Haizi Zheng

Meteor开发技术

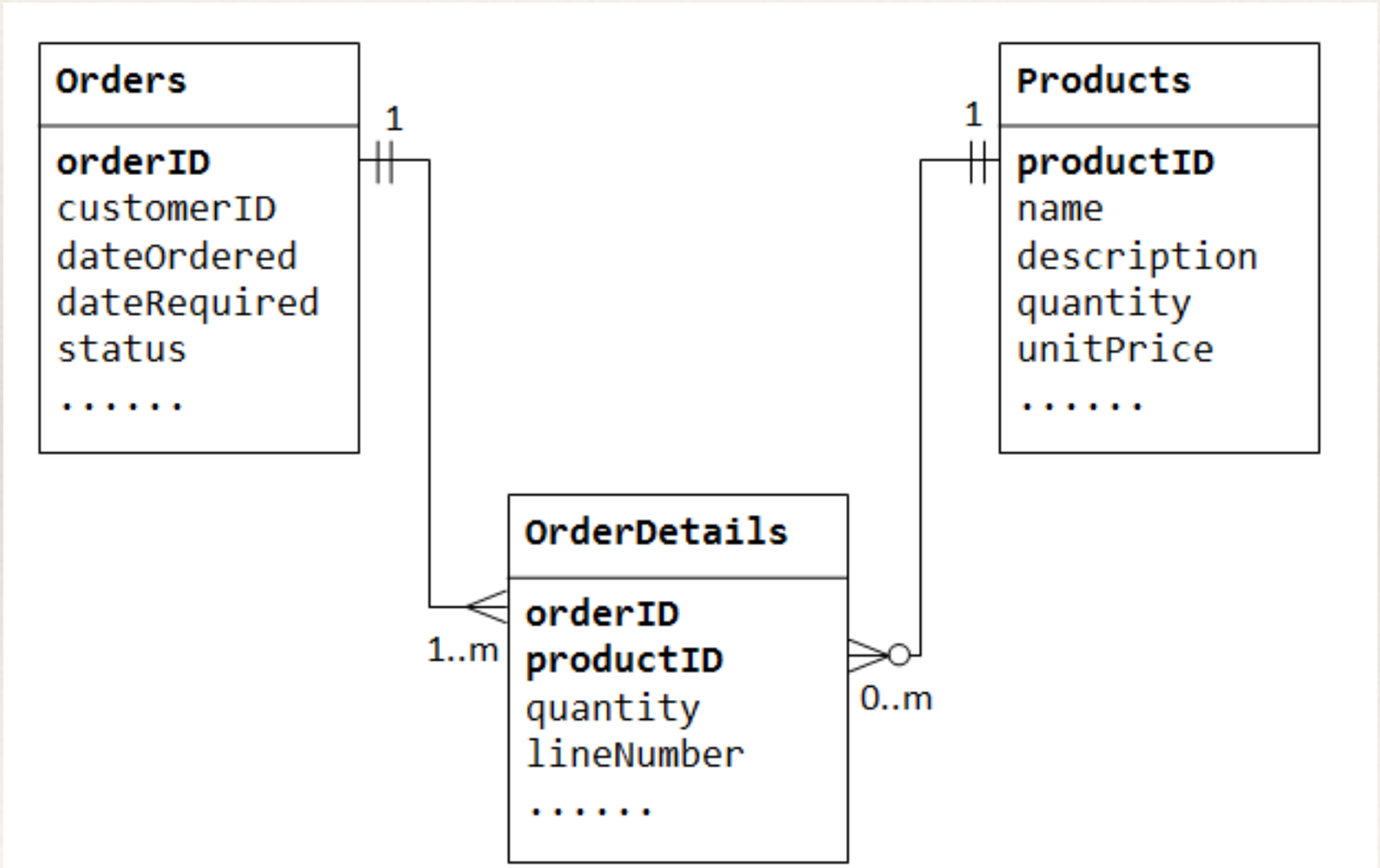
MongoDB

Outline

- MongoDB for the Modern Web
- MongoDB in Javascript: shell
- The Modelling
- Query: getting started
- Updates and deletes
- Query in advance: aggregation
- Replica set and the oplog

MongoDB for the Modern Web

关系型数据库一览



MongoDB for the Modern Web

关系型数据库的困境

我们得到了什么?

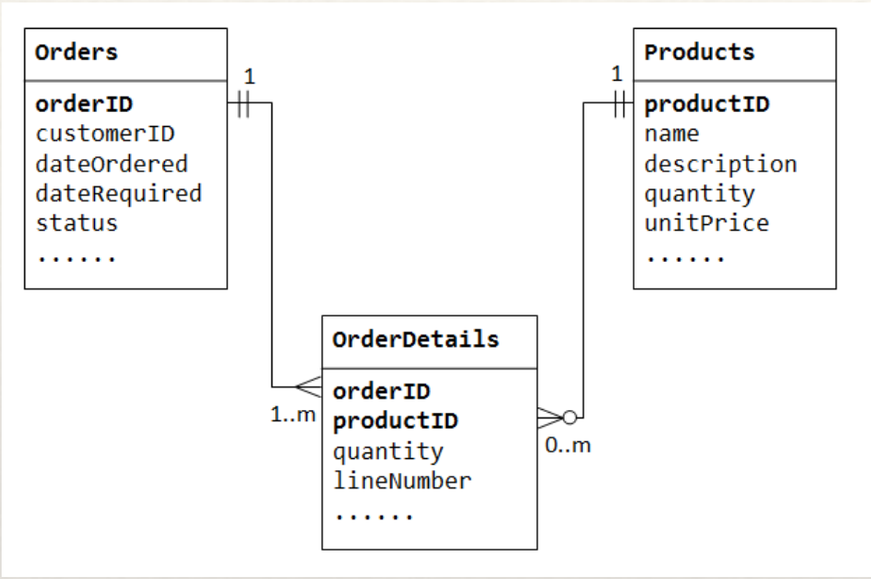
- 完备的事务实现(transaction)
- 定义良好的数据模型(schema)
- 强的数据一致性

我们失去了什么?

- 性能(performance)
- 高扩展性(high scalability)
- 高可用性(high availability)

MongoDB for the Modern Web

面向文档的数据库



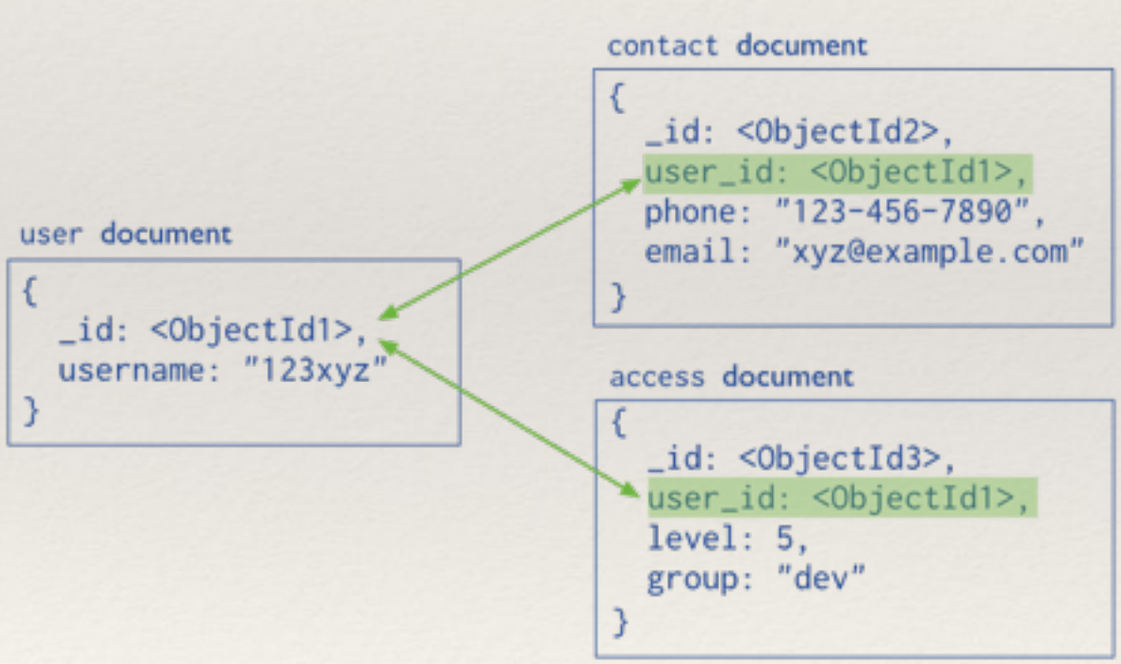

```
{
  "orderID": 1002582,
  "customerID": 823874,
  "dateOrdered": "1987-11-28T21:48:23.283Z",
  "dateRequired": "1987-11-29T11:33:21.892Z",
  "product": {
    "productID": "ASD2283941",
    "name": "Product name",
    "description": "",
    "price": 9283.89
  }
}
```

MongoDB Modelling

Data model

基于reference

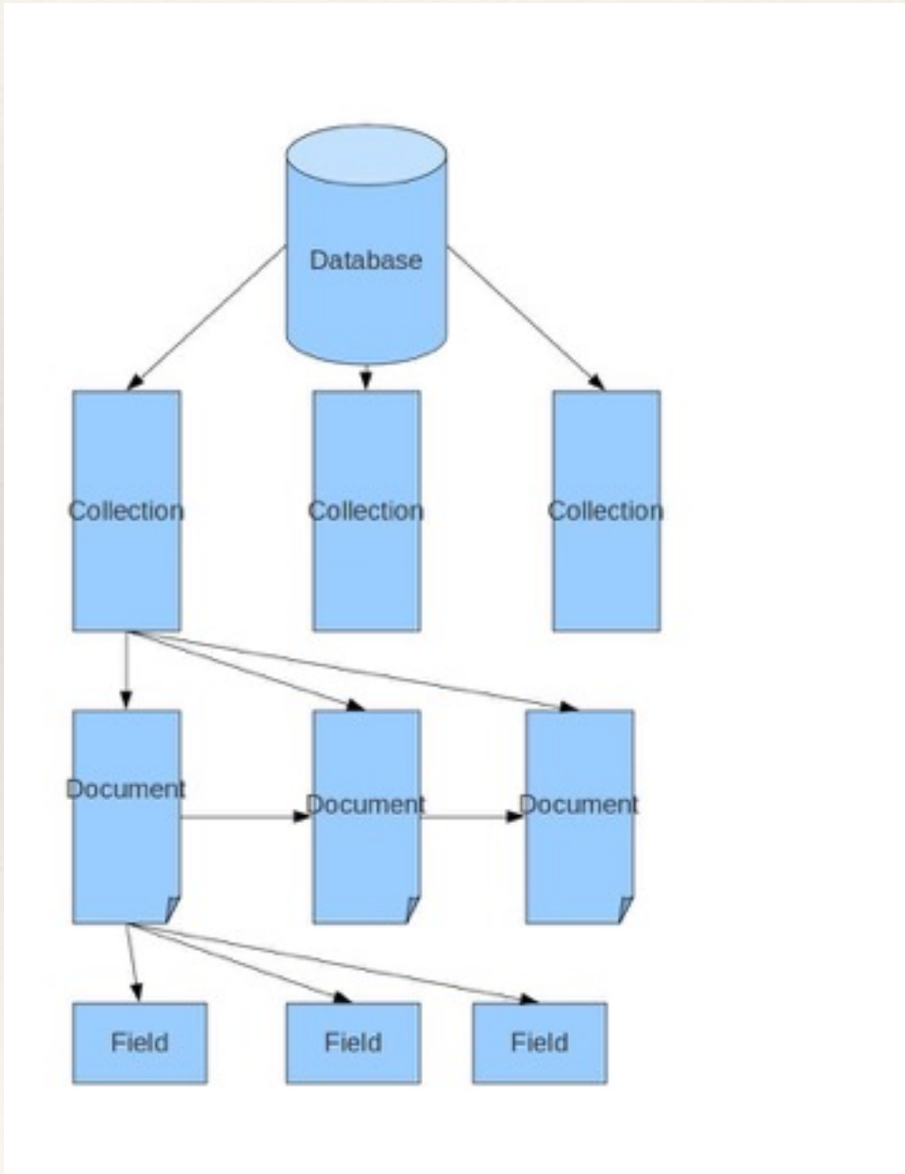
基于文档嵌入



MongoDB Modelling

和SQL的类比

database -> database
collection -> table
document -> row
key -> field



MongoDB for the Modern Web

那些令人激动的新特性

- 面向文档(document-oriented)
- schema-free
- 基于Javascript的查询语言
- 内置的高效聚合框架，甚至MapReduce
- 地理空间索引(geospatial indexing)
- 复制集(replica set)
- 高度自动化的分片机制(sharding)
- GridFS
- 其它种种

MongoDB CRUD

简单query语法

- 基本操作符
  - 数量操作符：$gt, $gte, $lt, $lte, $ne等
  - 逻辑操作符：$not, $or等
  - 修饰操作符：$size等
- 查找内嵌文档(dot notation)
- 限定结果集的返回字段
- 限定结果集数量
- 排序
- 使用$where字句

MongoDB CRUD

Update语法

- field operator
  - $set
  - $unset
  - $inc
- array operator
  - $addToSet
  - $pop
  - $push
  - $pull
- modifiers
  - $each

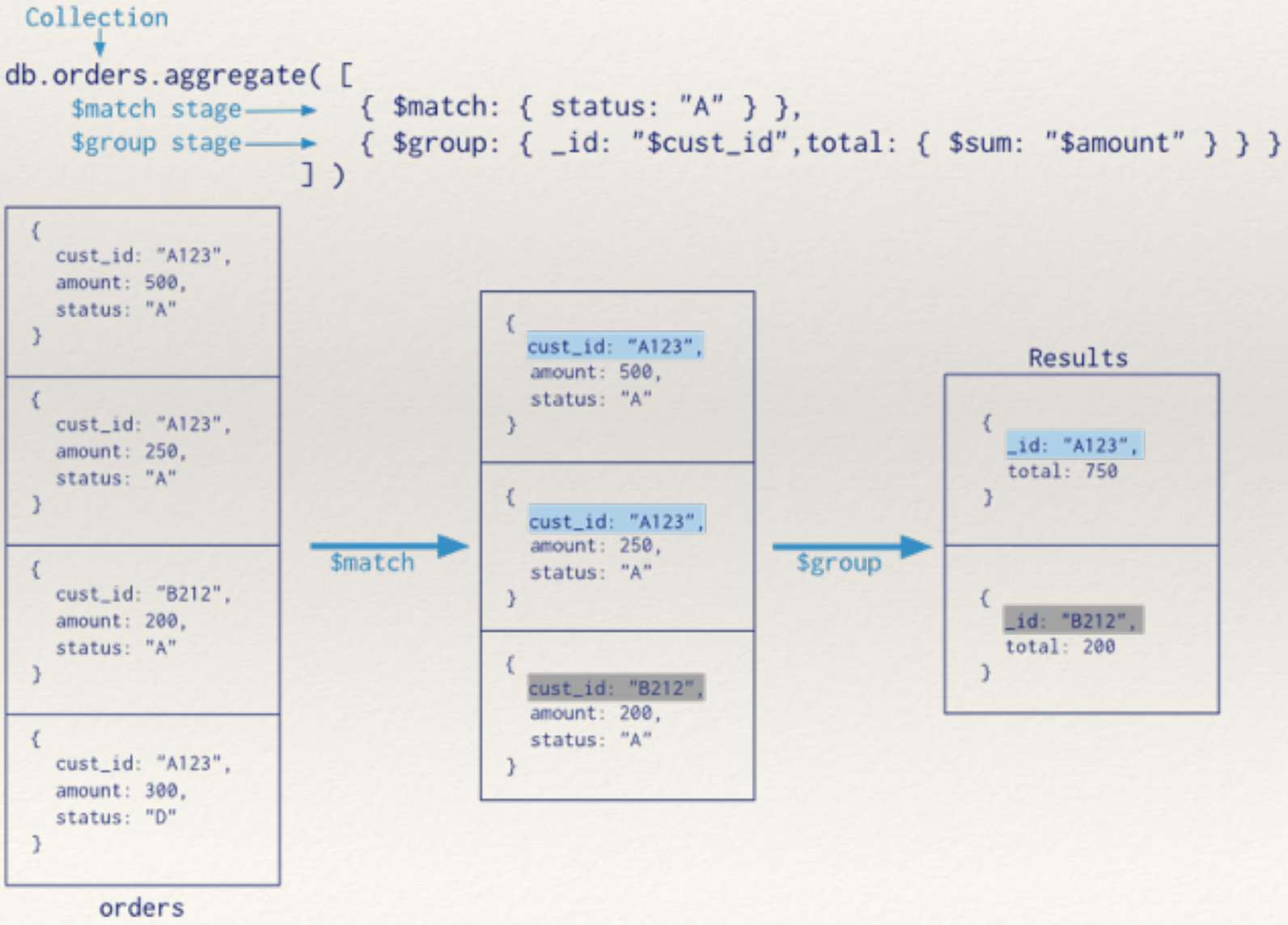
MongoDB Aggregation

什么是Aggregation?

- 统计数据的平均值
- 统计数据的最大值
- 将数据做类似GROUPBY的操作
- 其它一般性的降维变换

MongoDB Aggregation

Aggregation Framework

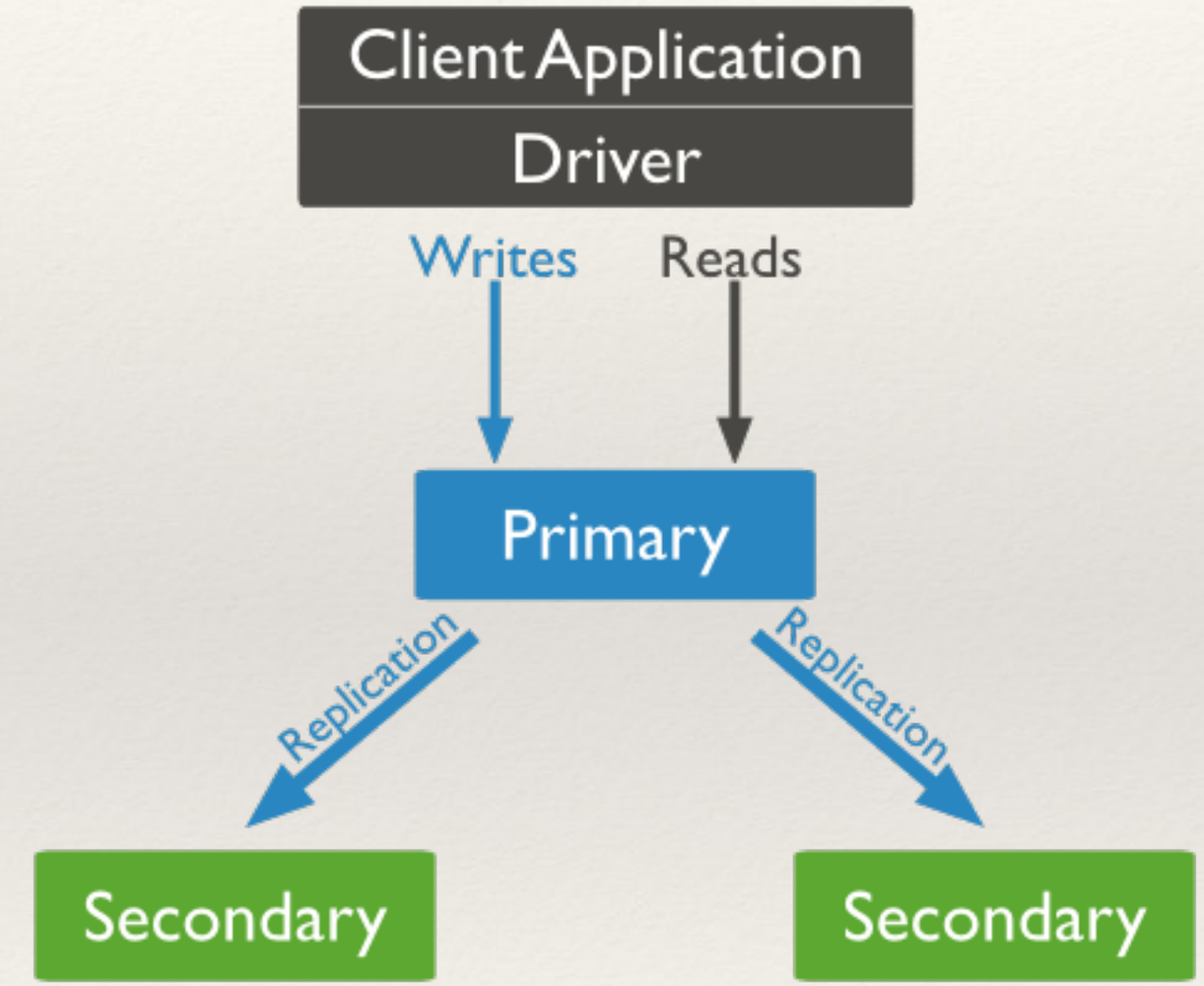


MongoDB CRUD

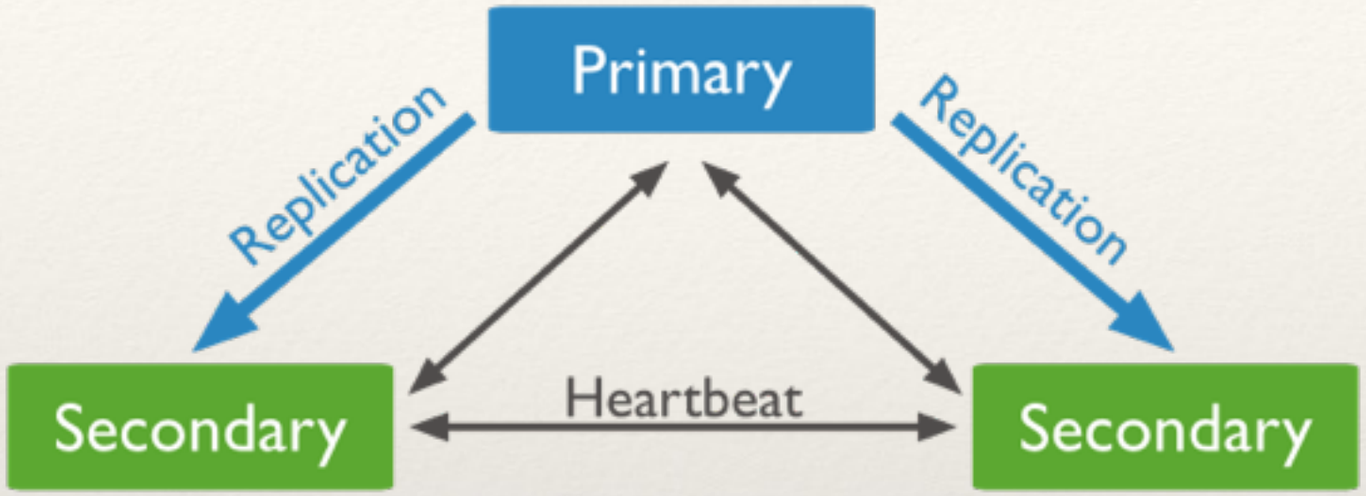
Aggregation练习

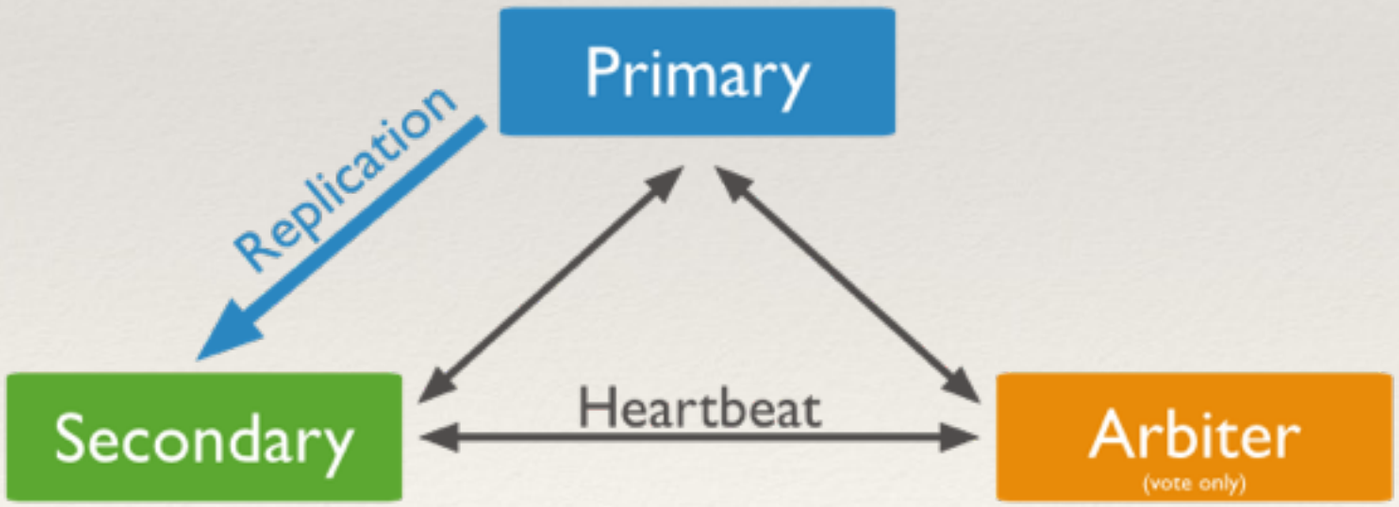
- 统计数据的平均值
- 统计数量
- 统计数据的最大值
- 将数据做类似GROUPBY的操作
- 其它一般性的降维变换

MongoDB Replica Set

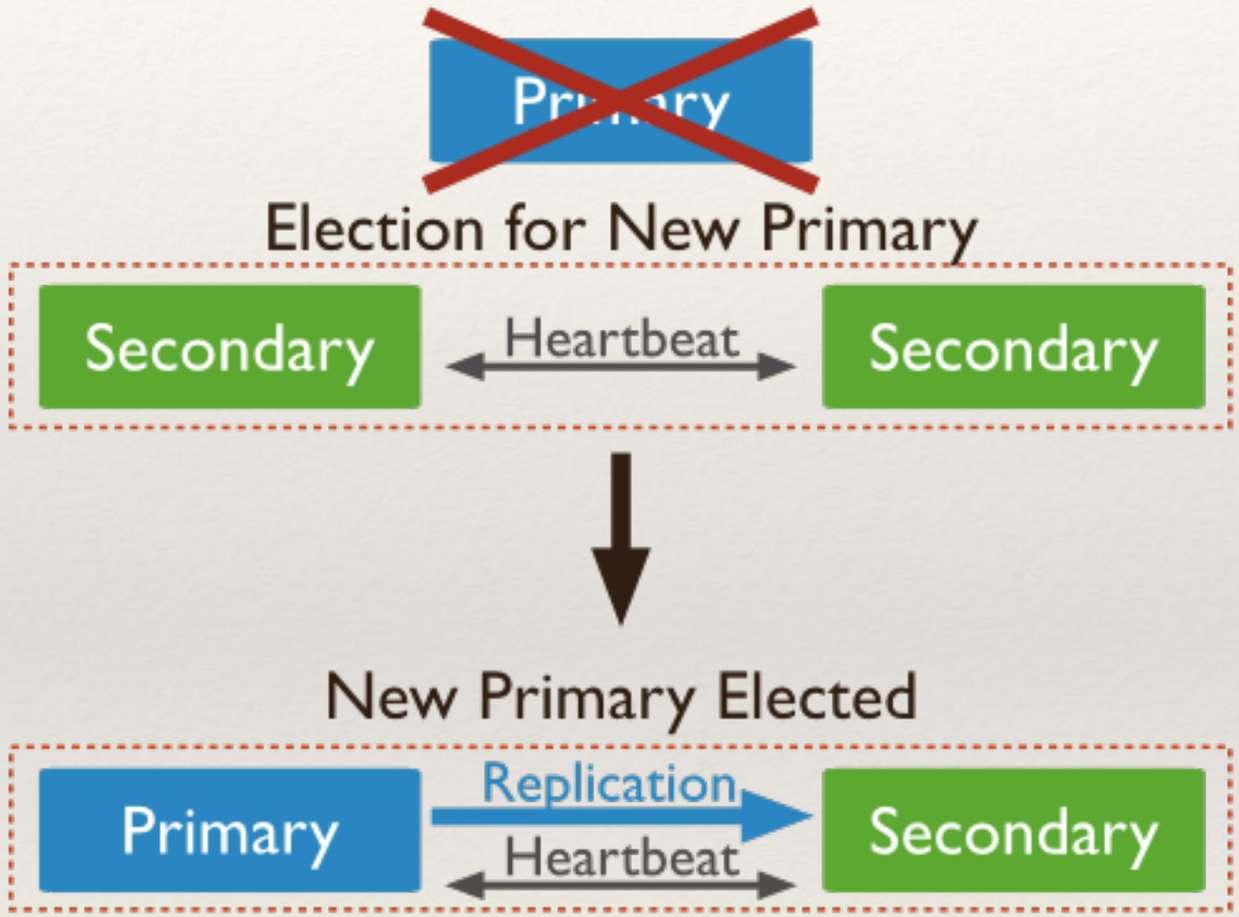


MongoDB Replica Set





MongoDB Replica Set



MongoDB Oplog

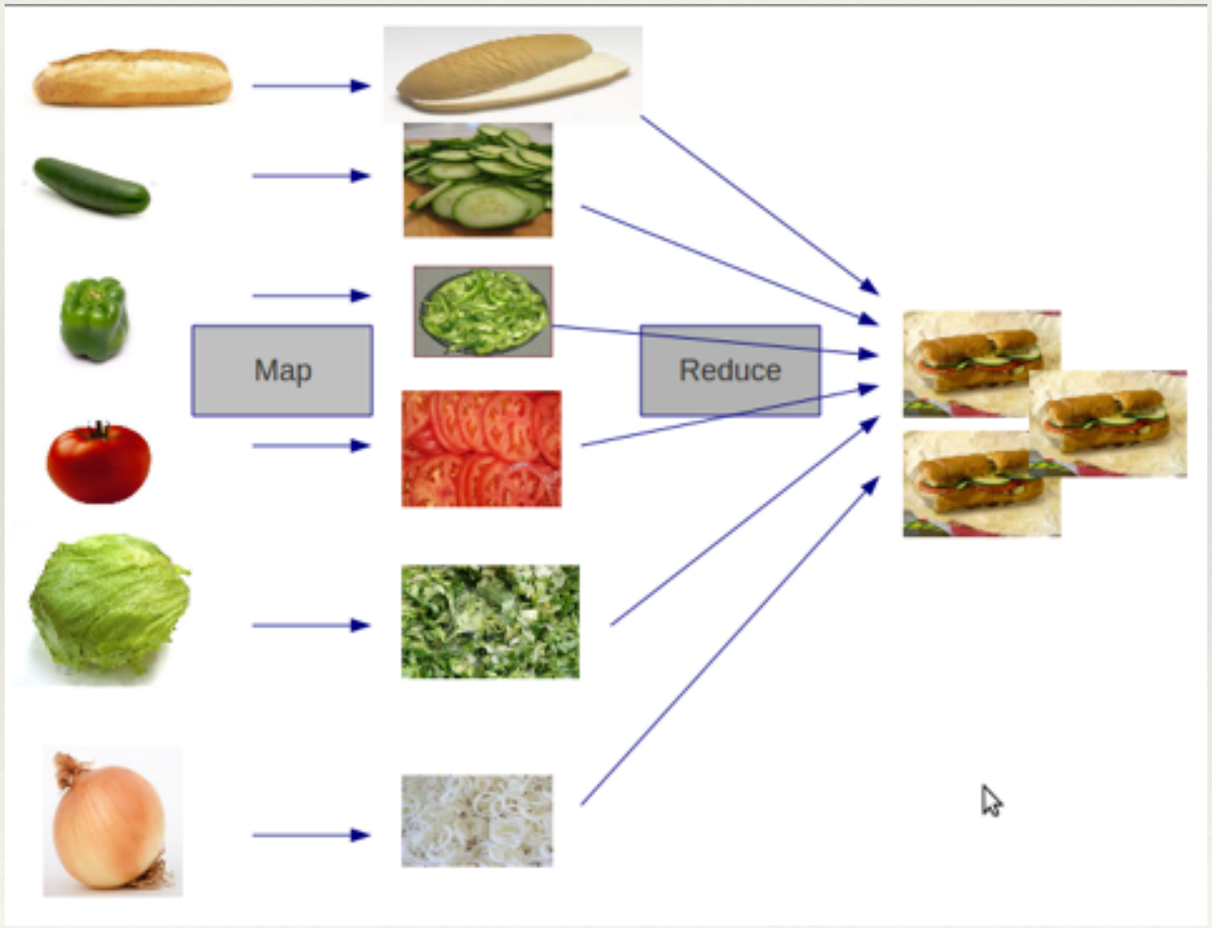
Oplog的作用：所有MongoDB操作的日志。
在复制集内部的节点之间，通过扩散-回放机制，可以高效、可靠地做到节点间的复制。

Oplog operators:

- i: insert
- u: update
- d: removal
- c: command

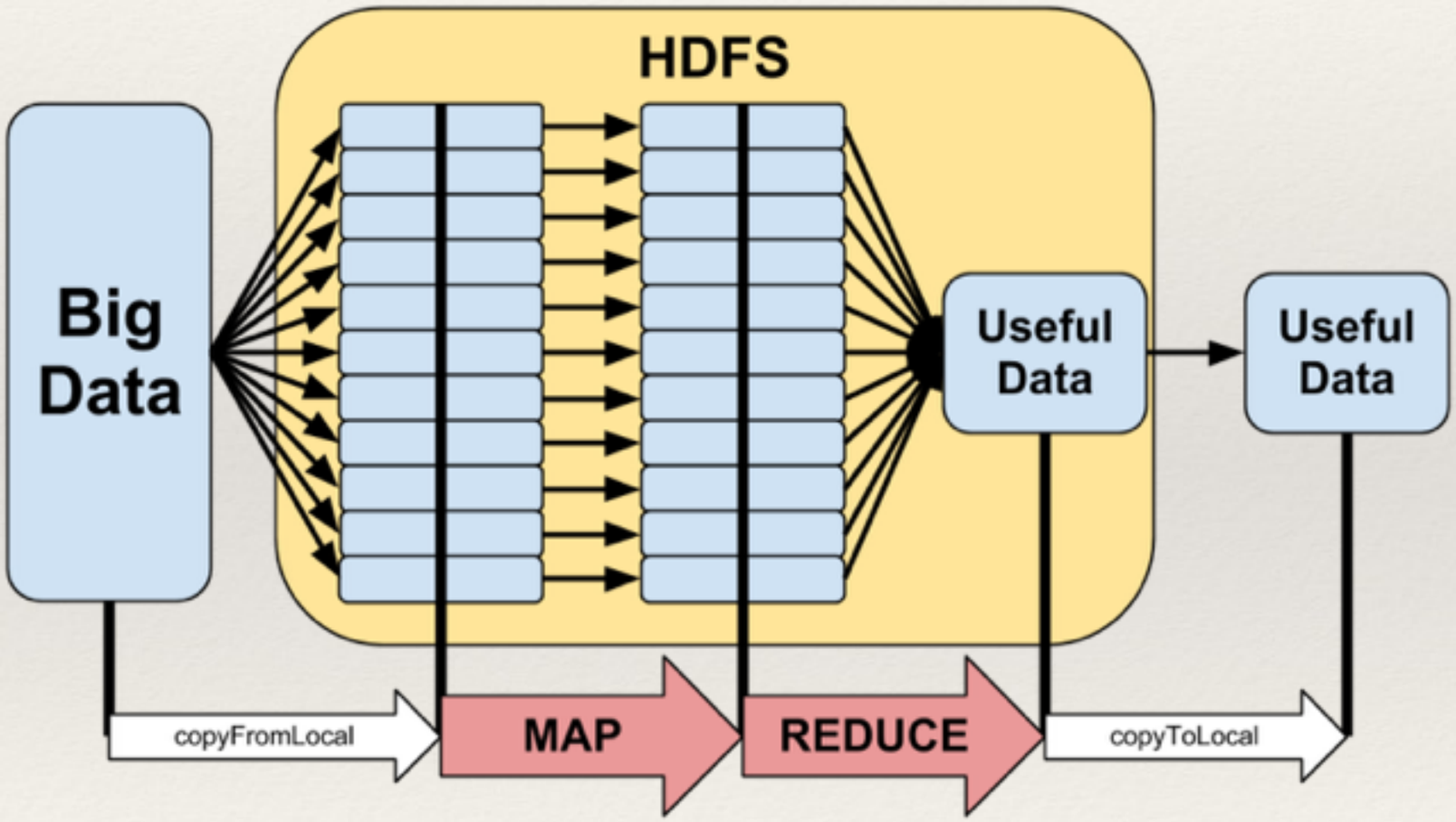
MongoDB Map Reduce

美食界的MapReduce



MongoDB Map Reduce

Hadoop中的MapReduce



MongoDB Map Reduce

MapReduce的核心步骤

- 向量化
- 并行化
- 合并

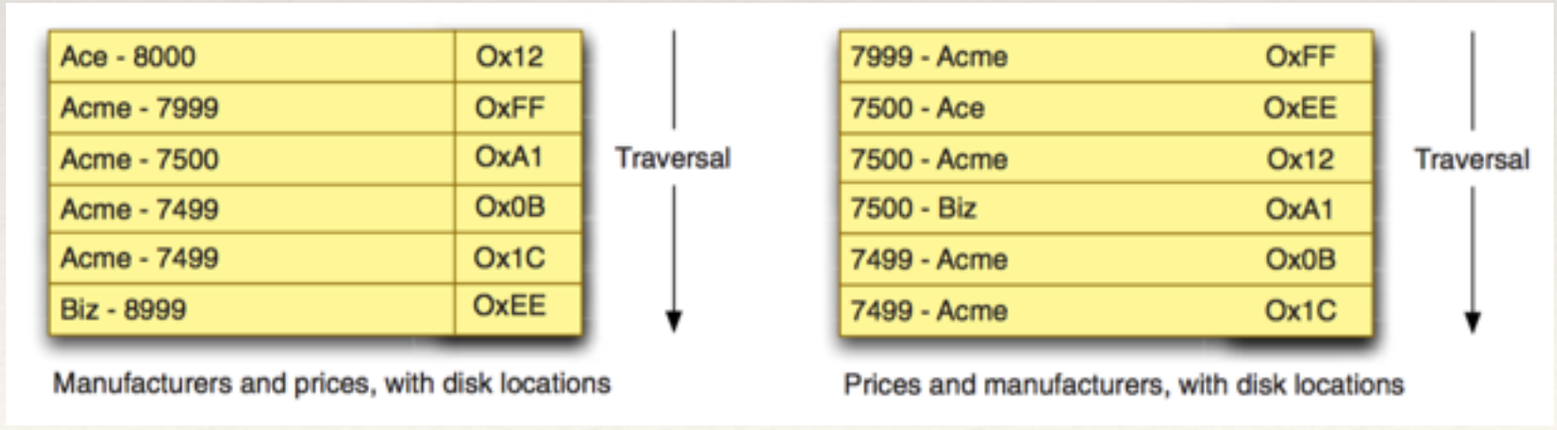
MongoDB Indexes

- 查看一个集合的索引情况
- 索引类型
  - 普通索引
  - unique: 唯一约束索引
  - partialFilterExpression: Partial indexes
  - sparse: 稀疏索引
  - expireAfterSeconds: TTL索引
  - 2dsphromere: 地理空间索引
- 索引对性能的影响

MongoDB Indexes

Single-key and compound-key

```
{
  "manufacturer": "acme",
  "price": 7999,
  "address": "Beijing",
  "_id": 182839
}
```



MongoDB Indexes

使用索引优化查询条件

```
lvxingpai-dev:PRIMARY> db.Proxy.find( {"validation.baidu.latency": {$lt: 0.1}}).explain()
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "andaman.Proxy",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "validation.baidu.latency" : {
                                "$lt" : 0.1
                        }
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "validation.baidu.latency" : {
                                        "$lt" : 0.1
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "a557d2a300aa",
                "port" : 27017,
                "version" : "3.0.5",
                "gitVersion" : "8bc4ae20708dbb493cb09338d9e7be6698e4a3a3"
        },
        "ok" : 1
}
```

```
lvxingpai-dev:PRIMARY> db.Proxy.find( {"validation.baidu.latency": {$lt: 0.1}}).explain()
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "andaman.Proxy",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "validation.baidu.latency" : {
                                "$lt" : 0.1
                        }
                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "validation.baidu.latency" : 1
                                },
                                "indexName" : "validation.baidu.latency_1",
                                "isMultiKey" : false,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "validation.baidu.latency" : [
                                                "[-inf.0, 0.1)"
                                        ]
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "a557d2a300aa",
                "port" : 27017,
                "version" : "3.0.5",
                "gitVersion" : "8bc4ae20708dbb493cb09338d9e7be6698e4a3a3"
        },
        "ok" : 1
}
```

MongoDB GeoSpatial

什么是GeoJSON

```
{
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  }
}


{
  "geometry": {
    "type": "Polygon",
    "coordinates": [ [ [ 0 , 0 ] , [ 3 , 6 ] , [ 6 , 1 ] , [ 0 , 0  ] ] ]
}
```

MongoDB GeoSpatial

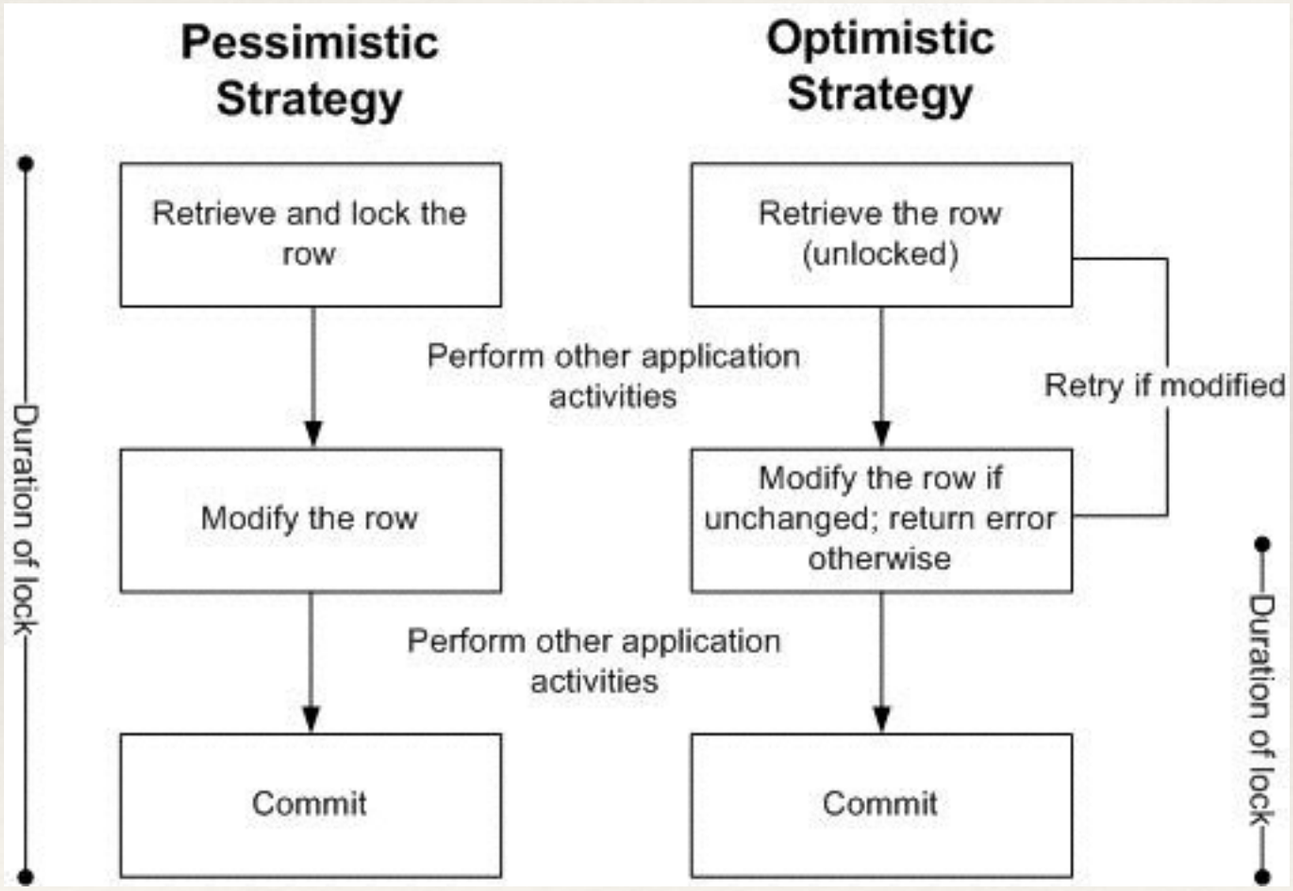
如何查询?

```
db.Locality.find( {location: {
  $near: {
    $geometry: {
      type: "Point",
      coordinates: [116.393525, 39.969654]
    },
    $maxDistance: 10000
  }
}}, {zhName: 1})
```

```
db.Locality.find( {location: {
  $geoWithin: {
    $geometry: {
      type: "Polygon",
      coordinates: [
       [ [ 0 , 0 ] , [ 3 , 6 ] ,
         [ 6 , 1 ] , [ 0 , 0  ] ] ]
    },
    $maxDistance: 10000
  }
}}, {zhName: 1})
```

MongoDB Concurrency

如何实现乐观锁?




```
{
  "manufacturer": "acme",
  "price": 7999,
  "address": "Beijing",
  "_id": 182839,
  "_version": 23
}
```

MongoDB in Action

如何实现分布式的悲观锁?

★ **etcd-lock** public

A distributed lock using etcd

A node package for distributed locks using etcd. It required ES6 support, so node >=4.0.0 should be used.

Usage

Instantiation

Use the exported constructor to instantiate new locks. The constructor has the signature `function Lock(etcd, key, id, ttl)`, where `etcd` is a `node-etcd` client, `key` is the etcd key to use for locking, `id` is a unique node identifier and `ttl` is the TTL of the lock in seconds.

```
var Lock = require("etcd-lock");
var Etcd = require("node-etcd");
var os = require("os");

var key = "/example/lock"
  , id = os.hostname()
  , ttl = 60 // seconds;

var lock = new Lock(new Etcd(), key, id, ttl);
```

Locking and unlocking

Each lock instance has the methods `lock()` and `unlock()`, both of which return a promise that will be fulfilled or rejected when the action is complete

MongoDB Best Practice

- 使用云服务
- 使用Docker来部署
- 始终启用备份
- 使用最高版本
  - 2.x vs 3.x
- 使用64位系统
- 内存最重要
- 正确使用update
- 不要部署在互联网或DMZ里面
- 使用审计功能

MongoDB Best Practice

如何备份?

- mongodump & mongorestore
- delayed replica set
- volume snapshot (lvm2)
- cloud service

MongoDB Best Practice

Mongo Connector

Topics

Using Mongo Connector

- Installation
- Getting Started
- Configuration Options
- Usage with Authentication
- Re-syncing the connector

Common Applications

- Usage with Solr
- Usage with Elasticsearch
- Usage with MongoDB

System Internals

- System Overview
- Oplog Progress File
- Handling rollbacks

应用领域：

- 模拟trigger
- 数据同步机制
  - 和Solr同步
  - 和Elasticsearch同步
- 数据中心间通讯
- 其它