mongoDB入门与开发技巧

BY 君三思 2012-11
http://weibo.com/junsansi

传说中的NoSQL

- NoSQL，不是不用SQL，而是：**N**ot **O**nly **SQL**
- NoSQL运动并不是项全新的数据库革命性运动，早期就有人提出，发展至2009年趋势越发高涨。NoSQL的拥护者们提倡运用非关系型的数据存储，相对于目前铺天盖地的关系型数据库运用，这一概念无疑是一种全新的思维的注入。

- **适用场景：**
  1. 对数据库高并发读写。
  2. 对海量数据的高效率存储和访问。
  3. 对数据库的高可扩展性和高可用性。
- **不适用场景：**
  1. 数据库事务一致性需求
  2. 数据库的写实时性和读实时性需求
  3. 对复杂的SQL查询，特别是多表关联查询的需求

NoSQL产品


Redis
HBASE
Cassandra
mongoDB
riak
CouchDB
relax
Tokyo Cabinet
Scalaris
Project Voldemort
Neo4j
the graph database
membase

传说中的mongoDB

- mongoDB是最像rdbms的，基于k/v的，文档类型数据库
- mongoDB是一个高性能，开源，无模式的文档型数据库，它在许多场景下可用于替代传统的关系型数据库或键/值存储方式。
- 不支持SQL，但有自己功能强大的查询语法
- 使用BSON作为数据存储和传输的格式

Created by 10gen

关系型数据库与mongoDB

- 关键知识点的对应

传统SQL术语	Mongo中的术语
database	database
table	collection
index	index
row	BSON document
column	BSON field
join	embedding and linking
primary key	_id field
group by	aggregation

mongoDB简介

- mongoDB与MySQL较为相近，也是由一个个的db组成；
- 传统rdbms中的db存储的是tables，mongoDB中则是collections；
- 传统rdbms中的tables存储的是rows，mongoDB中则是documents;

- 注意：mongoDB中的database和collection不需要创建

线上部署 – 选择版本

- MongoDB的版本由三部分组成：A.B.C
  - A is the **major** version. This will rarely change and signify very large changes
  - B is the **release** number. This will include many changes including features and things that possible break backwards compatibility. Even Bs will be stable branches, and odd Bs will be development.
  - C is the **revision** number and will be used for bugs and security issues.
- B为奇数时表示开发版本，B为偶数时表示稳定版本；
- http://www.mongodb.org/display/DOCS/Version+Numbers
- 我们选择了最新的2.2GA

线上部署 – 选择平台

- Windows平台？支持；
- Linux平台？支持；
- 连Solaris都支持；
- 实在不行的话，就源码编译吧；
- 64bit：必须的；

- 提示：mongoDB对内存敏感，内存越大，性能越好！

线上部署 – 呃，部署

- 下载：
  - http://www.mongodb.org/downloads
- 解压：
  - # tar xvfz mongodb-linux-x86_64-2.2.0.tgz
- 运行：
  - # mongod &
- Done!

mongod参数

名称	说明
dbpath	数据文件存放路径，每个数据库会在其中创建一个子目录，用于防止同一个实例多次运行的mongod.lock 也保存在此目录中
logpath	日志文件
logappend	日志采用追加模式（默认是覆写模式）
bind_ip	对外服务的绑定ip，一般设置为空，及绑定在本机所有可用ip 上
port	对外服务端口。Web 管理端口在这个port 的基础上+1000
fork	以后台Daemon 形式运行服务
journal	开启日志功能，通过保存操作日志来降低单机故障的恢复时间
syncdelay	系统同步刷新磁盘的时间，单位为秒，默认是60 秒
directoryperdb	每个db 存放在单独的目录中，建议设置该参数
maxConns	最大连接数
repairpath	执行repair 时的临时目录。在如果没有开启journal，异常down 机后重启，必须执行repair操作

mongoDB中的mongo

- mongo，类似MySQL中的mysql，或Oracle中的sqlplus；
- 简单易用的命令行交互工具；

- 细节要猛**戳**：
- http://www.mongodb.org/display/DOCS/mongo+-+The+Interactive+Shell

操作database

- 常用命令：
  - 查看所有数据库：
    - show dbs;
  - 使用/创建db(如果use的db不存在，则插入记录时会自动创建)：
    - use [dbname];
  - 查看当前使用的db(默认使用的test db)：
    - db.getName();
  - 查看当前db状态：
    - db.stats();
  - 查询当前db中的对象(collections)：
    - show tables;
  - 删除当前使用的db：
    - db.dropDatabase();

操作collection

- 忽略它吧，对开发来说就是透明的

简单操作：增/删/改/查

- 增加记录(会自动创建db和collection)：
  - db.[coll_name].insert(obj)：

```
mongo> db.users.insert({username: "junsansi", mobile: 13901356973, qq:5454589});
```

  - db.[coll_name].save(obj)与之等效；
- 查询记录：
  - db.[coll_name].find……

```
mongo> db.users.find();
```

- 修改记录：
  - db.[coll_name].update……

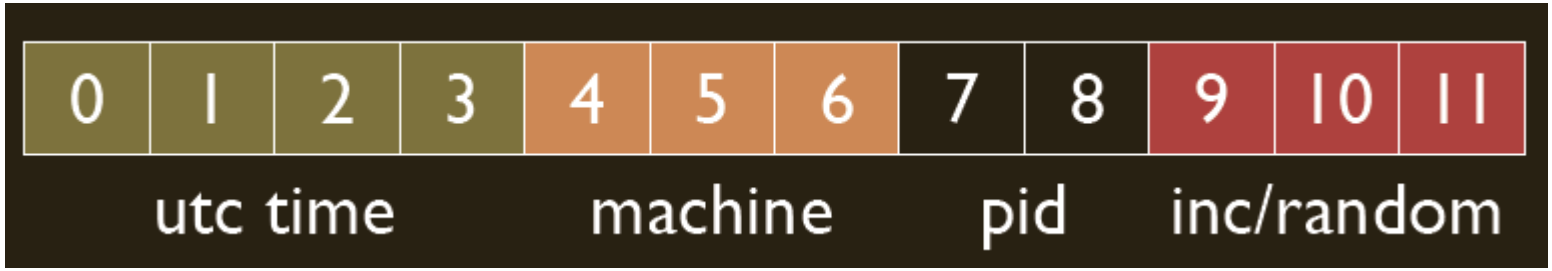
```
mongo> db.users.update({username: "junsansi"},{$set: {age:30}});
```

- 删除记录：
  - db.[coll_name].remove……

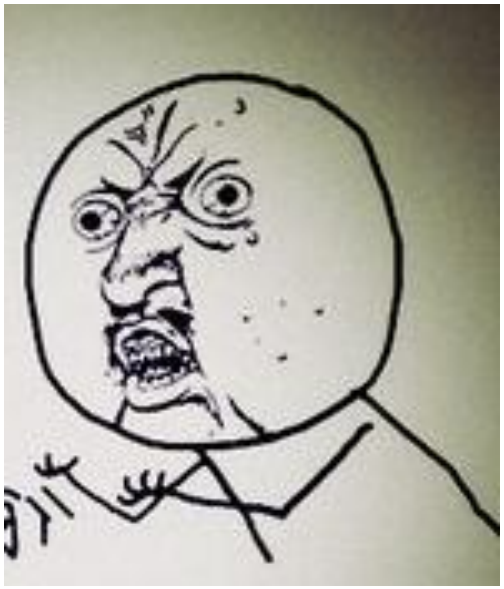
```
mongo> db.users.remove({username: "junsansi"});
```

Object ID的底细

- “_id” : ObjectId(“50a300ec1db69a0429229519”) 嘛玩易！



- 50a300ec timestamp
- 1db69a machine id
- 0429 process id
- 229519 counter



结构化存储的优势

- 插入一些记录，完全不用管范式及对象结构：
  - db.users.insert({username:"jss"});
  - db.users.insert({username:"junsansi", sex:"male"});
  - db.users.insert({username:"brain",sex:"male",phone:{tel:010,mobile:139}});

- db.users.find();

{ "_id" : ObjectId("50a30c8b1db69a042922951a"), "username" : "jss" }

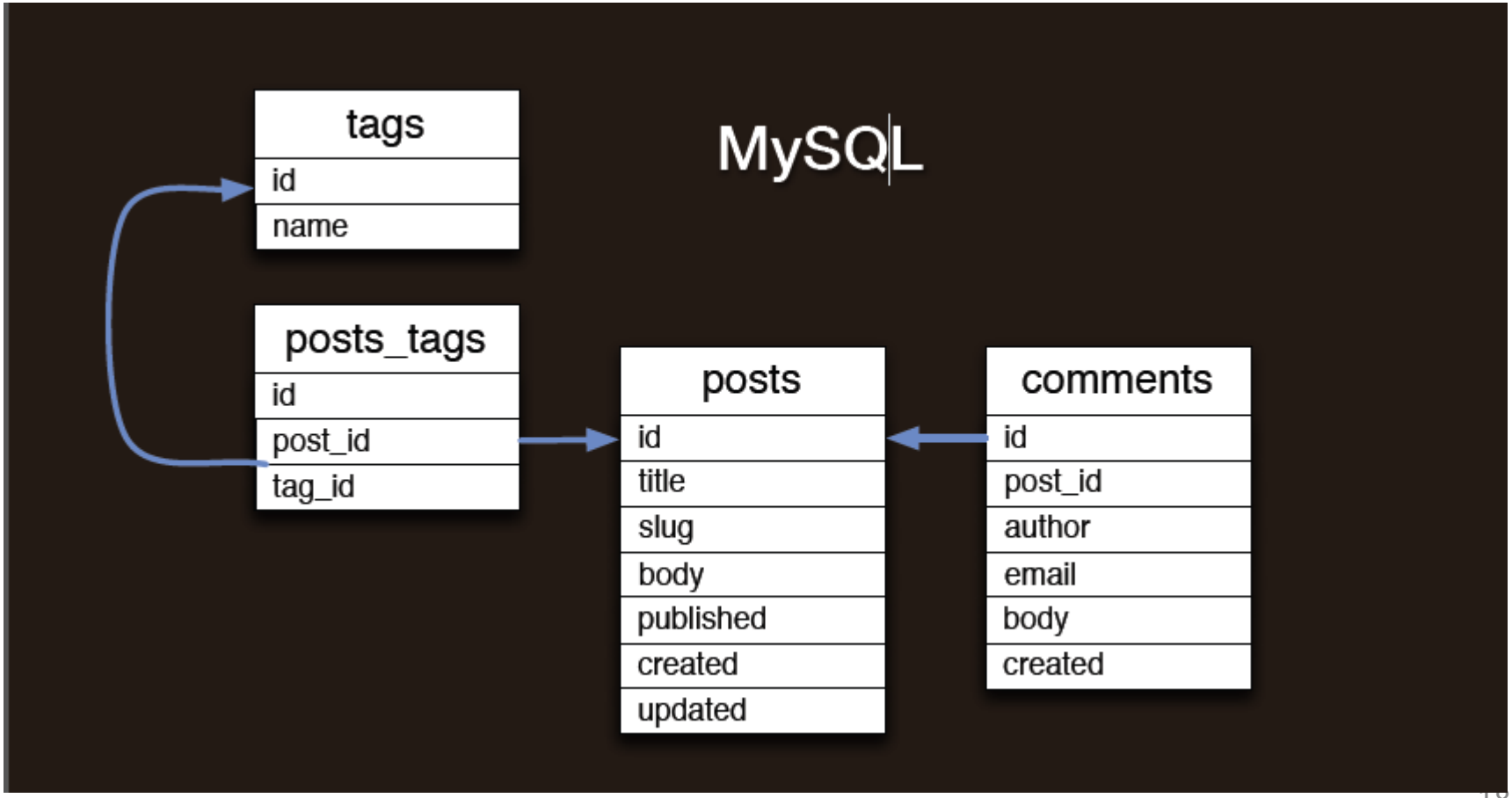
{ "_id" : ObjectId("50a30cbd1db69a042922951b"), "username" : "junsansi", "sex" : "male" }

{ "_id" : ObjectId("50a30d3f1db69a042922951c"), "username" : "brain", "sex" : "male", "phone" : { "tel" : 010, "mobile" : 139 } }

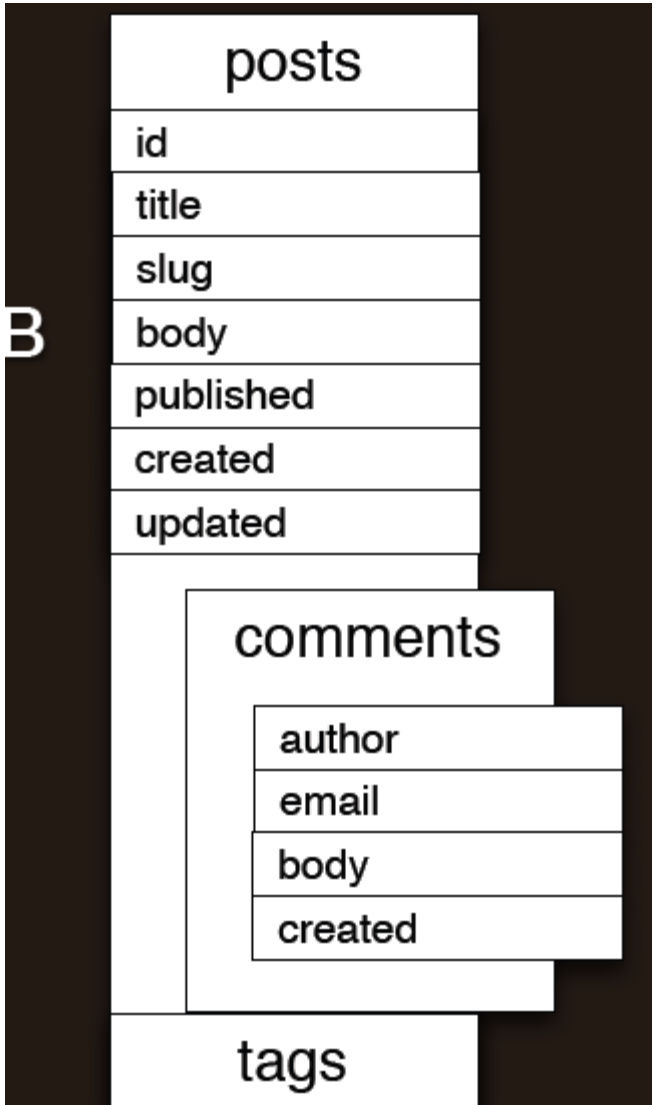
还有数组哟！

- 一本书的标签有多个，mongodb中可以这样：
  - db.t1.save({books:"step by step in oracle",tag:["tech","oracle","database"]});
  - db.t1.save({books:"colors",tag:["art","color"]});
- 查询书名为color的记录：
  - db.t1.find({books:"colors"});
  - { "_id" : ObjectId("50a44bf9333758d4323c0f82"), "books" : "colors", "tag" : ["art", "color"] }
- 为该书增加一条标签：
  - db.t1.update({books:"colors"},{$push: {tag:"pop"}});
- 再次查询验证：
  - db.t1.find({books:"colors"});
  - { "_id" : ObjectId("50a44bf9333758d4323c0f82"), "books" : "colors", "tag" : ["art", "color", "pop"] }

关系型数据库的对象设计



现在，我们用mongoDB



嵌套对象如何操作？

基本上，没啥区别！

- 查询联系方式中电话是8的：
  - db.users.find({"phone.tel":8});
- 更新电话：
  - db.users.update({"phone.tel":8},{$set:{"phone.tel":"010"}});
- 增加一条联系方式：
  - db.users.update({username:"brain"},{$set:{"phone.call":189}});
- 查询验证：
  - db.users.find({username:”brain”});
  - { "_id" : ObjectId("50a45c92333758d4323c0f84"), "phone" : { "call" : 189, "mobile" : 139, "tel" : "010" }, "sex" : "male", "username" : "brain" }

与SQL语句比较(1)

SQL SELECT语句	mongoDB find语句
SELECT * FROM users	db.users.find( ) *or* db.users.find({ })
SELECT id, user_id, status FROM users	db.users.find({ }, { user_id: 1, status: 1 })
SELECT user_id, status FROM users	db.users.find({ }, { user_id: 1, status: 1, _id: 0 })
SELECT * FROM users WHERE status = "A"	db.users.find({ status: "A" })
SELECT user_id, status FROM users WHERE status = "A"	db.users.find({ status: "A" }, { user_id: 1, status: 1, _id: 0 })
SELECT * FROM users WHERE status != "A"	db.users.find({ status: { $ne: "A" } })
SELECT * FROM users WHERE status = "A" AND age = 50	db.users.find({ status: "A", age: 50 })

与SQL语句比较(2)

SQL SELECT语句	mongoDB find语句
SELECT * FROM users WHERE status = "A" OR age = 50	db.users.find({ $or: [{ status: "A" } , { age: 50 }] })
SELECT * FROM users WHERE age > 25	db.users.find({ age: { $gt: 25 } })
SELECT * FROM users WHERE age < 25	db.users.find({ age: { $lt: 25 } })
SELECT * FROM users WHERE age > 25 AND age <= 50	db.users.find({ age: { $gt: 25, $lte: 50 } })
SELECT * FROM users WHERE user_id like "%bc%"	db.users.find({ user_id: /bc/ })
SELECT * FROM users WHERE user_id like "bc%"	db.users.find({ user_id: /^bc/ })
SELECT * FROM users WHERE status = "A" ORDER BY user_id ASC	db.users.find({ status: "A" }).sort({ user_id: 1 })

与SQL语句比较(3)

SQL SELECT语句	mongoDB find语句
SELECT * FROM users WHERE status = "A" ORDER BY user_id DESC	db.users.find({ status: "A" }).sort({ user_id: -1 })
SELECT COUNT(*) FROM users	db.users.count() *or* db.users.find().count()
SELECT COUNT(user_id) FROM users	db.users.count({ user_id: { $exists: true } }) *or* db.users.find({ user_id: { $exists: true } }).count()
SELECT COUNT(*) FROM users WHERE age > 30	db.users.count({ age: { $gt: 30 } }) *or* db.users.find({ age: { $gt: 30 } }).count()

与SQL语句比较(4)

SQL SELECT语句	mongoDB find语句
SELECT DISTINCT(status) FROM users	db.users.distinct("status")
SELECT * FROM users LIMIT 1	db.users.findOne() *or* db.users.find().limit(1)
SELECT * FROM users LIMIT 5 SKIP 10	db.users.find().limit(5).skip(10)
EXPLAIN SELECT * FROM users WHERE status = "A"	db.users.find({ status: "A" }).explain()
SQL DELETE语句	mongoDB remove语句
DELETE FROM users WHERE status = "D"	db.users.remove({ status: "D" })
DELETE FROM users	db.users.remove( )

分析函数对照参考：
http://docs.mongodb.org/manual/reference/sql-aggregation-comparison/

不多但实用的运算符

- $gt：greater than, >
- $lt：less than. <
- $gte：greater than or equal, >=
- $lte：less than or equal, <=
- $all：
- $exists：
- $mod：
- $ne：not equal, <>
- $in：
- $nin：not in
- $or

索引也很强大哟

- MongoDB的索引跟传统数据库的索引相似，默认_id列会创建为主键，并且不可删除；
- 单个collection 最多有64个index，单个query只会选择1个index；
- Key值长度不能超过800字节；
- MongoDB 中的索引是传统的B*Tree索引；

索引也很强大哟

- 创建索引：
  - db.user.ensureIndex({colname:1}); 其中1表示升序，-1表示降序

  创建索引时可以指定多个选项，复合索引当然也是可以的：unique,sparse,dropDups,expireAfterSeconds等，分别对应不同场景
- 查询collection上创建的索引：
  - db.user.getIndexes();
- 重建索引：
  - db.user.reIndex();
- 删除索引：
  - db.user.dropIndex(indname);

创建索引

db.user.ensureIndex({mobile:1}) 其中1表示升序，-1表示降序
db.user.getIndexes()

```
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },
                "ns" : "test.user",
                "name" : "_id_"
        },
        {
                "v" : 1,
                "key" : {
                        "mobile" : 1
                },
                "ns" : "test.user",
                "name" : "mobile_1"
        }
]
```

创建索引时可以指定多个选项，复合索引当然也是可以的：unique,sparse,dropDups,expireAfterSeconds等，分别对应不同场景

复合索引

- MongoDB可对多个字段建立复合索引，字段后面的1表示升序，-1表示降序，是用1还是用-1主要是跟排序的时候或指定范围内查询的时候有关的
- 示例： db.user.ensureIndex({name:1,mobile:-1})
- 利用复合索引的多值性，可以通过以索引字段开头的组合条件进行查询，例如：以a,b,c三个字段建立索引，则查询可利用索引的查询条件为：a或者a,b或者a,b,c

地理位置索引

- 此索引是MongoDB的一大亮点，也是foursquare选择它的原因之一。原理是建立索引时根据坐标对平面进行多次geohash，近似查询时就可以利用相同前缀的geohash值，mongoDB默认进行26次geohash，hash次数可控

- 示例：

```
db.map.ensureIndex({point : “2d”})
db.map.find( {point:[50,50]} )
db.map.find( {point:{$near:[50,50]}})
db.map.find( {point:{$within:{$box:[[40,40],[60,60]]}}} )
```

索引管理

- 可通过往system.indexes中插入记录来创建索引，如：

```
var spec = {ns: “test.user", key: {‘name’: 1}, name: ‘name_index’}
db.system.indexes.insert(spec, true)
```

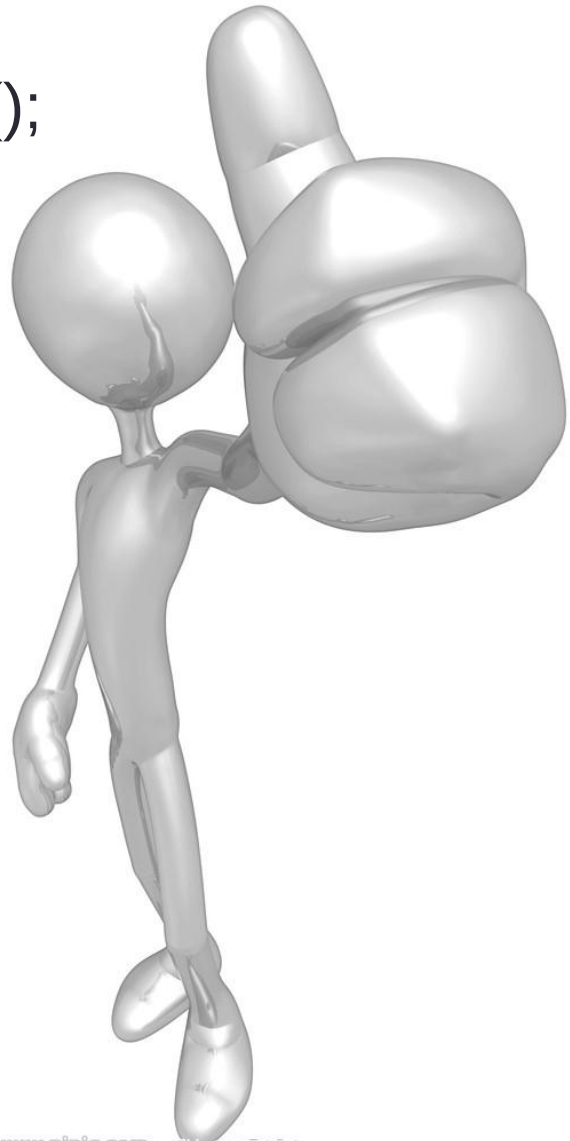
- 删除索引命令：

```
db.user.dropIndexes()删除该集合所有索引
db.user.dropIndexes({name:1})
db.runCommand({dropIndexes:'user', index : {name:1}})
db.runCommand({dropIndexes:'user', index :'*'})
```

- 索引重建：db.user.reIndex()

连执行计划都有

- db.collection.find(query).explain();



还有很多高级技巧的哟

自己去探索吧！

开发语言支持

Language	Packages	Source	API Reference
C	source tarballs	GitHub	API
C#	packages	GitHub	API
C++	source tarballs	GitHub	API
Erlang	source tarballs	GitHub	API
Haskell	Hackage	GitHub	API
Javascript	Package	GitHub	API
Java	jar	GitHub	API
Perl	cpan	GitHub	API
PHP	PECL	GitHub	API
Python	PyPI	GitHub	API
Ruby	RubyGems	GitHub	API
Scala (via Casbah)	tarball	GitHub	API

因为猛犸，咱们只说c#

- 驱动从哪下？

- 下列类包必须引用：

```
using MongoDB.Bson;
using MongoDB.Driver;
```

- 连接MongoDB：

```
var connectionString = "mongodb://localhost:27017";
var server = MongoServer.Create(connectionString);
```

- 选择数据库：

```
var database = server.GetDatabase("test"); // "test" is the name of the database
```

增/删/改/查

- 先确定一个collection：

```
var collection = database.GetCollection<Entity>("entities");
```

- 增加：

```
var entity = new Entity { Name = "Tom" };
collection.Insert(entity);
var id = entity.Id;
```

- 查询：

```
var query = Query.EQ("_id", id);
var entity = collection.FindOne(query);
```

增/删/改/查

- 保存：

```
entity.Name = "Dick";
collection.Save(entity);
```

- 或更新：

```
var query = Query.EQ("_id", id);
var update = Update.Set("Name", "Harry"); // update modifiers
collection.Update(query, update);
```

- 删除：

```
var query = Query.EQ("_id", id);
collection.Remove(query);
```

给段完整的代码示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using MongoDB.Bson;
using MongoDB.Driver;
using MongoDB.Driver.Builders;


namespace ConsoleApplication1
{
    public class Entity
    {
        public ObjectId Id { get; set; }
        public string Name { get; set; }
    }

    class Program
    {
        static void Main(string[] args)
        {
            var connectionString = "mongodb://localhost/?safe=true";
            var server = MongoServer.Create(connectionString);
            var database = server.GetDatabase("test");
            var collection = database.GetCollection<Entity>("entities");

            var entity = new Entity { Name = "Tom" };
            collection.Insert(entity);
            var id = entity.Id;

            var query = Query.EQ("_id", id);
            entity = collection.FindOne(query);

            entity.Name = "Dick";
            collection.Save(entity);

            var update = Update.Set("Name", "Harry");
            collection.Update(query, update);

            collection.Remove(query);
        }
    }
}
```

说说程序仲么连

配置写操作连接到primary，读操作循环方式由slave响应：

- mongodb://host1,host2,host3/?connect=replicaset;replicaset=rs0;slaveok=true

- 亲，扩展性非常强，而且对于开发完全透明的哟！

- 详细配置参数参考：
- http://www.mongodb.org/display/DOCS/Connections

- Q&A