# MongoDB

## mongoDB

## Introduction

- Is open source
- Written in C++
- Document Oriented Database
- Data Structure used are pairs of Key-Value pairs (Similar to JSON Objects)

## Key Features

## Document Oriented Storage

- Rich Data Model
- Powerful query language
- Aggregation, Text Search, Geospatial, Indexing
- Schema-less, so document can have any number of fields

## Replication and High Availability

- Just a group of mongod processes with the same data set
- Redundancy
- High availability
- Scalability

## Indexing

- Used to make read operations faster.
- The power of indexes is equally visible in any of fields that is indexed by the value of the field.
- Used to limit the number of documents that must be scanned.
- Examples:
  - Text Search: All documents in collection
  - Prefix or substring
  - Exactly fields equals

## Map Reduce

## Map Reduce

- Condenses large volumes of data into useful aggregated results
- Map phase emits key-value pairs
- Reduce phase aggregates values across same keys
- Works parallel way
  - Map function can be parallelized by document
  - Reduce function can be parallelized by key

## Who uses it ?

Examples of Internet Businesses:

- 10gen
- bit.ly
- Foursquare
- SOURCE FORGE
- ARE WE?

Traditional enterprises:

- Telecom
- Financial services
- Government
- Healthcare
- Media
- Retail
- Manufacturing
- Insurance
- Pharmaceuticals
- Energy

AND MANY MANY MORE ...

## Update

```
db.collection.update(
    <query>,
    <update>,
    {
        upsert : <boolean>,
        multi : <boolean>
    }
)
```

## Read

The result then a find is a cursor whose iteration is object-at-a-time.

```
db.collection_name.find([criteria, projection])

where,
    criteria = { } or null - gives all the results
    projection = fields to return

SORT, OR SKIP BY or HOW
db.collection_name.find(<criteria>, [projection], [skip], [limit], [sort])
```

## Create

```
db.mycollection.insert(<document_or_array_of_documents>)

OR

db.collection_name.save(<document>)

db.mycollection.insert({name:"Jack"})
db.mycollection.insert({name:"Jill"})
db.mycollection.find()
{ "_id" : ObjectId("4d9f0a0a0a0a0a0a0a0a0a0a"), "name" : "Jack" }
{ "_id" : ObjectId("4d9f0a0a0a0a0a0a0a0a0a0b"), "name" : "Jill" }
```

## Basic CRUD Commands

## Seeing it in Action

- MongoDB Server :
  - mongod --help
- MongoDB Client :
  - mongo --help

## JSON

## Installation

- Goto the location : /Users/Downloads
- Run the setup script : ./install.sh

## SQL – Mongo Mapping

| SQL | MongoDB |
|---|---|
| Database | Database |
| Table | Collection |
| Row | Document |
| Index | Index |
| Table Joins | Not Supported |
| Primary Key | Primary Key (Default key _id provided by mongodb itself) |
| Aggregation (e.g. "select count(*), ...") | Map reduce or aggregation |
| Transaction (e.g. "lock table ...") | Not Supported |

## Delete

Remove all documents in a collection :

```
db.collection.remove()
```

Remove documents matching a query :

```
db.collection.remove(query)
```

Limiting the deletion to just one document :

```
db.collection.remove(query, <justOneOptionalBoolean>)
```

## Grouping

```
db.collection.group(
    {
        key,
        initial,
        reduce,
        [cond],
        [finalize]
    }
)
```

## Some other Commands ...

| | |
|---|---|
| Rename a collection | db.collection.renameCollection() |
| Find a collection | db.getCollectionInfos() |
| Find the list of existing collections | db.collection.getIndexes() |
| Drop a collection | db.collection.drop() |
| Drop the database | db.dropDatabase() |
| Find the database | db.getMongo().getDB() |
| Monitor field updates | db.collection.watch() |
| Replace existing document | db.col.save(newdoc) |
| Explain an existing query | db.collection.find().explain() |
| Find a list of collections in a database | db.collection.getCollections() |

- Alternative ways to find a document
  - Find one document in collection : db.collection.findOne()
  - Find a document by matching criteria : db.collection.find(criteria)
  - Find a document by matching criteria and limit the number of results : db.collection.find(criteria).limit(n)
  - Find a document by matching criteria and skip the first n results : db.collection.find(criteria).skip(n)
  - Find a document by matching criteria and sort the results by a field : db.collection.find(criteria).sort({field:1})
  - Find a document by matching criteria and count the number of results : db.collection.find(criteria).count()
- Array operations
  - Find a document by matching an element in an array : db.collection.find({array:"element"})
  - Find a document by matching a range of elements in an array : db.collection.find({array:{$gt:n, $lt:m}})
  - Find a document by matching a specific element in an array : db.collection.find({array.n:"element"})
  - Find a document by matching a specific element in an array and return only that element : db.collection.find({array.n:"element"}, {array.n:1})
  - Find a document by matching a specific element in an array and return all elements except that element : db.collection.find({array.n:"element"}, {array.n:0})
  - Find a document by matching a specific element in an array and return the first element in the array : db.collection.find({array:"element"}, {array:1})
  - Find a document by matching a specific element in an array and return the last element in the array : db.collection.find({array:"element"}, {array:-1})
  - Find a document by matching a specific element in an array and return the first n elements in the array : db.collection.find({array:"element"}, {array:n})
  - Find a document by matching a specific element in an array and return the last n elements in the array : db.collection.find({array:"element"}, {array:-n})

## Replica Sets

Process of replicating data across multiple servers.

You can have nodes dedicated to reporting, backup, or disaster recovery.

Advantages:

- Redundancy and increased data availability
- No single point of failure
- Recovery is quick and simple

## How they work ?

## Learning MongoDB?

- www.mongodb.org
- Play around with online tutorials and queries in the mongo shell
- The Little MongoDB Book - Karl Seguin
- Google Groups :
  http://groups.google.com/forum/#!forum/mongodb-user

Prezi

# mongoDB

Nikita Gupta
ThoughtWorks Inc.

Prezi

# Introduction

- Is open source
- Written in C++
- Document Oriented Database
- Data Structure used comprises of Key-Value pairs (Similar to JSON objects)

Prezi

# Key Features

Prezi

# Document Oriented Storage

- No enforced structure
- No specific schema needed.
- Easily map real life objects to documents.
- Atomicity of write operations since the structure is denormalized.

Prezi

# Replication and High Availability

- Just a group of *mongod* processes with the same data set.
- Redundancy
- High availability
- Scalability

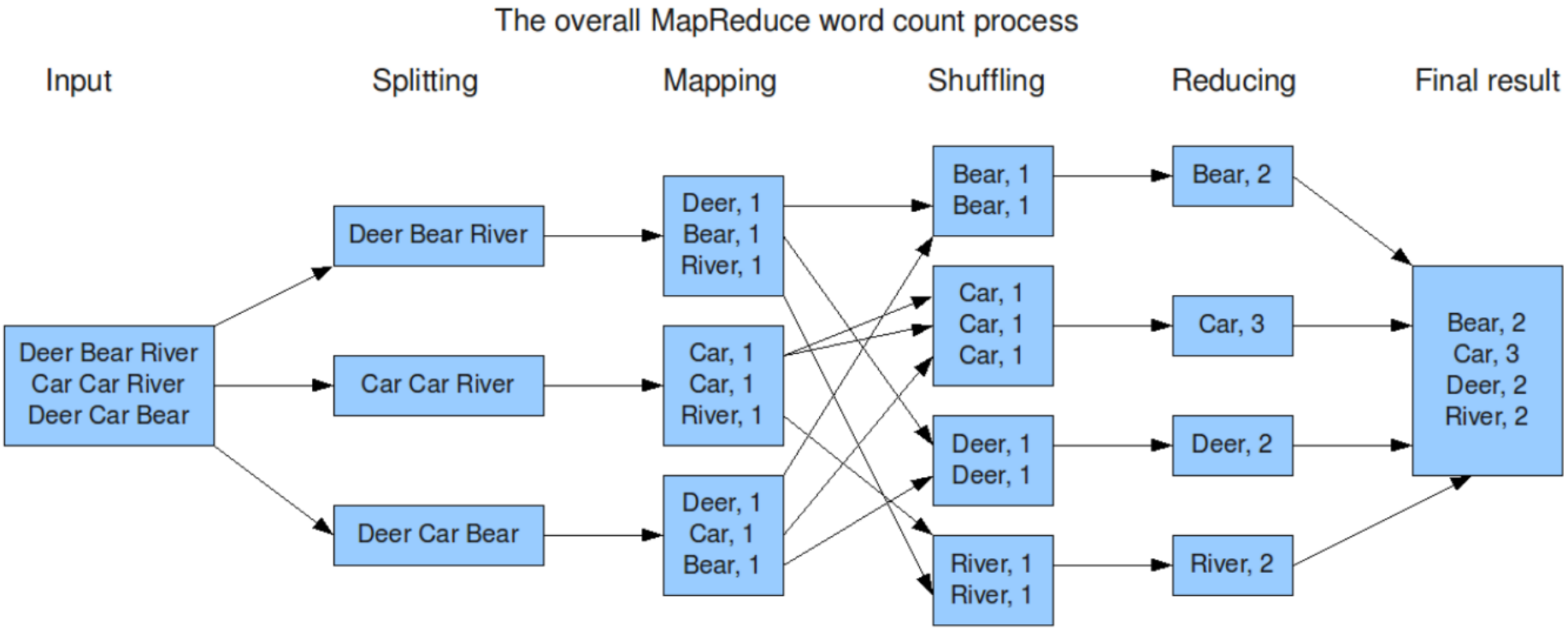
Prezi

# Indexing

- Used to make read queries faster.
- They store the value of a specific field or a set of fields that ordered by the value of the field.
- Used to limit the number of documents that must be inspected.
- Examples :
    - Default is *_id* (Also known as ObjectID).
    - Fields in a document
    - Multiple fields together.

Prezi

# Map Reduce


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

# Map Reduce

- Condenses large volumes of data into useful aggregated results.
- Map phase emits key-value pairs
- Reduce phase aggregates values across same keys.
- Allows parallelizing:
    - Map function can be parallelized by document
    - Reduce function can be parallelized by key

Prezi

# Who uses it ?

**Storage $ Content Management**

**Reporting / analytics**

**any many many more .....**

Prezi

# SQL - Mongo Mapping

| SQL | Mongo |
|---|---|
| Database | Database |
| Table | Collection |
| Row | Document |
| Column | Field |
| Index | Index |
| Table Joins | Embedded document and linking |
| Primary key | Primary key ( _id field by default ) |
| | |
| Server - mysqld | Server - mongod |
| Client - mysql | Client - mongo |

Prezi

# Installation

- Goto the location: /Nosql/mongo
- Run the setup script: ./install.sh

Prezi

# JSON

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

Object

Array

Prezi

# Seeing it in Action

- MongoDB Server :
  - *mongod --help*
- MongoDB Client :
  - *mongo --help*

Prezi

# Basic CRUD Commands

Prezi

# Create

db.createCollection(name, options)

OR

db.*collection_name*.insert(*documents*)

*documents* - single or array of documents

Prezi

# Read

The result from a find is a cursor whose execution is delayed until needed.

db.collection_name.find ( conditions , projection )

where,

conditions = {} or null , gives all the results

projection = fields to retrieve

**SORT (ORDER BY in SQL)**

db.collection_name.find ( conditions , projection ).sort({field : (1/-1) })

Prezi

# Update

```
db.collection.update(
        <query>,
        <update>,
        {
            upsert : <boolean>,
            multi  : <boolean>
        }
    )
```

Prezi

# Delete

**Remove all documents in a collection :**

db.collection.remove ()

**Remove documents matching a query :**

db.collection.remove(<query>)

**Limiting the deletion to just one document :**

db.collection.remove (<query> , <justOne(true/false)>)

Prezi

# Grouping

```
db.collection.group(
        {
            key,
            reduce,
            initial,
            [keyf,]
            [cond,]
            finalize
        }
    )
```

Prezi

# Some other Commands ...

- **Rename a collection** : db.collection.renameCollection()
- **Find and Modify** : db.collection.findAndModify()
- **Find the 1st matching document** : db.collection.findOne()
- **Number of documents** : db.collection.count()
- **Drop the collection** : db.collection.drop()
- **Distinct field values** : db.collection.distinct()
- **Update existing/ insert new** : db.collection.save()
- **Total size of collection & indexes** : db.collection.totalSize()

- **Clone existing database** : db.cloneDatabase()
- **Remove current database** : db.dropDatabase()
- **List common methods** : db.help()
- **List commands** : db.listCommands()
- **Shut the server** : db.shutdownServer()

Prezi

# Replica Sets

Process of synchronizing data across multiple servers.

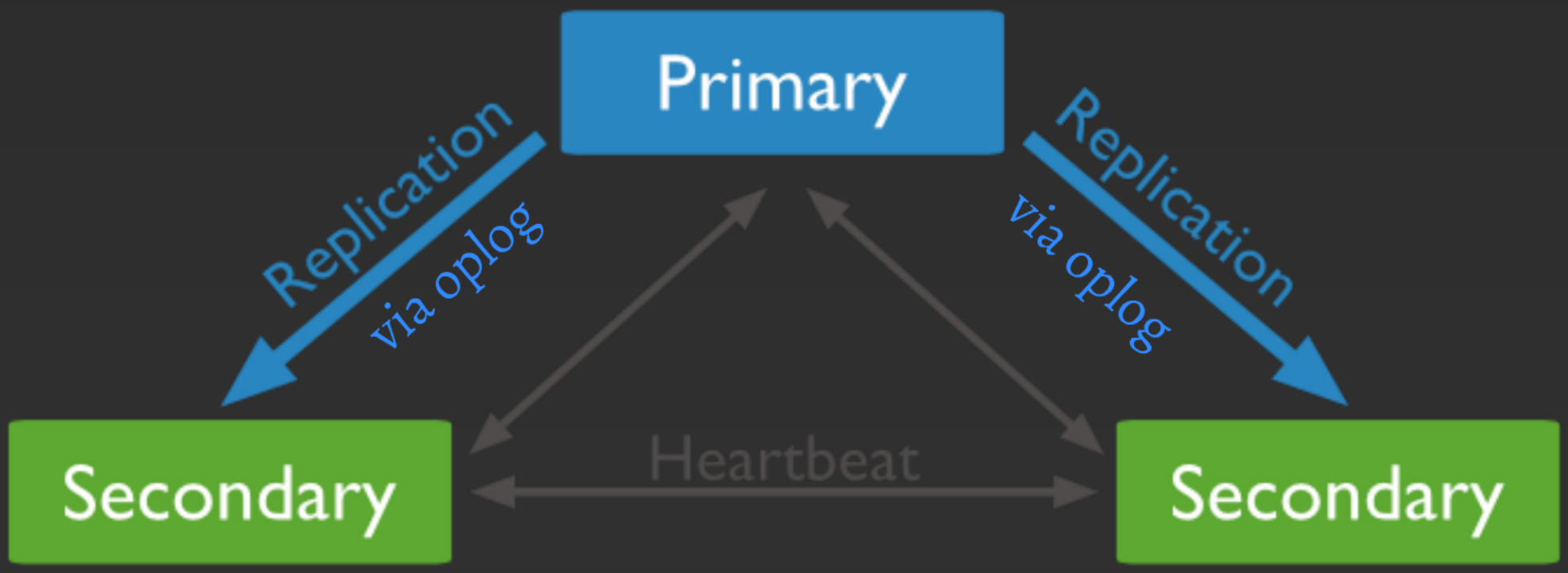
You can have nodes dedicated to reporting, backup, or disaster recovery.

Advantages :

- Redundancy and increased data availability
- No single server losses
- Recovery is quicker and simpler

Prezi

# How they work ?




Prezi

# Learning MongoDB?

- www.mongodb.org
- Play around with commands and queries in the mongo shell
- The Little MongoDB book - Karl Seguin
- Google Group :
  https://groups.google.com/forum/#!forum/mongodb-user

Prezi