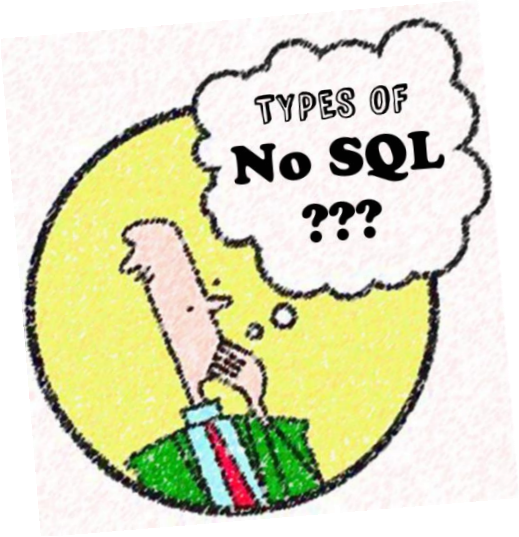
Overview over MongoDB

LSDMA Topical Meeting: Demonstrator Day
June 16, 2014

Parinaz Ameri, Jörg Meyer

NoSQL Databases

Key/Value	Column	Document	Graph
• Dynamo • Redis	• BigTable • Hbase • Cassandra	• MongoDB • CouchDB • Riak	• Neo4j • FlockDB



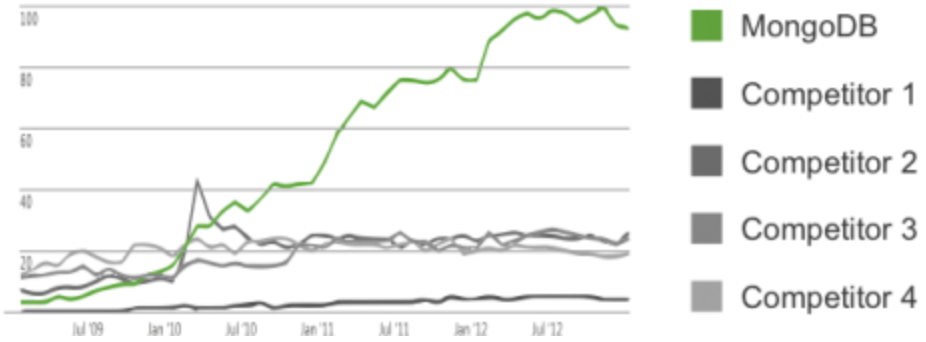
Typical Usage:

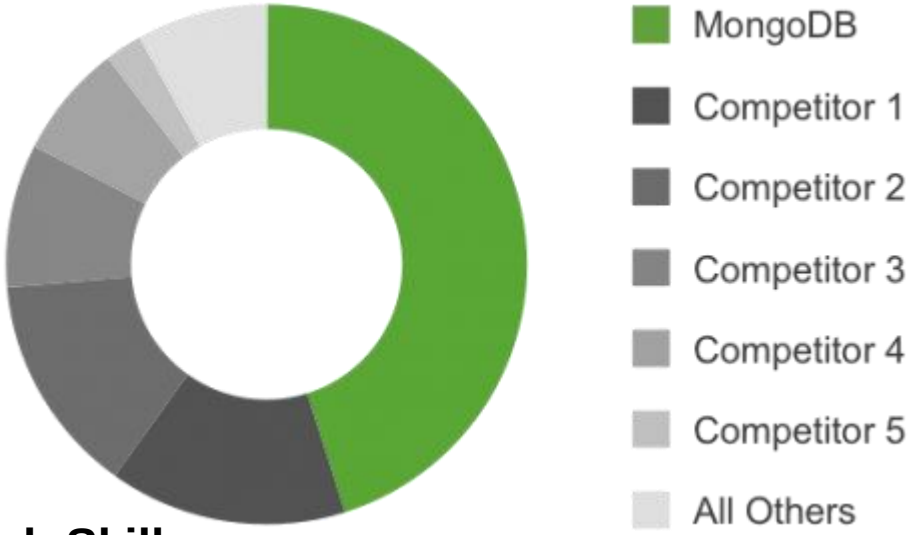
- **Key/Value**: Distributed Hash table, Caching
- **Column families**: Distributed data storage
- **Document Store**: Web applications
- **Graph databases**: Social networking, Recommendations

MongoDB (from “humongous“)

- Open source, written in C++
- Developed by company: **MongoDB Inc.** (formerly 10gen)
- First released 2009
- Last stable released version is 2.6.1 (May 2014)
- MongoDB and its drivers are available under free licenses, but commercial support also exists
- Mongo shell is an interactive JavaScript shell

Google Search





LinkedIn Job Skills

Getting started

Installation

- get repositories, packages or source code from http://www.mongodb.org

- Red Hat/Fedora/CentOS
  yum install mongodb-org-server (from repo)
  rpm –ihv mongodb-org-server-2.6.1-2.x86_64.rpm (from file)
- Ubuntu/Debian
  apt-get install mongodb-org (from repo)
  dpkg –i mongodb-org_2.6.1_amd64.deb (from file)

Configuration

- edit /etc/mongod.conf

Starting service

  service mongod start

Mongo shell

- Shell for interactive work
- JavaScript interpreter

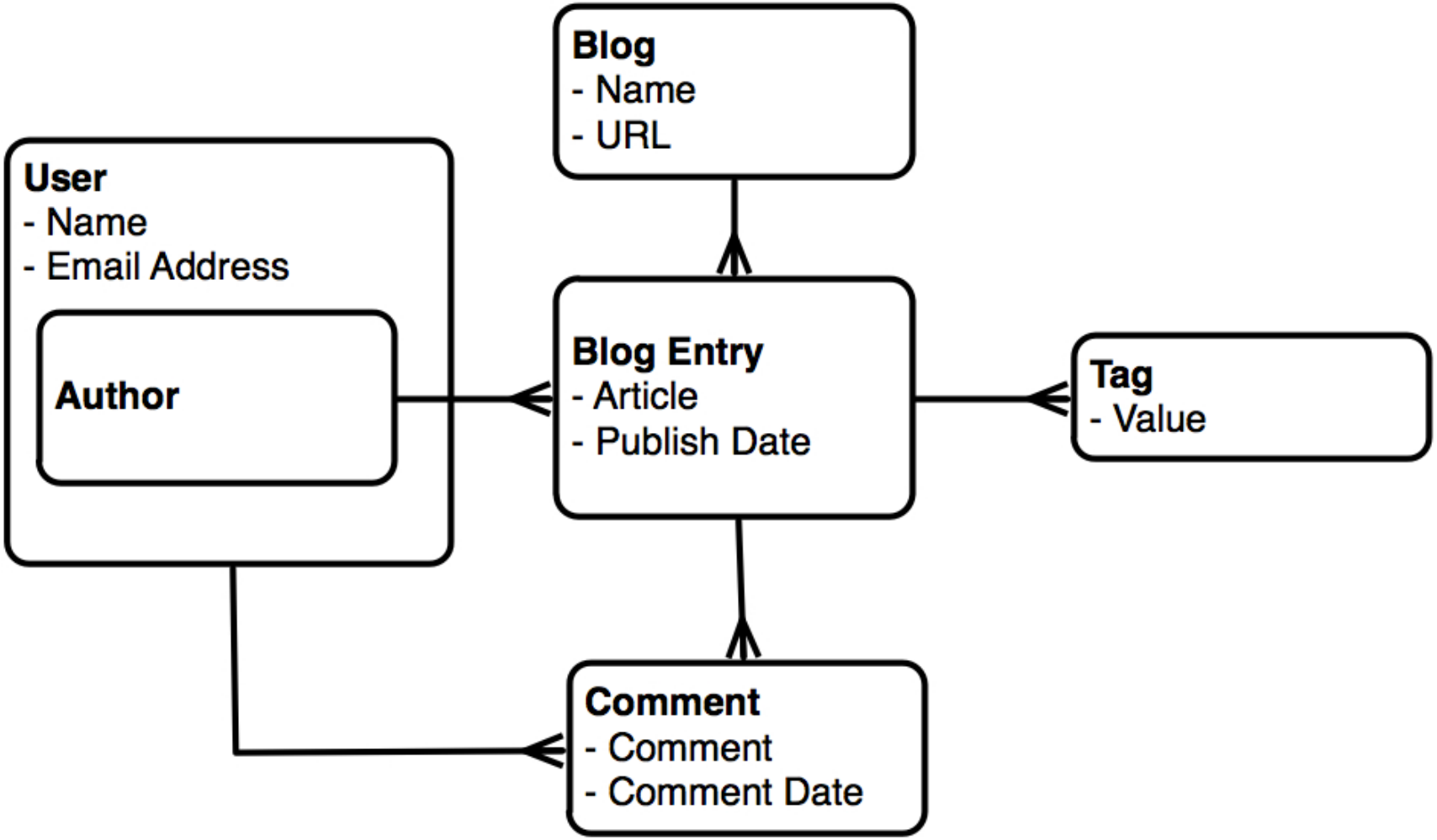
$mongo (connects to localhost:27017)
MongoDB shell version: 2.6.1
connecting to: test
>

useful commands
>show dbs
>db
>use myDB
>show collections

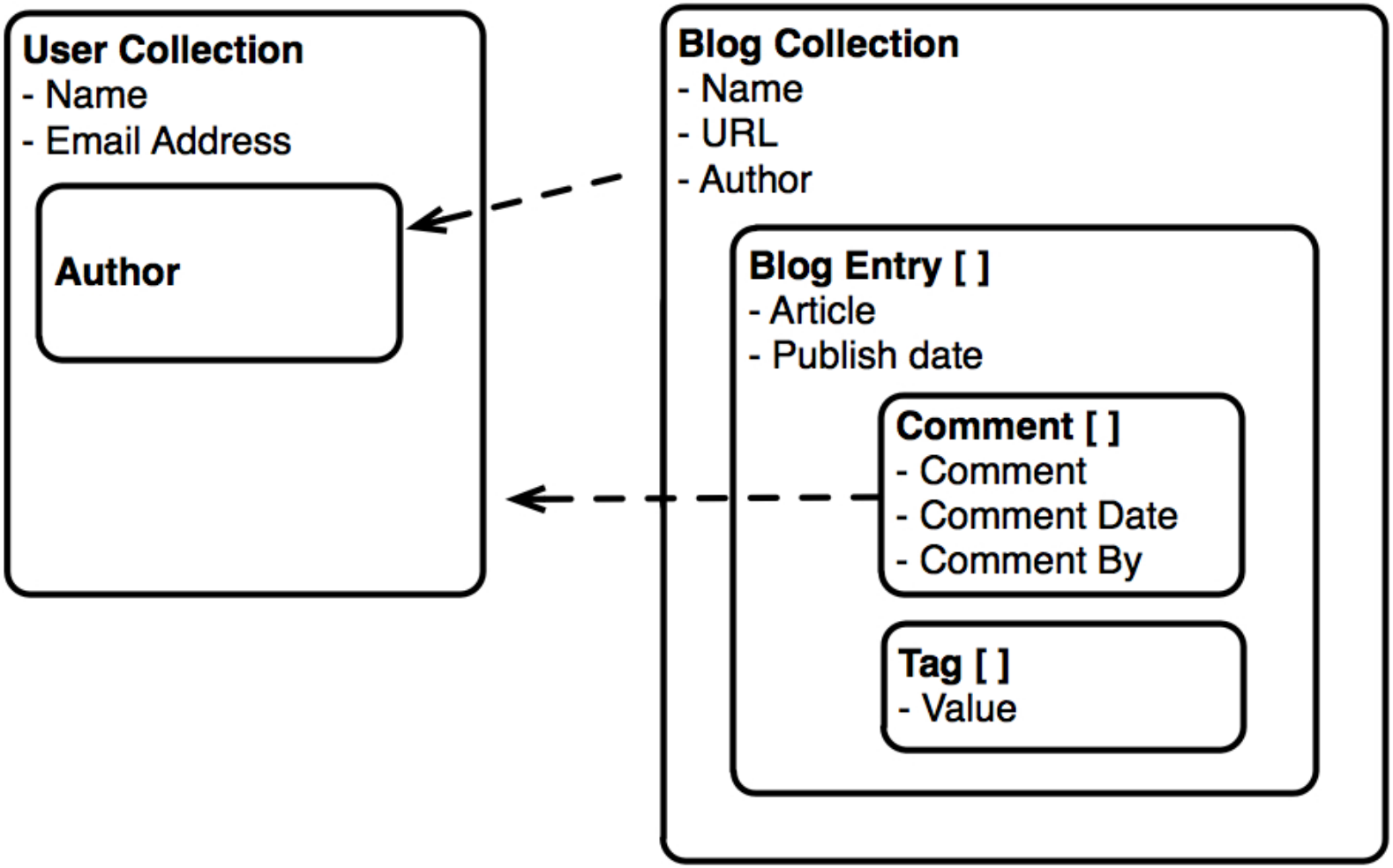
Terminology

RDBMS	MongoDB
Table	Collection
Row(s)	BSON Document
Column	Field
Index	Index
Join	Embedding & Linking
Partition	Shard
Partition Key	Shard Key

Schema Design Relational Database



Schema Design MongoDB



CREATERUD

Create operations add new records or documents to a collection in MongoDB

- *insert*
- *upsert*

Implicitly created on first insert() operation. The primary key _id is automatically added if _id field is not specified.

```
db.authors.insert({
    id:         2,
    first_name: "Arthur",
    last_name:  "Miller“,
    age: 25
    })
```

```
CREATE TABLE authors(
    id MEDIUMINT NOT NULL
        AUTO_INCREMENT,
    user_id  Varchar(30),
    first_name  char(15),
    last_name  char(15),
    age  Numer,
    PRIMARY KEY  (id)
    )
```

Indexing

Create operations add new records or documents to a collection in MongoDB

- *insert*
- *upsert*

db.authors.ensureIndex({ **id**: 1 })

CREATE INDEX idx_user_id_asc
ON users(id)

db.authors.ensureIndex({ **id**: 1, **age**: -1 })

CREATE INDEX
idx_id_asc_age_desc
ON users(id, age **DESC**)

- find
- findOne: selects and returns a single document from a collection

```
>db.authors.find({"last_name": "Miller"});

{
_id: 2,
first_name: "Arthur",
last_name: "Miller",
age: 25
}
```

```
SELECT  *
FROM  authors
WHERE  last_name = “Miller“
```

QUERY OPERATORS

- Conditional operators
  $gt, $gte, $lt, $lte, $ne, $all, $in, $nin, $size, $and, $or, $nor, $mod, $type, $exists
- Regular expressions
- Value in an array
  $elemMatch
- Cursor methods and modifiers
  count(), limit(), skip(), snapshot(), sort(), batchSize(), explain(), hint()

CRUPDATED

Update can either replace the existing document with the new document or update specific fields in the existing document.

```
review = {
    user: 1,
    text: "I did NOT like this book."
    };

db.books.update({ _id: 1},
    { $push: { reviews: review }});
```

ATOMIC MODIFIERS

Update specific fields within a document

- $set, $unset
- $push, $pushAll
- $addToSet, $pop
- $pull, $pullAll
- $rename
- $bit

CRUDELETE

The **remove()** method has the following syntax:
db.collection.remove(<query>, <justOne>)

db.authors.remove({ _id: 1})

DLETE FROM authors
WHERE id = “1"

- **remove():** removes all documents from a collection, w/o indexes.
- **drop():** drops the entire collection, including the indexes

Python driver: pymongo

```
import csv
import pymongo
from pymongo import MongoClient
import json

conMongo = MongoClient()
db = conMongo["pari"]
coll = db.about

with open("train.tsv",'rt') as csvfile:
        reader = csv.reader(csvfile, delimiter="\t", quotechar ="")
        headerCache=[]
        for row in reader:
                try:
                        if headerCache[0]:
                                coll.insert(dict(zip(headerCache, row)))
                except IndexError:
                        headerCache= row

print coll.find().count()

conMongo.disconnect()
```

Script to read data from a text file and to write it to a Mongo collection

Geospatial operations

- Documents with GeoJSON location (http://geojson.org/geojson-spec.html)

```
> db.cities.find({"city": "KARLSRUHE"})
{     "_id" : ObjectId("539abd165b5f19e448997712"),
      "city" : "KARLSRUHE",
      "loc" : { "type" : "Point", "coordinates" : [ 8.4, 49.017 ] } }
```

- Special geo index

```
db.cities.ensureIndex(loc: “2dsphere“)
```

- Geospatial queries: find all cities around Karlsruhe:

```
db. cities.find({"loc": { $geoWithin: { $center: [[8.4, 49.017 ], 0.1] }}})
```

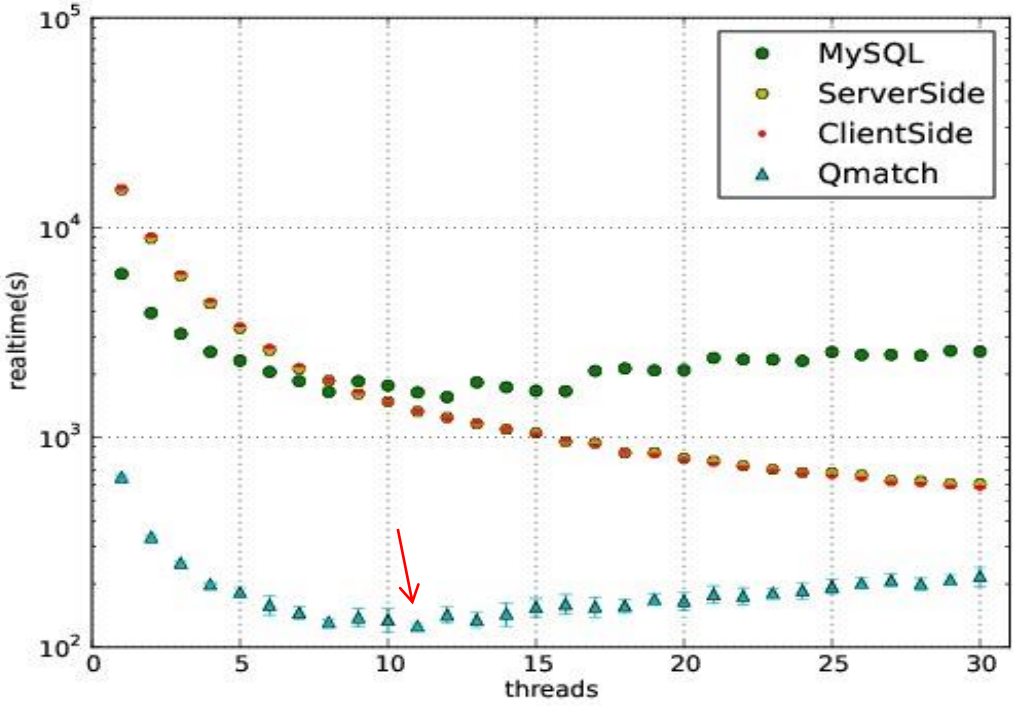
([[longitude, latitude], radius])

- http://docs.mongodb.org/manual/reference/operator/query-geospatial/

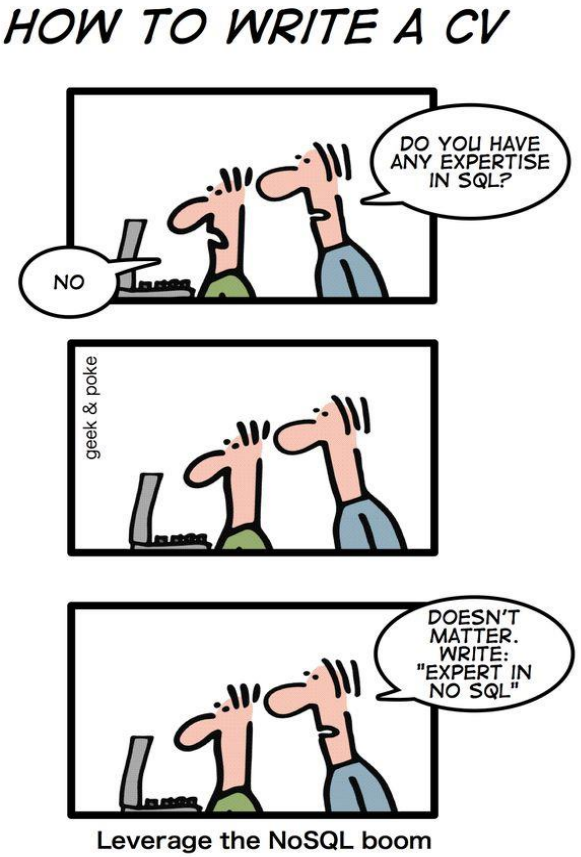
“The $geoWithin operator is a geospatial query operator that queries for a defined point, line or shape that exists entirely within another defined shape. “

Parallel threads: application in DLCL

- Collections with metadata of satellites (millions of documents)
- Matching of geolocations and time
- MongoDB scales with number of threads (bottleneck: 12 physical client CPU cores)



Questions?

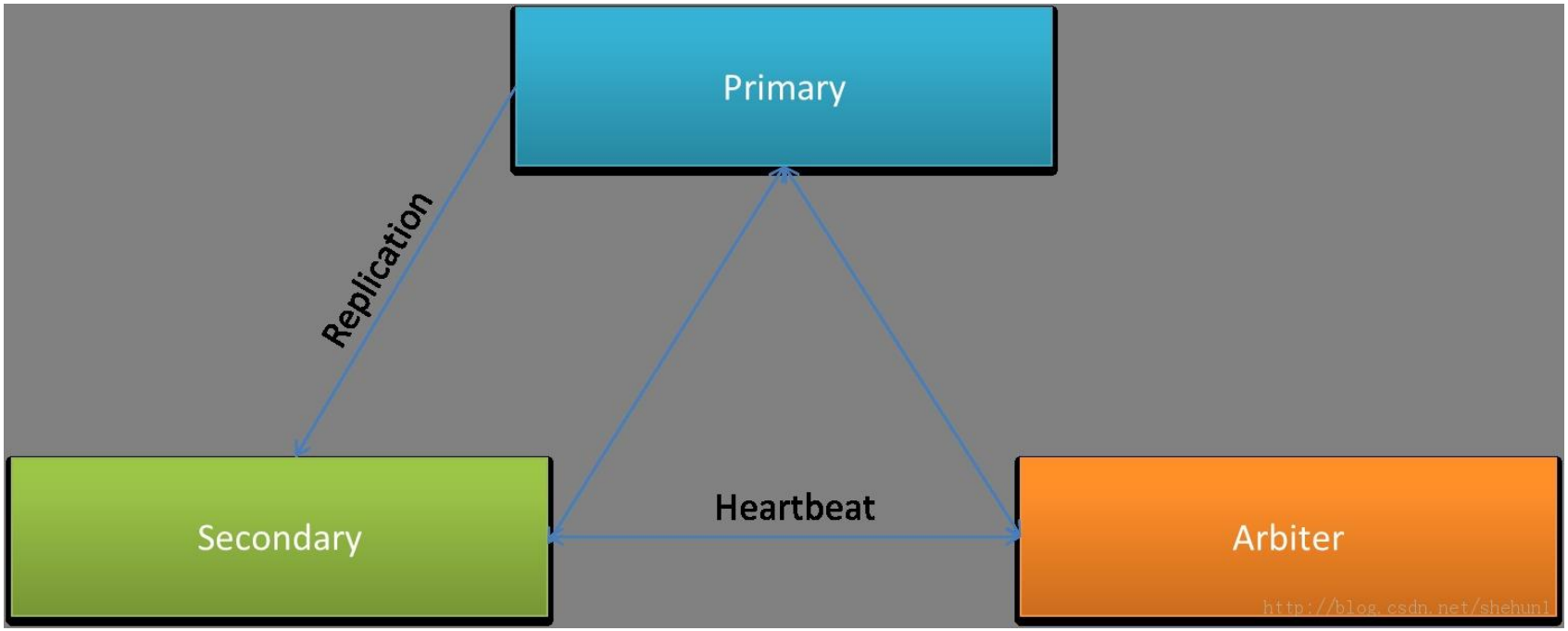

HOW TO WRITE A CV
DO YOU HAVE ANY EXPERTISE IN SQL?
NO
geek & poke
DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"
Leverage the NoSQL boom

Replication

Database replication ensures:

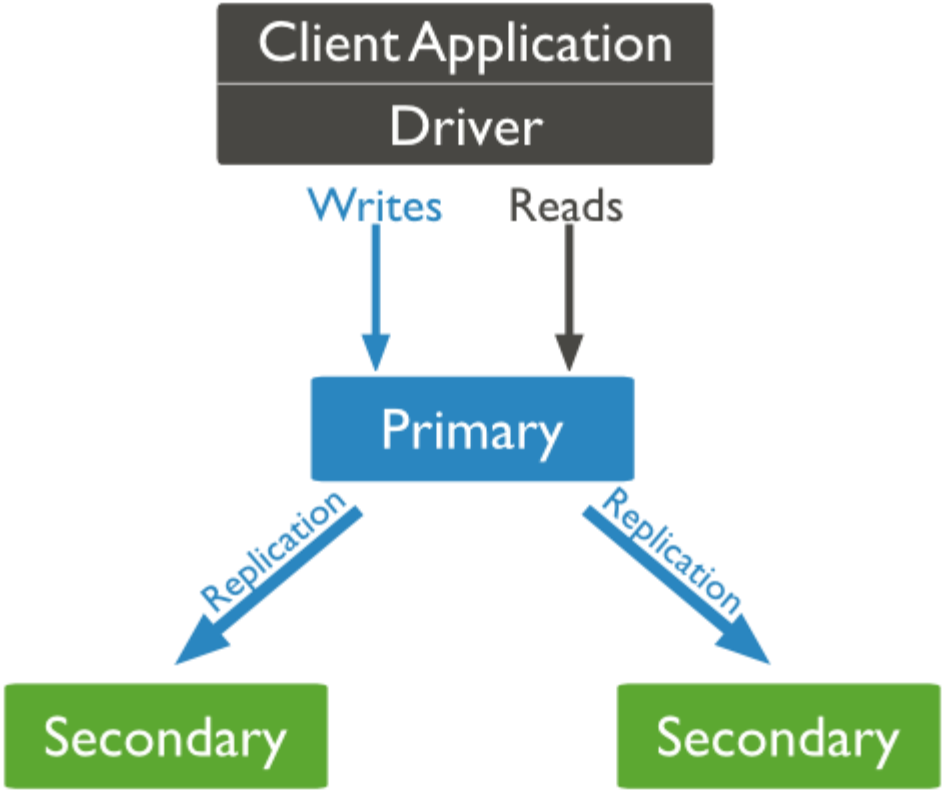
- Redundancy
- Backup
- Automatic failover.

Replication occurs through groups of servers known as replica sets.



Replica Set

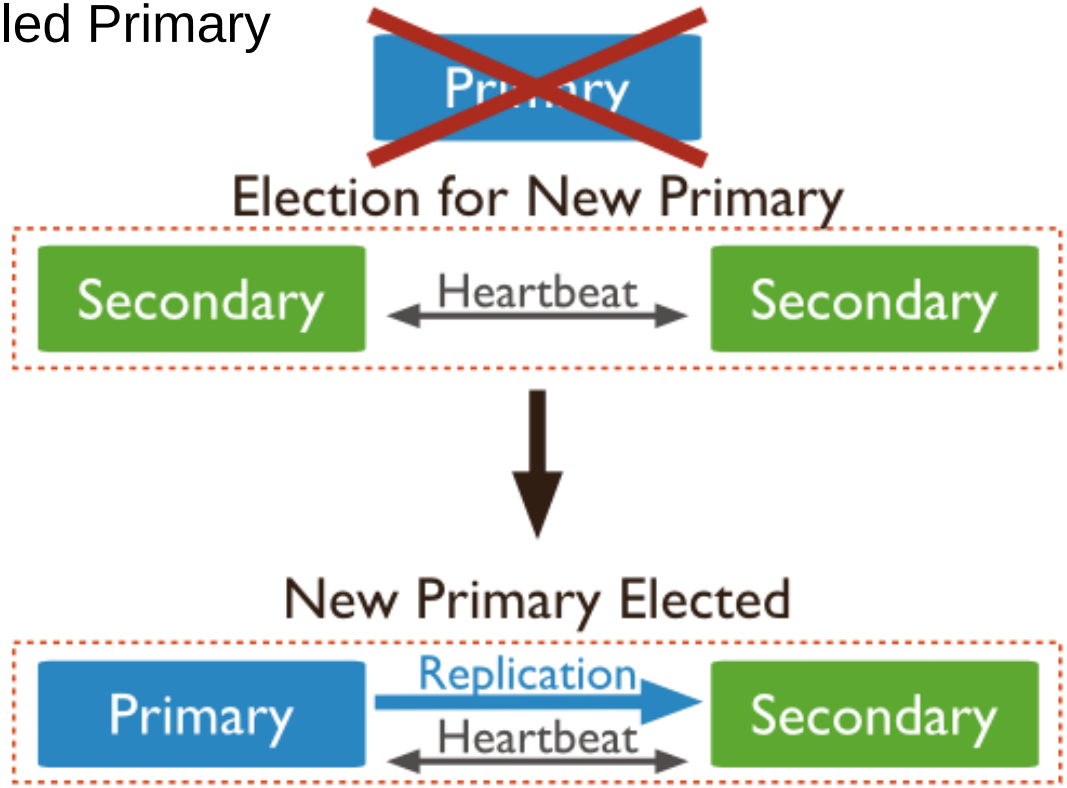
Clients direct all writes to the primary, while the secondary members replicate from the primary asynchronously.



Replica Set

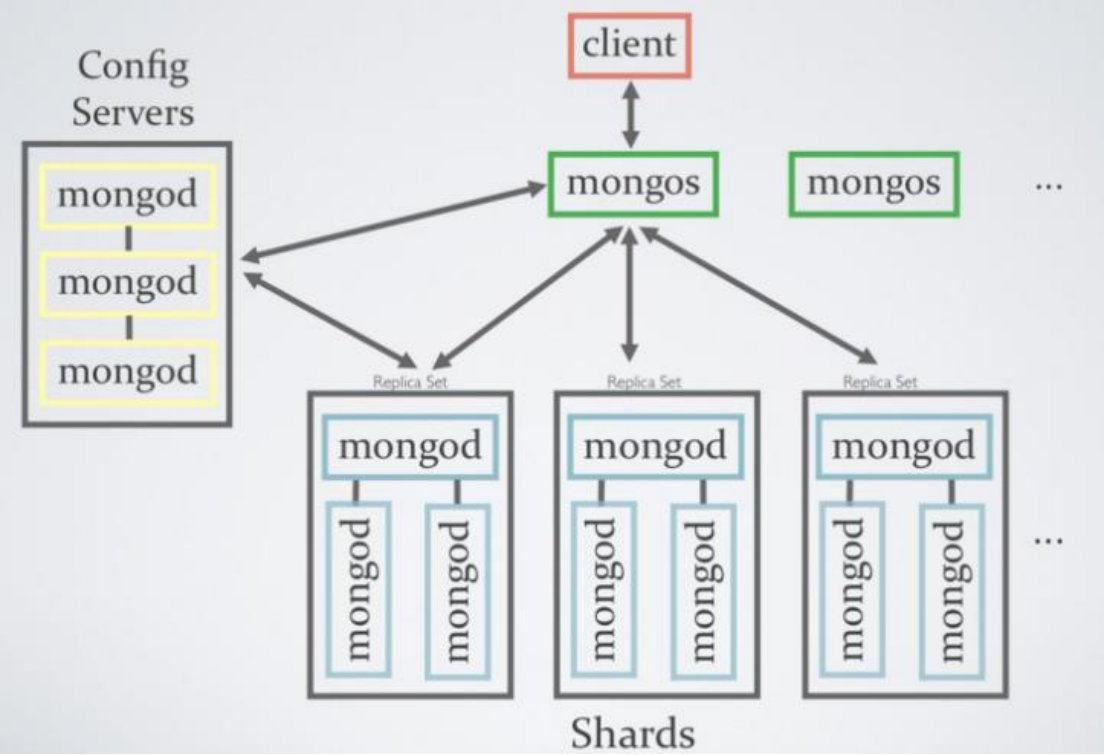
Elections of Primary

- Most up to date
- Highest priority
- Less than 10s behind failed Primary



Sharding

- Automatic partitioning and management
- Range based
- Convert to sharded system with no downtime
- Fully consistent
- No code changes required



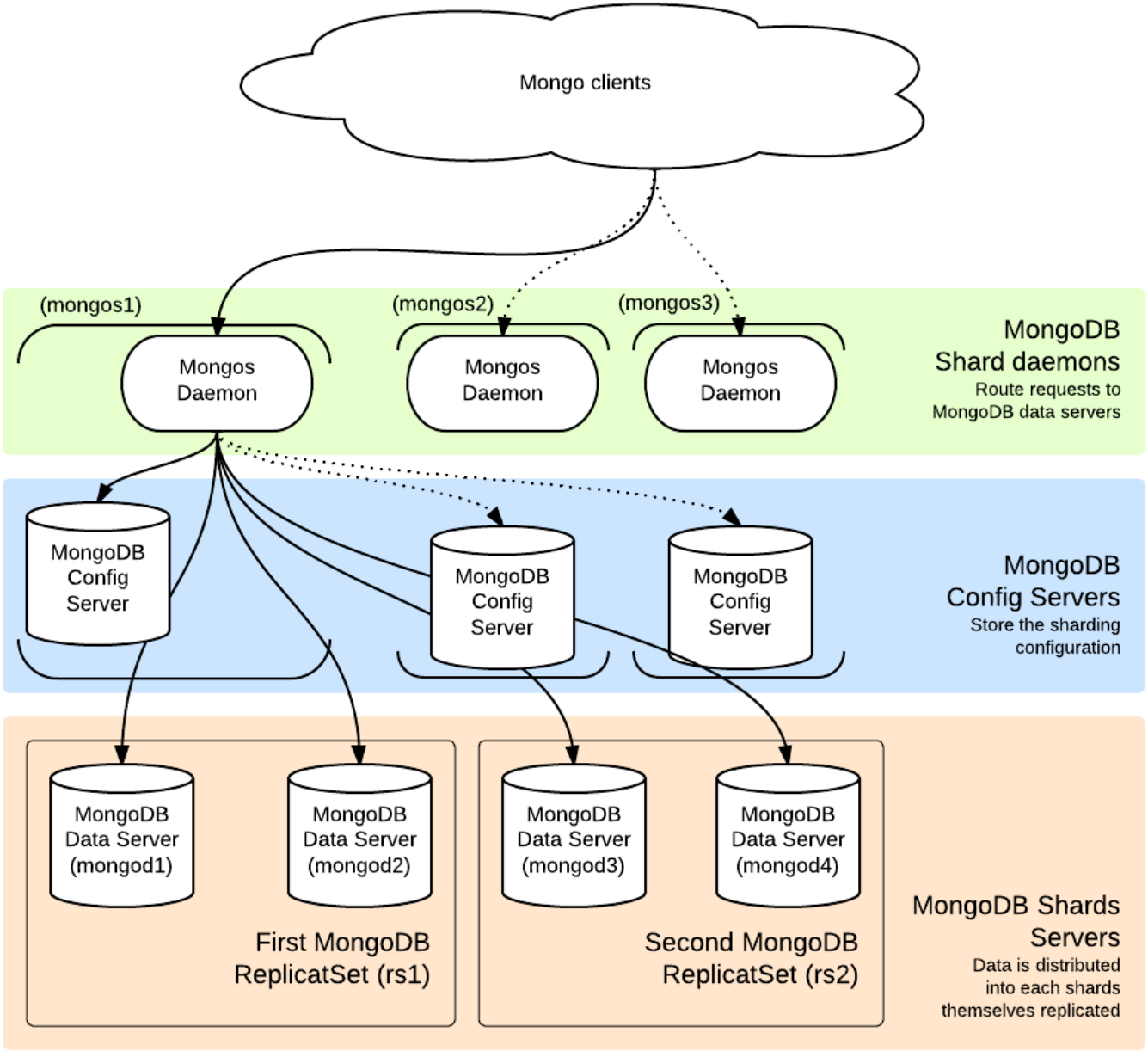
SHARDING COMPONENTS

1. Mongos: is a routing service for MongoDB shard configurations that processes queries from the application layer, and determines the location of this data in the sharded cluster.

2. Config servers: Each config server stores a complete copy of the cluster's metadata.

3. Shards:

- mongod: the primary daemon process for the MongoDB system
- Replica sets


Mongo clients
(mongos1)
(mongos2)
(mongos3)
Mongos Daemon
Mongos Daemon
Mongos Daemon
MongoDB Shard daemons
Route requests to MongoDB data servers
MongoDB Config Server
MongoDB Config Server
MongoDB Config Server
MongoDB Config Servers
Store the sharding configuration
MongoDB Data Server (mongod1)
MongoDB Data Server (mongod2)
MongoDB Data Server (mongod3)
MongoDB Data Server (mongod4)
First MongoDB ReplicatSet (rs1)
Second MongoDB ReplicatSet (rs2)
MongoDB Shards Servers
Data is distributed into each shards themselves replicated