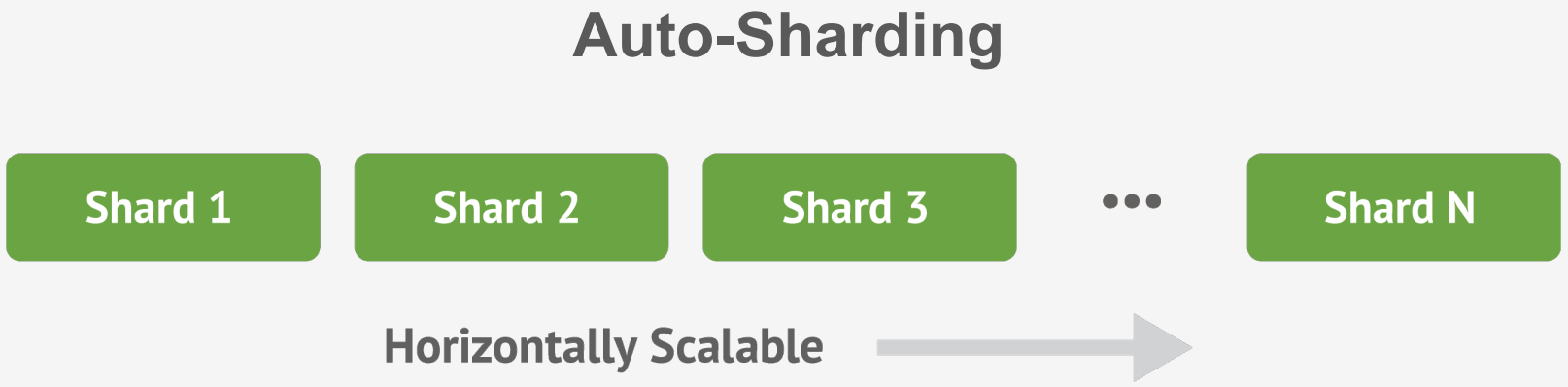


In-Memory Caching



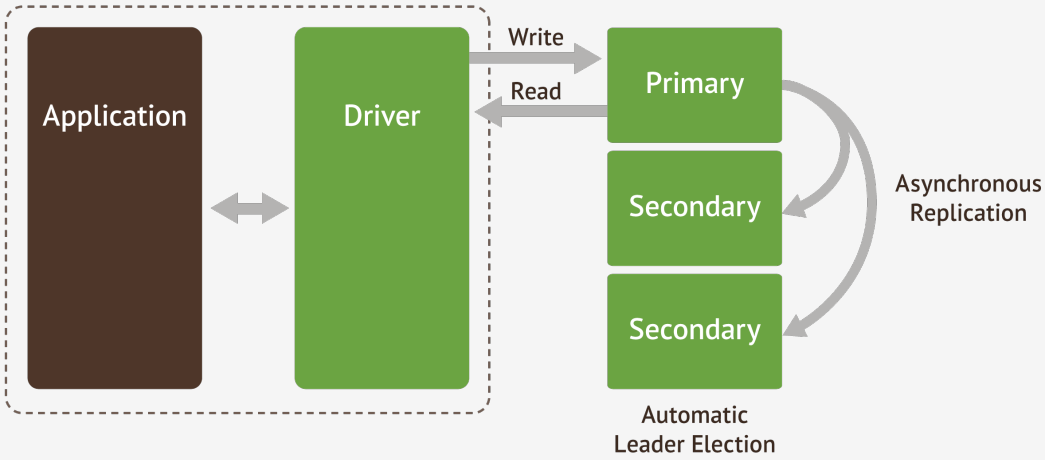
In-Place Updates

4. Scalability



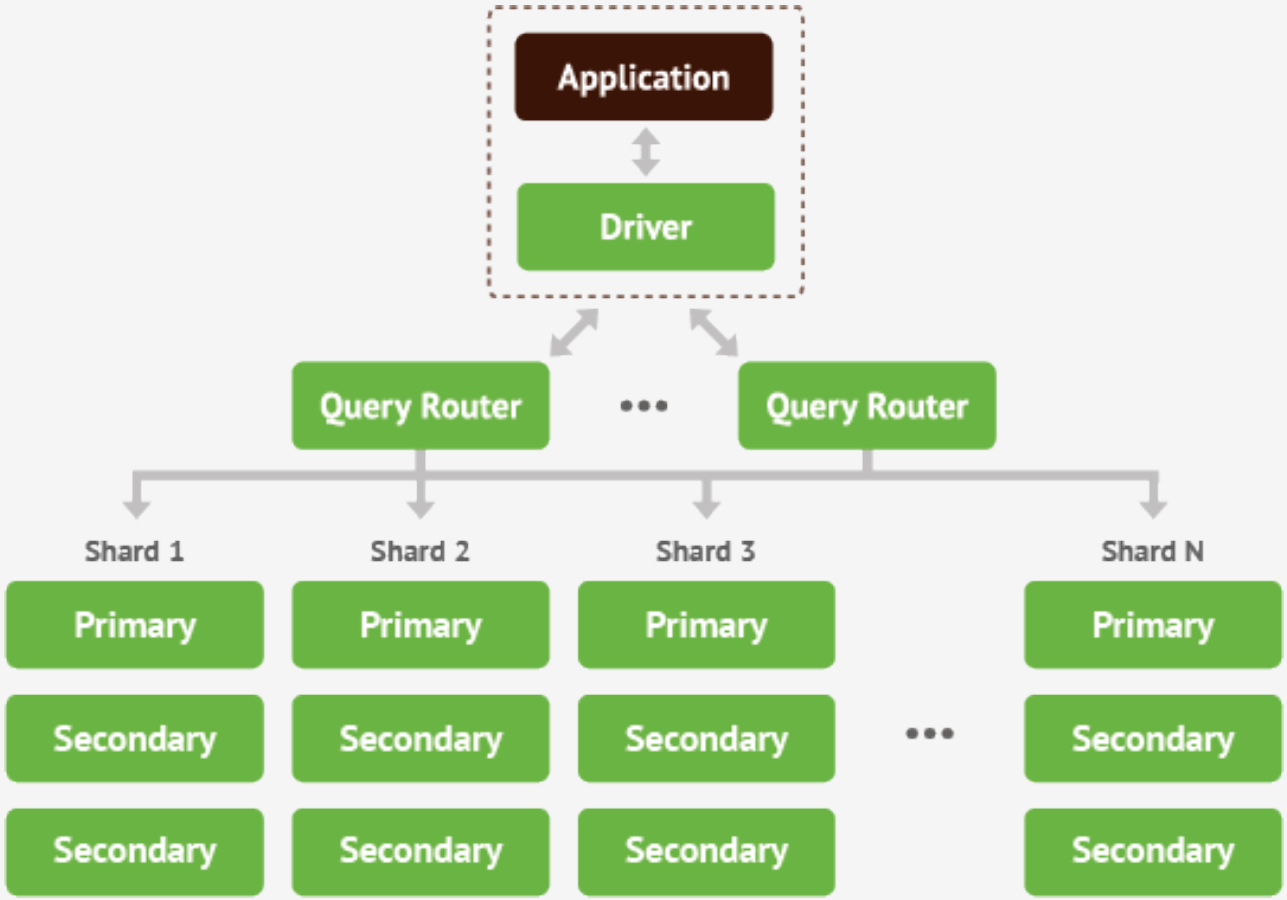
- Increase capacity as you go
- Commodity and cloud architectures
- Improved operational simplicity and cost visibility

4. High Availability



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

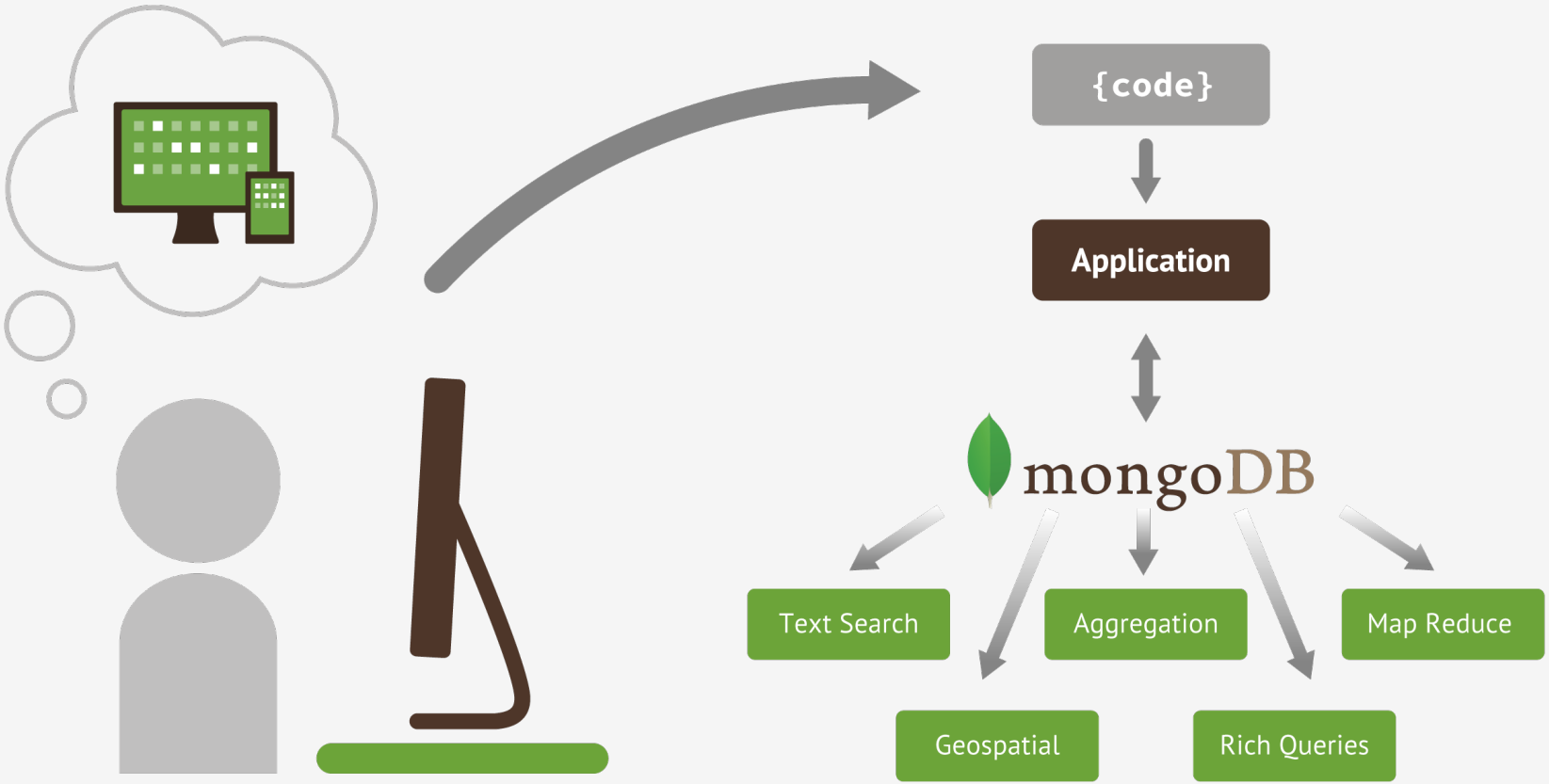
4. MongoDB Architecture



5. Full Featured

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

5. MongoDB is Fully Featured

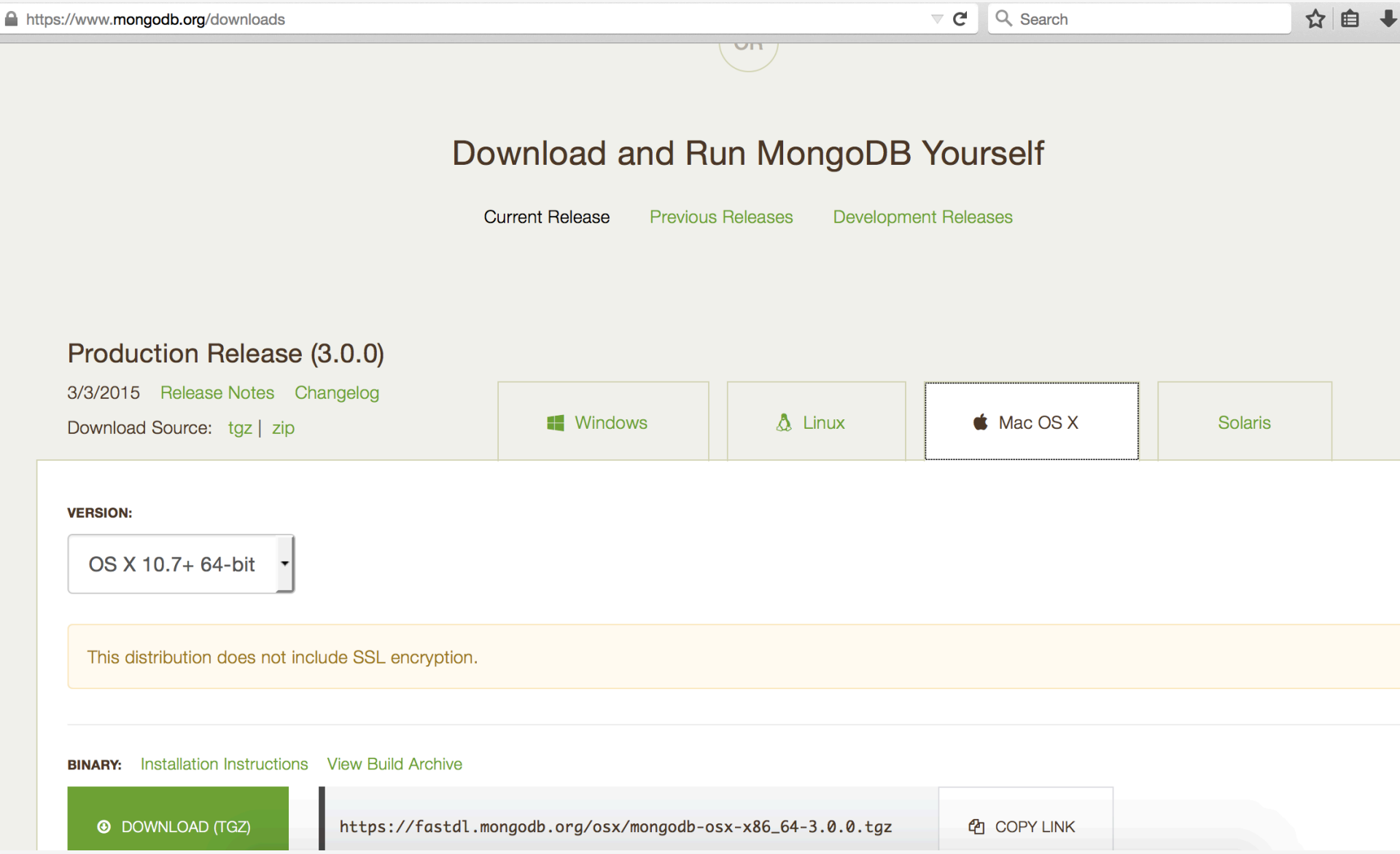


5. MongoDB is full featured

Rich Queries	• Find Ruby's cars • Find everybody who has an office mobile
Geospatial	• Find all of the employees in Gurgaon
Text Search	• Find all the contacts described as being Mobile
Aggregation	• What's the average number of contacts per person in the entire database
Map Reduce	• For each year, how many employees' contact exist?

MongoDB

```
{
  first_name: 'Ruby',
  surname: 'Mishra',
  city: 'Bangalore',
  location: [45.123,47.232],
  contact: [
    { type: 'Office Mobile',
      yearSince: 2011,
      value: 9922994444, … },
    { type: 'Personal Email',
      value: "ruby2601@gmail.com",
            … }
  ]
}
```


https://www.mongodb.org/downloads
Search
Download and Run MongoDB Yourself
Current Release
Previous Releases
Development Releases
Production Release (3.0.0)
3/3/2015
Release Notes
Changelog
Download Source:
tgz | zip
Windows
Linux
Mac OS X
Solaris
VERSION:
OS X 10.7+ 64-bit
This distribution does not include SSL encryption.
BINARY:
Installation Instructions
View Build Archive
DOWNLOAD (TGZ)
https://fastdl.mongodb.org/osx/mongodb-osx-x86_64-3.0.0.tgz
COPY LINK

mongoDB

Running MongoDB

```
$ tar –zxvf mongodb-osx-x86_64-2.6.0.tgz
$ cd mongodb-osx-i386-2.6.0/bin
$ mkdir –p /data/db
$ ./mongod
```

Mongo Shell

```
MacBook-Pro-:~ $ mongo
MongoDB shell version: 2.6.0
connecting to: test
> db.articles.insert({text: 'Welcome to MongoDB'})
> db.articles.find().pretty()
{
	"_id" : ObjectId("51c34130fbd5d7261b4cdb55"),
	"text" : "Welcome to MongoDB"
}
```

_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
```

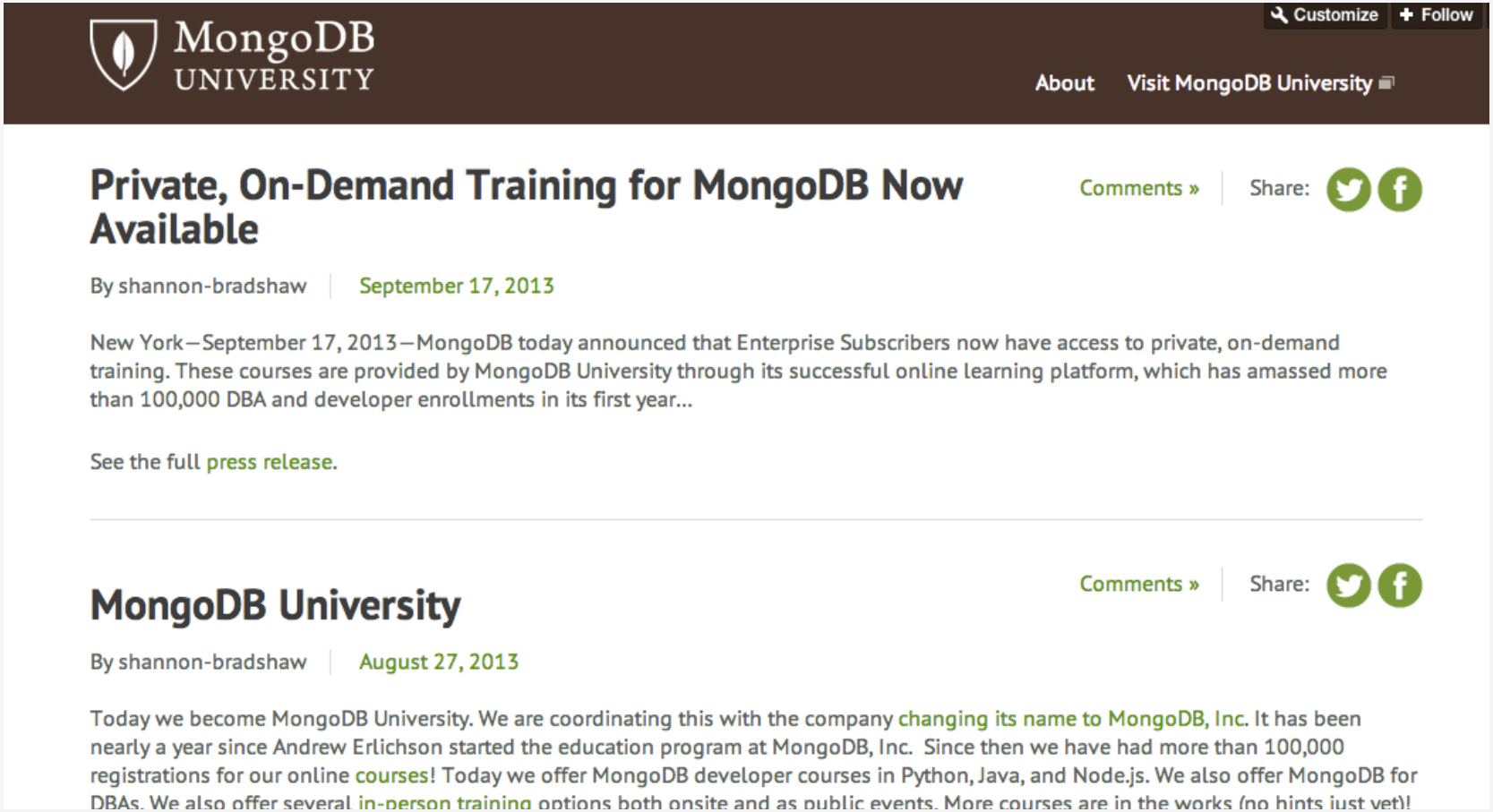
Document Database

mongoDB

Terminology

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

Let's Build a Blog



First step in any application is

Determine your entities

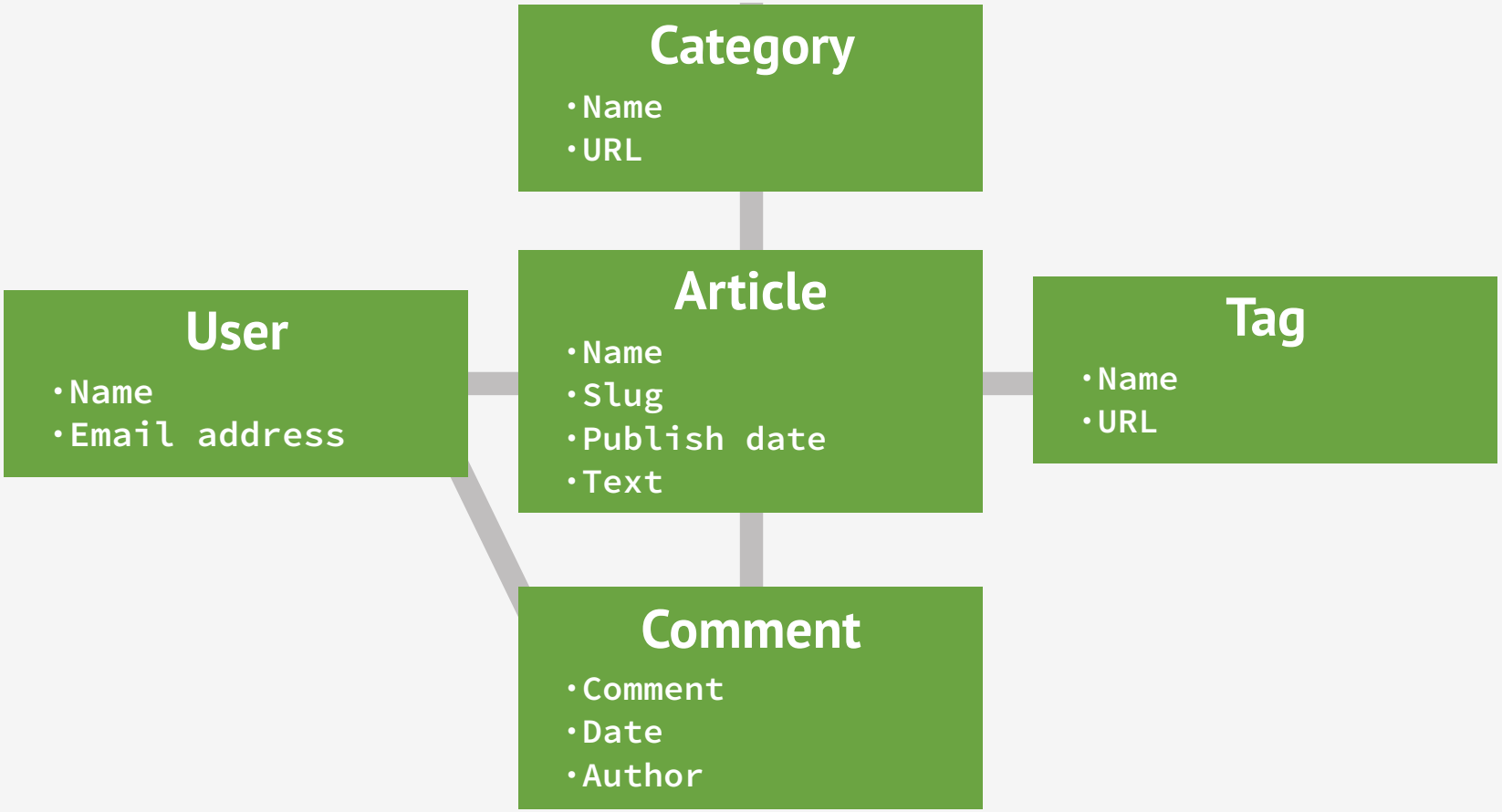
Entities in our Blogging System

- Users (post authors)
- Article
- Comments
- Tags, Category
- Interactions (views, clicks)

In a relational base app

We would start by doing schema design

Typical (relational) ERD

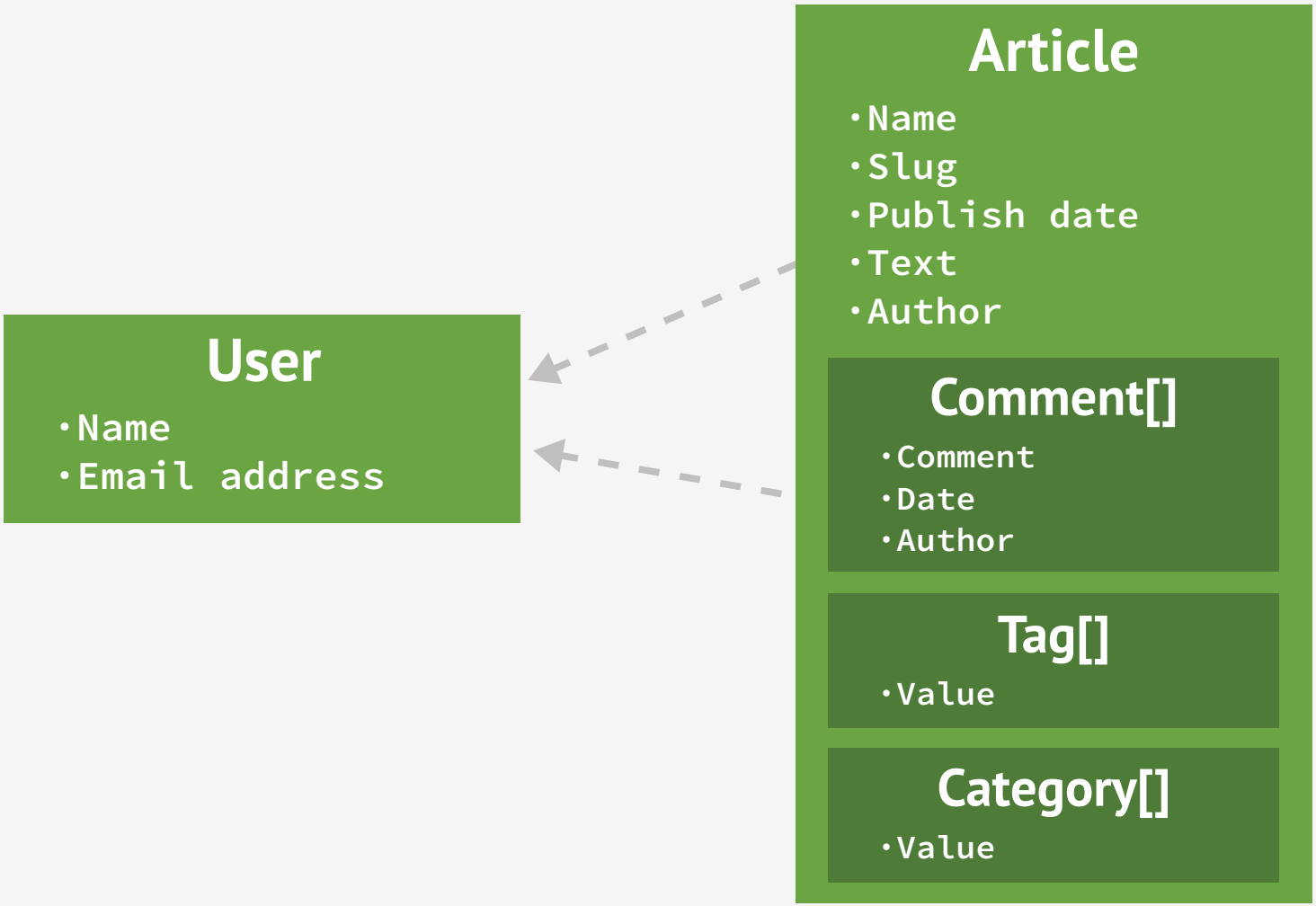


In a MongoDB based app

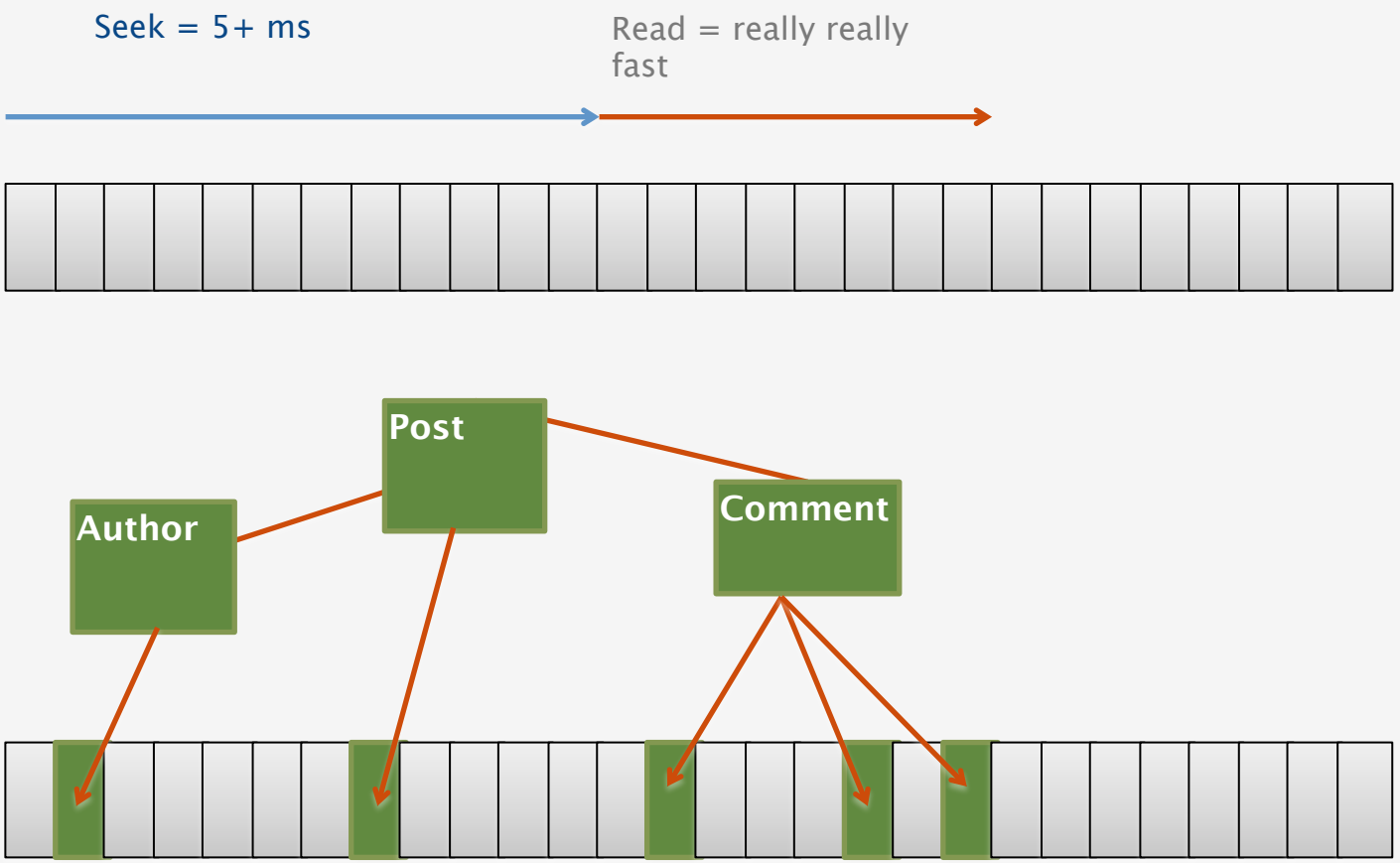
We start building our app

and let the schema evolve

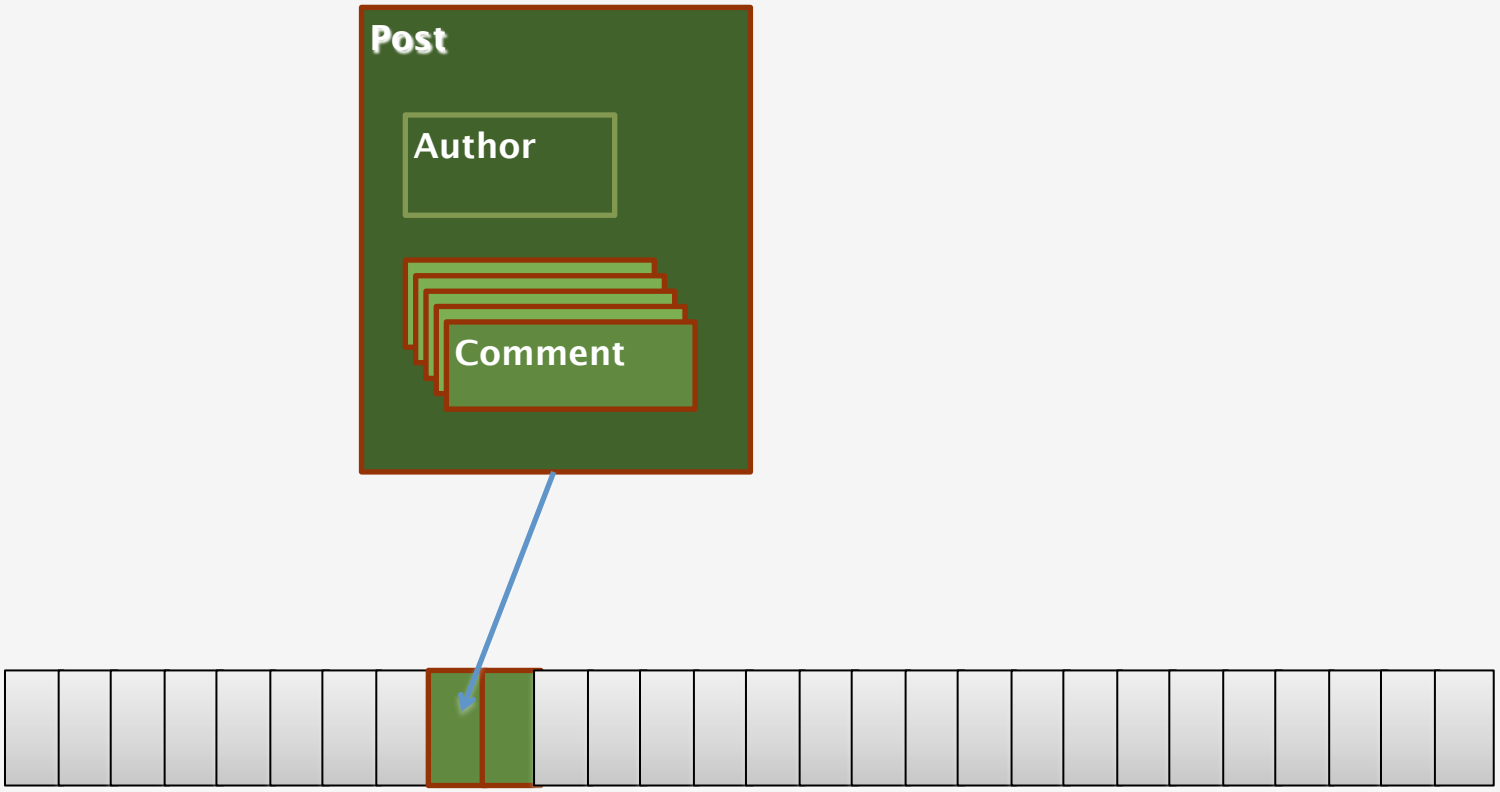
MongoDB ERD


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

Disk seeks and data locality



Disk seeks and data locality



MongoDB Language Driver

mongoDB

Real applications are not built in the shell

MongoDB has native bindings for over 12 languages

Drivers & Ecosystem

Drivers


Java


Ruby


Perl

Support for the most popular languages and frameworks


Python

Frameworks

MEAN Stack

Morphia

MongoDB Drivers

- Official Support for 12 languages
- Community drivers for tons more
- Drivers connect to mongo servers
- Drivers translate BSON into native types
- mongo shell is not a driver, but works like one in some ways
- Installed using typical means (maven, npm, pecl, gem, pip)

Working With MongoDB

mongoDB

Design schema.. In application code

```
# Python dictionary (or object)
>>> article = { ‘title’ : ‘Schema design in MongoDB’,
                ‘author’ : ‘prasoonk’,
                ‘section’ : ‘schema’,
                ‘slug’ : ‘schema-design-in-mongodb’,
                ‘text’ : ‘Data in MongoDB has a flexible schema.
                          So, 2 documents needn’t have same structure.
                          It allows implicit schema to evolve.’,
                ‘date’ :  datetime.utcnow(),
                ‘tags’ : [‘MongoDB’, ‘schema’] }
>>> db[‘articles’].insert(article)
```

Let's add a headline image

```
>>> img_data = Binary(open('article_img.jpg').read())
>>> article = { 'title' : 'Schema evolutionin MongoDB',
                'author' : 'mattbates',
                'section' : 'schema',
                'slug' : 'schema-evolution-in-mongodb',
                'text' : 'MongoDb has dynamic schema. For good
                          performance, you would need an implicit
                          structure and indexes',
               'date' :  datetime.utcnow(),
               'tags' : ['MongoDB', 'schema', 'migration'],
               'headline_img' : {
                 'img' : img_data,
                 'caption' : 'A sample document at the shell'
               }}
```

And different types of article

```
>>> article = { ‘title’ : ‘Favourite web application framework’,
                ‘author’ : ‘prasoonk’,
                ‘section’ : ‘web-dev’,
                ‘slug’ : ‘web-app-frameworks’,
                ‘gallery’ : [
                    { ‘img_url’ : ‘http://x.com/45rty’, ‘caption’ : ‘Flask’, ..},
                    ..
                ]
                ‘date’ :  datetime.utcnow(),
                ‘tags’ : [‘Python’, ‘web’],
              }
>>> db[‘articles’].insert(article)
```

Users and profiles

```
>>> user = {
        'user' : 'prasoonk',
        'email' : 'prasoon.kumar@mongodb.com',
        'password' : 'prasoon101',
        'joined' : datetime.utcnow(),
        'location' : { 'city' : 'Mumbai' },
}
}
>>> db[‘users’].insert(user)
```

Retrive using Comparison Operators

$gt, $gte, $in, $lt, $lte, $ne, $nin

- Use to query documents

```
db.articles.find(    { 'title' :   'Intro to MongoDB' } )

db.articles.find(    { 'date' :   { '$lt' :
                                    {ISODate("2014-02-19T00:00:00.000Z") }} )

db.articles.find(    { 'tags' : {   '$in' : ['nosql','database'] } } );
```

- **Logical**: $or, $and, $not, $nor
- **Element**: $exists, $type
- **Evaluation**: $mod, $regex, $where
- **Geospatial**: $geoWithin, $geoIntersects, $near, $nearSphere

Modelling comments (1)

- Two collections – **articles** and **comments**
  - Use a reference (i.e. foreign key) to link together
  - But.. N+1 queries to retrieve article **and** comments

```
{
    '_id' : ObjectId(..),
    'title' : 'Schema design in MongoDB',
    'author' : 'mattbates',
    'date' : ISODate(..),
    'tags' : ['MongoDB', 'schema'],
    'section' : 'schema',
    'slug' : 'schema-design-in-mongodb',
    'comments' : [ ObjectId(..),...]
}
```

```
{   '_id' : ObjectId(..),
    'article_id' : 1,
    'text' : 'A great article, helped me
              understand schema design',
    'date' : ISODate(..),,
    'author' : 'johnsmith'
}
```

Comparison Operators

$gt, $gte, $in, $lt, $lte, $ne, $nin

- Use to query documents

```
db.articles.find(    { 'title' :   'Intro to MongoDB' } )

db.articles.find(    { 'date' :   { '$lt' :
      {ISODate("2014-02-19T00:00:00.000Z") }} )

db.articles.find(    { 'tags' : {   '$in' : ['nosql','database'] } } );
```

- **Logical**: $or, $and, $not, $nor
- **Element**: $exists, $type
- **Evaluation**: $mod, $regex, $where
- **Geospatial**: $geoWithin, $geoIntersects, $near, $nearSphere

Modelling comments (2)

- Single **articles** collection – embed comments in article documents
- Pros
  - Single query, document designed for the access pattern
  - Locality (disk, shard)
- Cons
  - Comments array is unbounded; documents will grow in size (remember 16MB document limit)

```
{
    '_id' : ObjectId(..),
    'title' : 'Schema design in MongoDB',
    'author' : 'mattbates',
    'date' : ISODate(..),
    'tags' : ['MongoDB','schema'],
    ...
    'comments' : [
        {
          'text' : 'A great article, helped me
          understand schema design',
          'date' : ISODate(..),
          'author' : 'johnsmith'
        },
        ...
    ]
}
```

Modelling comments (3)

- Another option: hybrid of (2) and (3), embed top x comments (e.g. by date, popularity) into the article document
  - Fixed-size (2.4 feature) comments array
  - All other comments 'overflow' into a comments collection (double write) in buckets
  - Pros
    - Document size is more fixed – fewer moves
    - Single query built
    - Full comment history with rich query/aggregation

Modelling comments (3)

```
{
        '_id' : ObjectId(..),
        'title' : 'Schema design in MongoDB',
        'author' : 'mattbates',
        'date' : ISODate(..),
        'tags' : ['MongoDB', 'schema'],
        ...
        'comments_count' : 45,
        'comments_pages' : 1
        'comments' : [
                {
                  'text' : 'A great article, helped me
                        understand schema design',
                  'date' : ISODate(..),
                  'author' : 'johnsmith'
                },
                ...
        ]
}
```

Total number of comments

- Integer counter updated by update operation as comments added/removed

Number of pages

- Page is a bucket of 100 comments (see next slide..)

Fixed-size comments array

- 10 most recent
- Sorted by date on insertion

Modelling comments (3)

```
{
      '_id' : ObjectId(..),
      'article_id' : ObjectId(..),
      'page' : 1,
      'count' : 42
      'comments' : [
        {
          'text' : 'A great article, helped me
                    understand schema design',
          'date' : ISODate(..),
          'author' : 'johnsmith'
        },
        ...
}
```

One comment bucket (page) document containing up to about 100 comments

Array of 100 comment sub-documents

Modelling interactions

- Interactions
  - Article views
  - Comments
  - (Social media sharing)
- Requirements
  - Time series
  - Pre-aggregated in preparation for analytics

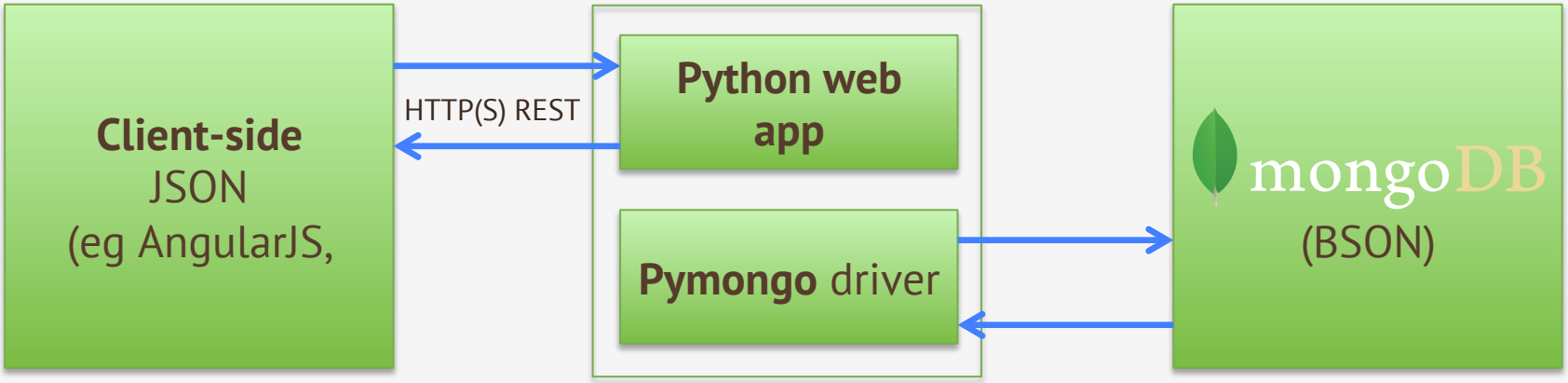
Modelling interactions

- Document per article per day – 'bucketing'
- Daily counter and hourly sub-document counters for interactions
- Bounded array (24 hours)
- Single query to retrieve daily article interactions; ready-made for graphing and further aggregation

```
{
    '_id' : ObjectId(..),
    'article_id' : ObjectId(..),
    'section' : 'schema',
    'date' : ISODate(..),
    'daily' : { 'views' : 45,'comments' : 150 }
    'hours' : {
      0 : { 'views' : 10 },
      1 : { 'views' : 2 },
      ...
      23 : { 'comments' : 14,'views' : 10 }
    }
}
```

JSON and RESTful API

Real applications are not built at a shell – let's build a RESTful API.



Examples to follow: Python RESTful API using Flask microframework

myCMS REST endpoints

Method	URI	Action
GET	/articles	Retrieve all articles
GET	/articles-by-tag/[tag]	Retrieve all articles by tag
GET	/articles/[article_id]	Retrieve a specific article by article_id
POST	/articles	Add a new article
GET	/articles/[article_id]/comments	Retrieve all article comments by article_id
POST	/articles/[article_id]/comments	Add a new comment to an article.
POST	/users	Register a user user
GET	/users/[username]	Retrieve user's profile
PUT	/users/[username]	Update a user's profile

Getting started with the skeleton code

```
$ git clone http://www.github.com/prasoonk/mycms_mongodb
$ cd mycms-mongodb
$ virtualenv venv
$ source venv/bin/activate
$ pip install –r requirements.txt

$ mkdir –p data/db
$ mongod --dbpath=data/db --fork --logpath=mongod.log

$ python web.py
[$ deactivate]
```

RESTful API methods in Python + Flask

```
@app.route('/cms/api/v1.0/articles', methods=['GET'])
def get_articles():
  """Retrieves all articles in the collection
  sorted by date
  """
  # query all articles  and return a cursor sorted by date
  cur = db['articles'].find().sort('date’)
  if not cur:
    abort(400)
  # iterate the cursor and add docs to a dict
  articles = [article for article in cur]
  return jsonify({'articles' : json.dumps(articles, default=json_util.default)})
```

RESTful API methods in Python + Flask

```
@app.route('/cms/api/v1.0/articles/<string:article_id>/comments', methods = ['POST'])
def add_comment(article_id):
  """Adds a comment to the specified article and a
  bucket, as well as updating a view counter
  """
  …
page_id = article['last_comment_id'] // 100
…
  # push the comment to the latest bucket and $inc the count
  page = db['comments'].find_and_modify(
    { 'article_id' : ObjectId(article_id),
      'page' : page_id},
    { '$inc' : { 'count' : 1 },
      '$push' : {
        'comments' : comment } },
    fields= {'count' : 1},
    upsert=True,
    new=True)
```

RESTful API methods in Python + Flask

```
# $inc the page count if bucket size (100) is exceeded
 if page['count'] > 100:
   db.articles.update(
     { '_id' : article_id,
       'comments_pages': article['comments_pages'] },
     { '$inc': { 'comments_pages': 1 } } )
 # let's also add to the article itself
 # most recent 10 comments only
 res = db['articles'].update(
   {'_id' : ObjectId(article_id)},
   {'$push' : {'comments' : { '$each' : [comment],
                '$sort' : {'date' : 1 },
                '$slice' : -10}},
    '$inc' : {'comment_count' : 1}})
…
```

RESTful API methods in Python + Flask

```
def add_interaction(article_id, type):
  """Record the interaction (view/comment) for the
  specified article into the daily bucket and
  update an hourly counter
  """
  ts = datetime.datetime.utcnow()
  # $inc daily and hourly view counters in day/article stats bucket
  # note the unacknowledged w=0 write concern for performance
  db['interactions'].update(
    { 'article_id' : ObjectId(article_id),
      'date' : datetime.datetime(ts.year, ts.month, ts.day)},
    { '$inc' : {
        'daily.{}'.format(type) : 1,
        'hourly.{}.{}'.format(ts.hour, type) : 1
    }},
    upsert=True,
    w=0)
```

Testing the API – retrieve articles

```
$ curl -i http://localhost:5000/cms/api/v1.0/articles
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 335
Server: Werkzeug/0.9.4 Python/2.7.5
Date: Thu, 10 Apr 2014 16:00:51 GMT

{
  "articles": "[{\"title\": \"Schema design in MongoDB\", \"text\": \"Data in MongoDB has a flexible schema..\", \"section\": \"schema\", \"author\": \"prasoonk\", \"date\": {\"$date\": 1397145312505}, \"_id\": {\"$oid\": \"5346bef5f2610c064a36a793\"}, \"slug\": \"schema-design-in-mongodb\", \"tags\": [\"MongoDB\", \"schema\"]}]"}
```

Testing the API – comment on an article

$ curl -H "Content-Type: application/json" -X POST -d '{"text":"An interesting article and a great read."}' http://localhost:5000/cms/api/v1.0/articles/52ed73a30bd031362b3c6bb3/comments

{

"comment": "{\"date\": {\"$date\": 1391639269724}, \"text\": \"An interesting article and a great read.\"}”

}

Schema iteration

New feature in the backlog?

Documents have **dynamic** schema so we just **iterate** the object schema.

```
>>> user = { 'username' : 'matt',
             'first' : 'Matt',
             'last' : 'Bates',
             'preferences' : { 'opt_out' : True } }
>>> user.save(user)
```

Next Steps

mongoDB

Further reading

- 'myCMS' skeleton source code: http://www.github.com/prasoonk/mycms_mongodb
- Data Models http://docs.mongodb.org/manual/data-modeling/
- Use case - metadata and asset management: http://docs.mongodb.org/ecosystem/use-cases/metadata-and-asset-management/
- Use case - storing comments: http://docs.mongodb.org/ecosystem/use-cases/storing-comments/

docs.mongodb.org

The MongoDB 3.0 Manual

MONGODB 3.0 RELEASED:

See Release Notes for MongoDB 3.0 for new features in MongoDB 3.0.

Welcome to the MongoDB Manual! MongoDB is an open-source, document-oriented database designed for ease of development and scaling.

The Manual introduces MongoDB and continues to describe the query language, operational considerations and procedures, administration, and application development patterns, and other aspects of MongoDB use and administration. See the Introduction to MongoDB for an overview of MongoDB's key features. The Manual also has a thorough reference section of the MongoDB interface and tools.

This manual is under constant development. See the About MongoDB Documentation for more information on the MongoDB Documentation project.

Getting Started	Developers	Administrators	Reference
Introduction to MongoDB	Database Operations	Production Notes	Shell Methods
Installation Guides	Aggregation	Replica Sets	Query Operators

Online Training at MongoDB University

https://university.mongodb.com

Search

ONLINE COURSES

M101P: MongoDB for Developers

M101JS: MongoDB for Node.js Developers

M101N: MongoDB for .NET Developers

M101J: MongoDB for Java Developers

M102: MongoDB for DBAs

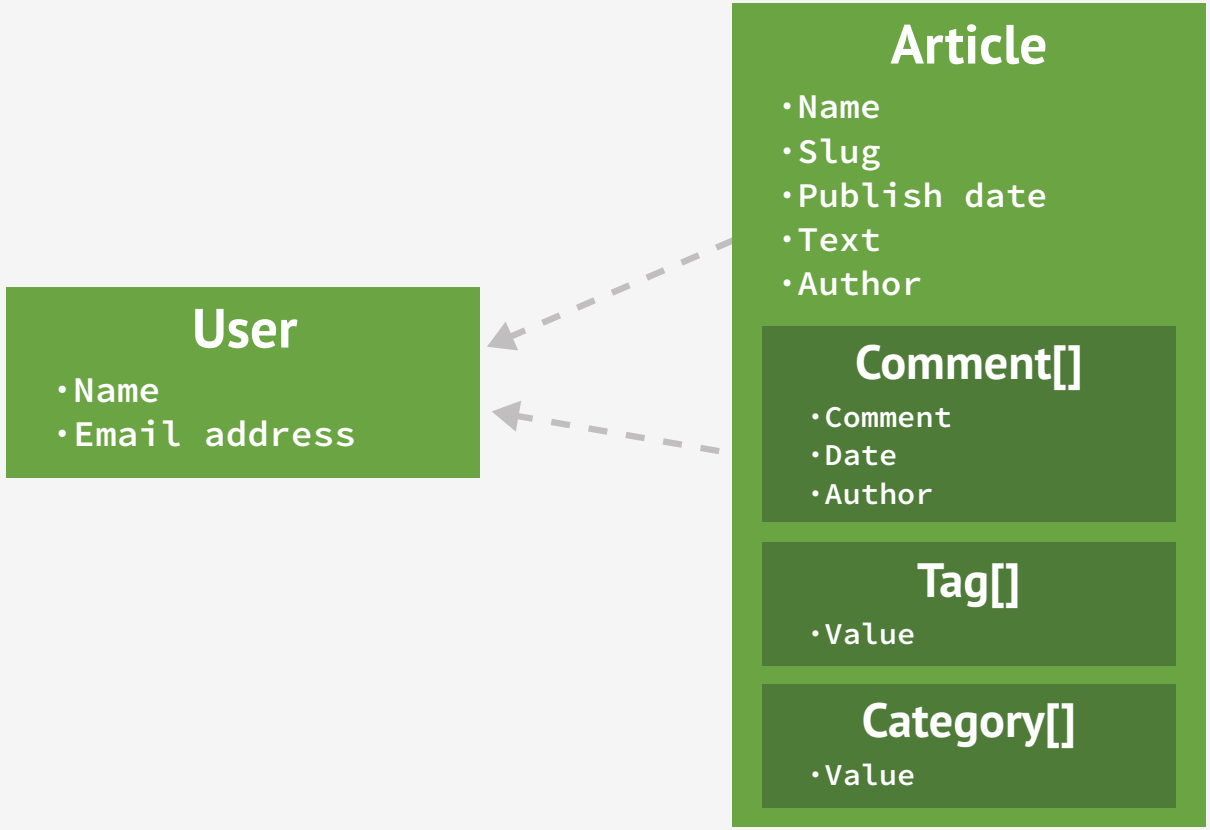
M202: MongoDB Advanced Deployments and Operations

For More Information

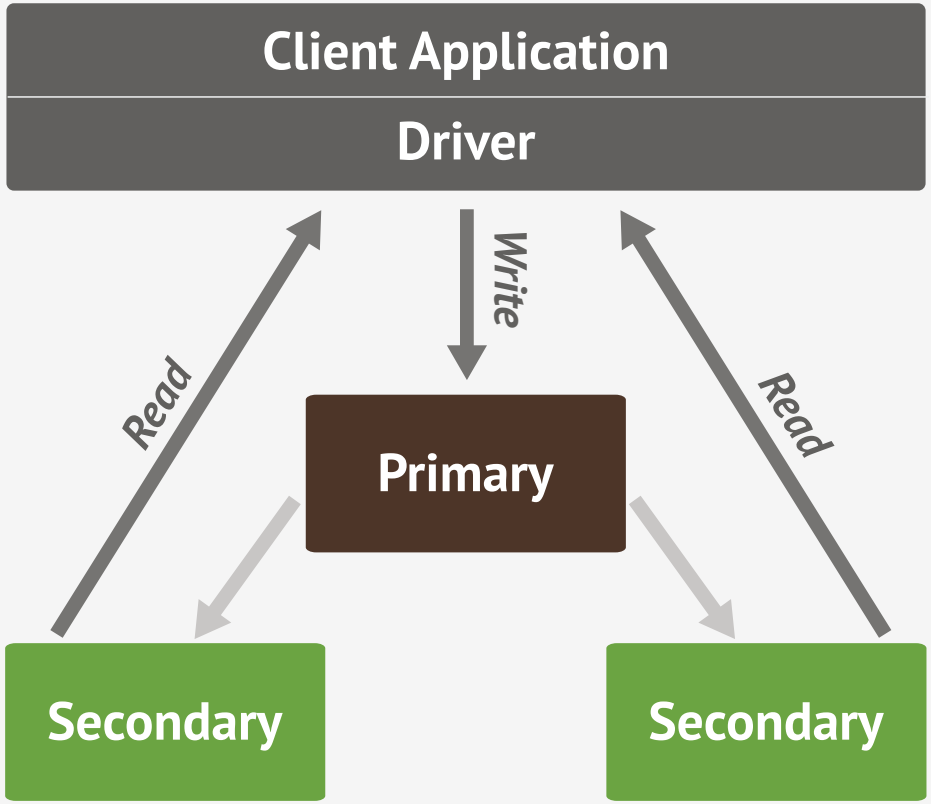
Resource	Location
MongoDB Downloads	mongodb.com/download
Free Online Training	education.mongodb.com
Webinars and Events	mongodb.com/events
White Papers	mongodb.com/white-papers
Case Studies	mongodb.com/customers
Presentations	mongodb.com/presentations
Documentation	docs.mongodb.org
Additional Info	info@mongodb.com

We've introduced a lot of concepts here

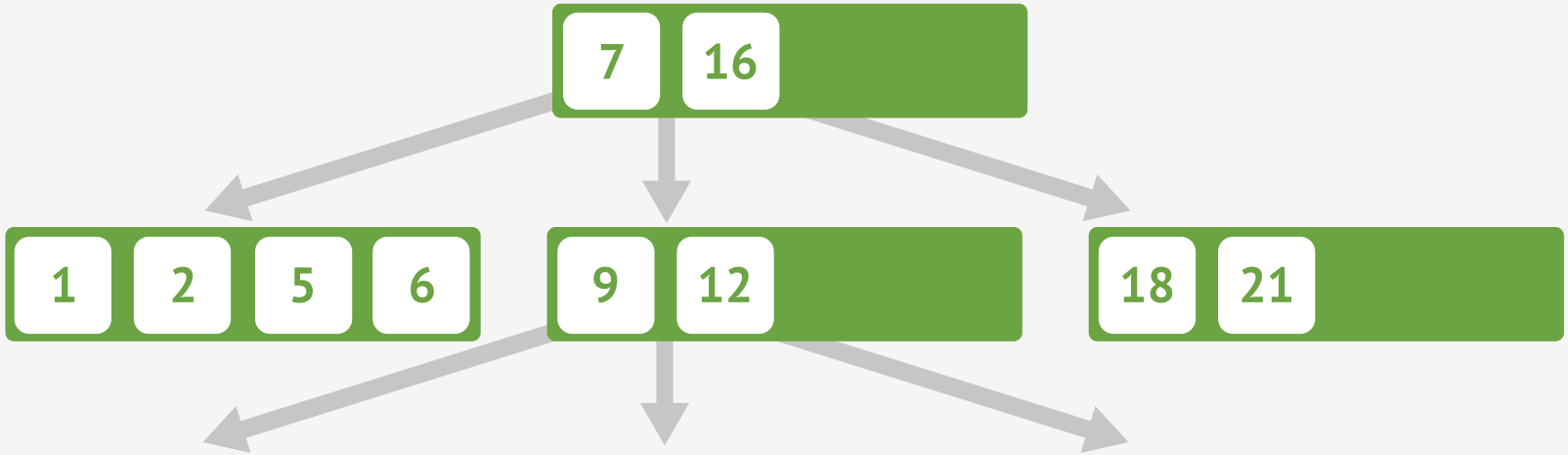
Schema Design @



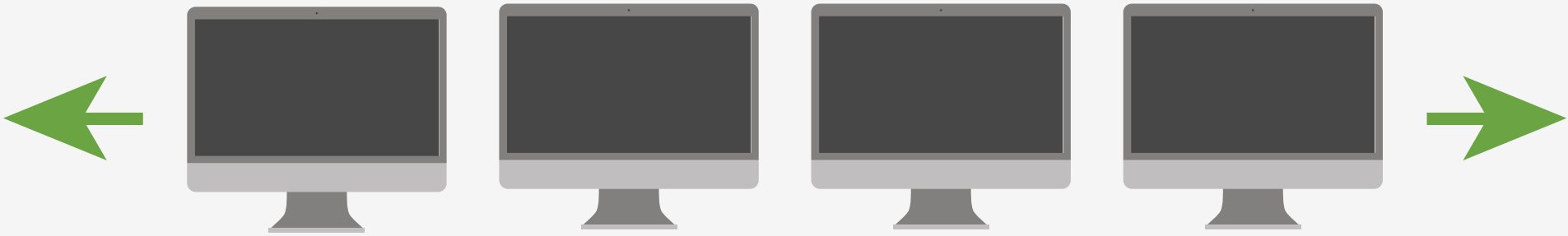
Replication @



Indexing @



Sharding @



Questions?

mongoDB

#BigData #MongoDB @prasoonk

Thank You

Prasoon Kumar

MongoDB Consultant

mongoDB