MongoDB

NoSQL,全称是"Not Only Sql",指的是非关系型的数据库。

非关系型数据库主要有这些特点:**非关系型的、分布式的、开源的、水平可扩展的**。

NoSQL 的拥护者们提倡运用非关系型的数据存储,通常的应用如:模式自由、支持简易复制、简单的 API、大容量数据等。

MongoDB 是一个介于关系数据库和非关系数据库之间的产品,是非关系数据库当中功能最丰富,最像关系数据库的。

MongoDB最大的特点：它支持的查询语言非常强大,其语法有点类似于面向对象的查询语言,几乎可以实现类似关系数据库单表查询的绝大部分功能。它是一个面向集合的,模式自由的文档型数据库。

1、 面向集合(Collection-Orented)
意思是数据被分组存储在数据集中, 被称为一个集合(Collenction)。每个集合在数据库中 都有一个唯一的标识名,并且可以包含无限数目的文档。集合的概念类似关系型数据库(RDBMS)里的表(table),不同的是它不需要定义任何模式(schema)。

2、 模式自由(schema-free)
存储在 MongoDB 数据库中的文件,我们不需要知道它的任何结构定义。提了这 么多次"无模式"或"模式自由",它到是个什么概念呢?例如,下面两个记录可以存在于同一个集合里面:
{"welcome" : "Beijing"}
{"age" : 28}

3、 文档型 意思是我们存储的数据是键-值对的集合,键是字符串,值可以是数据类型集合里的任意类型, 包括数组和文档. 文件存储格式为 BSON(一种 JSON 的扩展)

适用场景

1 网站数据:MongoDB 非常适合实时的插入,更新与查询,并具备网站实时数据存储所需的复制及高度伸缩性

2 缓存:由于性能很高,MongoDB 也适合作为信息基础设施的缓存层。在系统重启之后, 由 MongoDB 搭建的持久化缓存层可以避免下层的数据源过载

3 大尺寸,低价值的数据:使用传统的关系型数据库存储一些数据时可能会比较昂贵,在此之前,很

多时候程序员往往会选择传统的文件进行存储

4 高伸缩性的场景:MongoDB 非常适合由数十或数百台服务器组成的数据库。MongoDB的路线图中已经包 对 MapReduce 引擎的内置支持

5 用于对象及 JSON 数据的存储:MongoDB 的 BSON 数据格式非常适合文档化格式的存储及查询

mongodb配置文件

mongod.conf

```
dbpath=/Users/carmack/data/mongo
```

启动mongod

```
./mongod -f mongod.cnf
```

mongo客户端

```
进入客户端
./mongo

查看数据库命令
show dbs

打开数据库
use taobao
查看集合（对应mysql table）
show collections
```

文档：文档(Document)是MongodDB中的核心概念，他是MongoDB逻辑存储的最小基本单元，　　对应mysql记录
集合：多个文档组成的集合　　对应mysql table
数据库：多个集合组成的数据库　对应mysql database

MongodDB的数据类型是：BSON的数据类型
BSON：是Binary JSON是二进制的格式

创建数据库

```
use newdb1
show dbs
```

数据库并没有添加，当我们在给数据库中的集合插入一条文档的时候就会自动创建一条文档、一个集合、一个数据库。

```
db.users.insert({"name": 'carmack'})
show colletions;
```

插入一条数据

```
db.users.insert({"uid":2,"uname":"carmack","isvip":true,"sex":null,"favorite":["apple","banana",1,2,3,4,5],"regtime":new Date()})
```

查询数据

```
db.users.find()
db.users.findOne({"uid":2})
```

删除数据

```
db.users.remove({"uid":2})
```

清空集合

```
db.users.remove({})
```

删除集合

```
db.users.drop()
```

更新文档

第一个文档为查询的文档，第二个文档为修改为什么文档，后面的文档会覆盖我们要修改文档的整个内容

```
db.users.update({"uid":2},{"uname":"jon"})
```

使用修改器$inc更新

对uid为2的用户增加100块钱工资

```
db.users.update({"uid":2},{"$inc":{"salary":100}})
```

减100

```
> db.users.update({"uid":2},{"$inc":{"salary":-100}})
```

添加一个字段$set修改器

```
db.users.update({"uid":2},{"$set":{"age":18}})
```

删除一个字段$unset修改器

```
db.users.update({"uid":2},{"$unset":{"age":true}})
```

数组的更新

```
db.users.update({"uid":2},{"$push":{"email":"a"}})
```

$pushAll在元组中增加多个元素,但是他不检查元素是否存在

```
db.users.update({"uid":2},{"$pushAll":{"email":["a", "b","c","d"]}})
```

$addToSet 往数组中添加一个不重复的元素

```
db.users.update({"uid":2},{"$addToSet":{"email":"d"}})
```

添加多个不重复的元素，这时候就得需要用到*$eache*操作符了

```
db.users.update({"uid":2},{"$addToSet":{"email":{"$each":["e","g","f","d"]}}})
```

删除数组元素

```
db.users.update({"uid":2},{"$pop":{"email":-1}}) #从左侧删除一个元素
db.users.update({"uid":2},{"$pop":{"email":1}})#从右侧删除一个元素
db.users.update({"uid":2},{"$pull":{"email":"b"}}) #删除数组内的指定一个元素
db.users.update({"uid":2},{"$pullAll":{"email":["b","c"]}}) #删除数组内指定的多个元素
```

数组元素的更新

通过数组.下标修改

```
db.users.update({"uid":2},{"$set":{"email.0":"tim.qq.com"}})
```

等于 =

```
db.users.find({"salary":3000}).pretty()
```

小于 <

```
db.users.find({"salary":{$lt:3000}}).pretty()
```

小于等于 <=

```
db.users.find({"salary":{$lte:3000}}).pretty()
```

大于 >

```
db.users.find({"salary":{$gt:3000}}).pretty()
大于等于 >=
db.users.find({"salary":{$gte:3000}}).pretty()
不等于 !=
db.users.find({"salary":{$ne:3000}}).pretty()

MongoDB 的 find() 方法可以传入多个键(key)，每个键(key)以逗号隔开，即常规 SQL 的 AND 条件
db.users.find({"id":2, "sex": "M"}).pretty()

MongoDB OR 条件语句使用了关键字 $or,语法格式如下：
db.users.find(
   {
      $or: [
         {key1: value1}, {key2:value2}
      ]
   }
).pretty()
```