MongoDB入门使用

数据库

首先介绍了下NoSQL

NoSQL介绍

NoSQL，指的是非关系型的数据库。NoSQL有时也称作Not Only SQL的缩写，是对不同于传统的关系型数据库的数据库管理系统的统称。

简单的说，之前学习的mysql，是关系型数据库，数据表和数据表之间由关系限制的，现在NoSQL就去除了那些关系，把你想存的数据直接存储进去即可，并且数据的类型没有做出要求

CAP定理

在计算机科学中, CAP定理（CAP theorem），又被称作 布鲁尔定理（Brewer's theorem），它指出对于一个分布式计算系统来说，不可能同时满足以下三点:

- **一致性(Consistency)** (所有节点在同一时间具有相同的数据)
- **可用性(Availability)** (保证每个请求不管成功或者失败都有响应)
- **分隔容忍(Partition tolerance)** (系统中任意信息的丢失或失败不会影响系统的继续运作)

CAP的核心是，分布式系统不可能满足三点，那就两两分开，CAP将NoSQL分为三大类:

- CA - 单点集群，满足一致性，可用性的系统，通常在可扩展性上不太强大。
- CP - 满足一致性，分区容忍性的系统，通常性能不是特别高。
- AP - 满足可用性，分区容忍性的系统，通常可能对一致性要求低一些。

RDBMS vs NoSQL

RDBMS

- 高度组织化结构化数据
- 结构化查询语言（SQL） (SQL)
- 数据和关系都存储在单独的表中。
- 数据操纵语言，数据定义语言
- 严格的一致性
- 基础事务

NoSQL

- 代表着不仅仅是SQL
- 没有声明性查询语言
- 没有预定义的模式
  -键 - 值对存储，列存储，文档存储，图形数据库
- 最终一致性，而非ACID属性
- 非结构化和不可预知的数据
- CAP定理
- 高性能，高可用性和可伸缩性

再来说下MongoDB

介绍

MongoDB 是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统。

在高负载的情况下，添加更多的节点，可以保证服务器性能。

MongoDB 旨在为WEB应用提供可扩展的高性能数据存储解决方案。

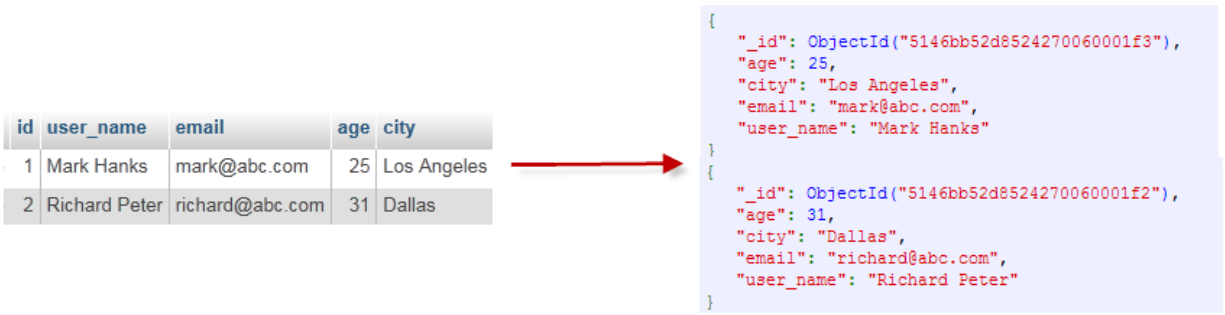
MongoDB 将数据存储为一个文档，数据结构由键值(key=>value)对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档，数组及文档数组。

mongo中概念

SQL概念	MongoDB概念	解释
database	database	数据库
table	collection	数据库表/集合
row	document	数据记录行/文档
column	field	数据字段/域
index	index	索引

table joins		表连接，MongoDB不支持
primary key	primary key	主键 MongoDB自动将_id字段设置为主键

通过图例，了解下MongoDB中的概念：



正式讲解

细说概念

数据库

mongoDB中可以建立多个数据库，不同数据库都有自己的集合和权限，不同的数据库也放置在不同的文件中

使用**show dbs**命令可以显示所有数据的列表

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

使用**db**命令可以显示当前数据库对象或集合

```
> db
local
```

使用**use**命令可以连接到一个指定的数据库

```
> use local
switched to db local
```

MongoDB中还有保留的数据库，可以直接访问这些特殊作用的数据库：

- admin： 相当于linux中的root用户，如果将一个用户添加到这个数据库中，那么该用户自动继承所有数据库的权限
- local：这个数据库永远不会被复制，可以用于存储限于本地单台服务器的任意集合
- config：当Mongo用于分片设置时，config数据库在内部使用，用于保存分片的相关信息

文档

文档就是一组键值对（即BSON），一个简单的文档例子如下：

```
{"site":"www.baidu.com", "name":"百度"}
```

需要注意：

1. 文档中键值对是有序的
2. 文档中的值有多种数据类型可以使用
3. 区分类型和大小写
4. 文档不能有重复的键
5. 文档的键是字符串

集合

集合就是文档组，相当于Mysql中的表格。

集合存于数据库中，没有固定的结构，所以可以对集合插入不同格式和类型的数据，但是一般情况下和Mysql中的表格相同，一个集合中存储的数据都是有关联性的。

数据类型

数据类型	描述
String	字符串 常用数据类型，在Mongo中UTF-8编码的字符串才合法
Integer	整形数值 根据服务器，可分为32位或64位
Boolean	布尔值
Double	双精度浮点值
Min/Max keys	将一个值与 BSON（二进制的 JSON）元素的最低值和最高值相对比。
Array	数组
Timestamp	时间戳
Object	用于内嵌文档
Null	空值

Symbol	符号 常用于特殊符号类型语言
Date	日期时间 用UNIX时间格式来存储日期
Object ID	对象ID
Binary Data	二进制数据
Code	代码类型
Regular expression	正则表达式类型

ObjectId

ObjectId 类似唯一主键，可以很快的去生成和排序，包含 12 bytes，含义是：

- 前 4 个字节表示创建 unix 时间戳,格林尼治时间 UTC 时间，比北京时间晚了 8 个小时
- 接下来的 3 个字节是机器标识码
- 紧接的两个字节由进程 id 组成 PID
- 最后三个字节是随机数

连接

MongoDB连接可以通过php连接，也可以通过MongoDB Shell连接。下面说下标准URI连接语法：

```
mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:po
rtN]]][/[database][?options]]
```

- mongodb:// 固定格式
- username:password@ 可选 可以连接远程数据库
- hostX 必须指定至少一个host 如果要连接复制集 请制定多个主机地址
- portX 可选指定端口 默认27017
- /database 指定数据库 默认test数据库
- ?options 选项全部是键值对

选项	描述
replicaSet=name	验证replica set的名称。 Impliesconnect=replicaSet.
slaveOk=true|false	• true:在connect=direct模式下，驱动会连接第一台机器，即使这台服务器不是主。在connect=replicaSet模式下，驱动会发送所有的写请求到主并且把读取操作分布在其他从服务器。 • false: 在 connect=direct模式下，驱动会自动找寻主服务器. 在connect=replicaSet模式下，驱动仅仅连接主服务器，并且所有的读写命令都连接到主服务器。
safe=true|false	• true: 在执行更新操作之后，驱动都会发送getLastError命令来确保更新成功。(还要参考 wtimeoutMS). false: 在每次更新之后，驱动不会发送getLastError来确保更新成功。
w=n	驱动添加 { w : n } 到getLastError命令. 应用于safe=true。
wtimeoutMS=ms	驱动添加 { wtimeout : ms } 到 getlasterror 命令. 应用于 safe=true.
fsync=true|false	• true: 驱动添加 { fsync : true } 到 getlasterror 命令.应用于 safe=true. • false: 驱动不会添加到getLastError命令中。
journal=true|false	如果设置为 true, 同步到 journal (在提交到数据库前写入到实体中). 应用于 safe=true
connectTimeoutMS=ms	可以打开连接的时间。
socketTimeoutMS=ms	发送和接受sockets的时间。

数据库操作

创建数据库

MongoDB创建数据库的语法格式如下：

```
use DATABASE_NAME
```

删除数据库

MongoDB 删除数据库的语法格式如下：

```
db.dropDatabase()
```

集合操作

创建集合

MongoDB 中使用 createCollection() 方法来创建集合。

语法格式：

```
db.createCollection(name, options)
```

参数说明：

- name：要创建的集合名称
- options：可选参数，指定有关内存大小及索引的选项

options有如下参数：

字段	类型	描述
capped	布尔	（可选）如果为 true，则创建固定集合。固定集合是指有着固定大小的集合，当达到最大值时，它会自动覆盖最早的文档。 **当该值为 true 时，必须指定 size 参数。**
autoIndexId	布尔	（可选）如为 true，自动在 _id 字段创建索引。默认为 false。
size	数值	（可选）为固定集合指定一个最大值（以字节计）。 **如果 capped 为 true，也需要指定该字段。**
max	数值	（可选）指定固定集合中包含文档的最大数量。

下面是带有几个关键参数的 createCollection() 的用法：

创建固定集合 mycol，整个集合空间大小 6142800 KB, 文档最大个数为 10000 个。

```
> db.createCollection("mycol", { capped : true, autoIndexId : true, size
 :
    6142800, max : 10000 } )
{ "ok" : 1 }
```

删除集合

MongoDB 中使用 drop() 方法来删除集合。

语法格式：

```
db.collection.drop()
```

实例：

```
> db.test1.insert({"11":"12"})
WriteResult({ "nInserted" : 1 })
> show collections
test
test1
> db.test1.drop()
true
> show collections
test
```

文档操作

插入文档

MongoDB 使用 insert() 或 save() 方法向集合中插入文档，语法如下：

```
db.COLLECTION_NAME.insert(document)
```

以下文档可以存储在 MongoDB 的 runoob 数据库 的 col 集合中：

```
>db.col.insert({title: 'MongoDB 教程',
    description: 'MongoDB 是一个 Nosql 数据库',
    by: 'lyh',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
})
```

插入文档也可以使用db.col.save(document)命令。如果不指定_id字段，save方法同insert方法。如果指定了_id字段，那么会更新该_id的数据

查看文档

查看已插入文档：

```
> db.col.find()
{ "_id" : ObjectId("5c9c9699dc1d85c788c52e96"), "title" : "MongoDB 教程",
"description" : "MongoDB 是一个 Nosql 数据库", "by" : "lyh", "tags" : [ "mo
ngodb", "database", "NoSQL" ], "likes" : 100 }
```

更新文档

update() 方法用于更新已存在的文档。语法格式如下：

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

参数说明：

- query：update的查询条件 相当于sql中where后面的
- update：一些更新的操作符 相当于sql中set后面的
- upsert：当不存在更新对象时，true为插入，false为不插入
- multi：是否更改多条符合信息
- writeConcern：抛出异常的级别

将MongoDB 教程改为mongoDB

```
> db.col.update({'title':'MongoDB 教程'},{$set:{'title':'mongoDB'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.col.find().pretty()
{
    "_id" : ObjectId("5c9c9699dc1d85c788c52e96"),
    "title" : "mongoDB",
    "description" : "MongoDB 是一个 Nosql 数据库",
    "by" : "lyh",
    "tags" : [
        "mongodb",
        "database",
        "NoSQL"
    ],
    "likes" : 100
}
```

save方法使用和insert方法使用相同，只是如果BSON中包含_id的话，可以直接更新原有的数据

删除文档

注意：请在remove删除之前使用find查询判断条件是否正确

remove() 方法的基本语法格式如下所示：

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

如果MongoDB是2.6版本以后的，语法格式如下：

```
db.collection.remove(
   <query>,
   {
     justOne: <boolean>,
     writeConcern: <document>
   }
)
```

参数说明：

- query：删除条件
- justOne：true只删除一个 false删除全部匹配的 默认是falsex
- writeConcern：抛出异常级别

删除title为MongoDB 教程的文档

```
> db.col.remove({'title':'MongoDB 教程'})
WriteResult({ "nRemoved" : 2 })
> db.col.find()
{ "_id" : ObjectId("5c9c9699dc1d85c788c52e96"), "title" : "mongoDB", "des
cription" : "MongoDB 是一个 Nosql 数据库", "by" : "lyh", "tags" : [ "mongod
b", "database", "NoSQL" ], "likes" : 100 }
```

查询文档

find() 方法以非结构化的方式来显示所有文档。

MongoDB 查询数据的语法格式如下：

```
db.collection.find(query, projection)
```

- query：可选 查询条件
- projection：可选 查询时返回文档中所有键值

可以在find（）后加pretty（）方法让结果显示的更加易读

除了find（）方法，还有一个findOne（）方法，只返回一个文档

下面是一些实例：

```
> db.test.find().pretty()
{ "_id" : ObjectId("5c9b423405b418e98b4285b7"), "qwe" : "1" }
{ "_id" : ObjectId("5ca2cc73056f79a88b52e4dc"), "qwe" : "2" }
> db.test.find({'qwe':'1'},{'_id':1})
{ "_id" : ObjectId("5c9b423405b418e98b4285b7") }
> db.test.find({'qwe':'2'})
{ "_id" : ObjectId("5ca2cc73056f79a88b52e4dc"), "qwe" : "2" }
```

MongoDB中的条件语句查询

AND条件

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。

语法格式如下：

```
>db.col.find({key1:value1, key2:value2}).pretty()
```

OR条件

MongoDB OR 条件语句使用了关键字 $or,语法格式如下：

```
>db.col.find(
   {
      $or: [
         {key1: value1}, {key2:value2}
      ]
   }
).pretty()
```

MongoDB条件操作符

MongoDB中条件操作符有：

- (>) 大于 - $gt
- (<) 小于 - $lt
- (>=) 大于等于 - $gte
- (<=) 小于等于 - $lte
- (!=)不等于 - $ne

操作	格式	范例	RDBMS中的类似语句
等于	{<key>:<value>}	db.col.find({"by":"菜鸟教程"}).pretty()	where by = '菜鸟教程'
小于	{<key>:{$lt:<value>}}	db.col.find({"likes":{$lt:50}}).pretty()	where likes < 50
小于或等于	{<key>:{$lte:<value>}}	db.col.find({"likes":{$lte:50}}).pretty()	where likes <= 50
大于	{<key>:{$gt:<value>}}	db.col.find({"likes":{$gt:50}}).pretty()	where likes > 50
大于或等于	{<key>:{$gte:<value>}}	db.col.find({"likes":{$gte:50}}).pretty()	where likes >= 50
不等于	{<key>:{$ne:<value>}}	db.col.find({"likes":{$ne:50}}).pretty()	where likes != 50

MongoDB 索引

创建索引

MongoDB使用 createIndex() 方法来创建索引。

createIndex()方法基本语法格式如下所示：

```
>db.collection.createIndex(keys, options)
```

语法中 Key 值为你要创建的索引字段，1 为指定按升序创建索引，如果你想按降序来创建索引指定为 -1 即可。

Key可以是一个，也可以是多个，如果设置多个键，那么将生成混合索引

createIndex() 接收可选参数，可选参数列表如下：

Parameter	Type	Description
background	Boolean	建索引过程会阻塞其它数据库操作，background可指定以后台方式创建索引，即增加 "background" 可选参数。 "background" 默认值为**false**。
unique	Boolean	建立的索引是否唯一。指定为true创建唯一索引。默认值为**false**.
name	string	索引的名称。如果未指定，MongoDB的通过连接索引的字段名和排序顺序生成一个索引名称。
dropDups	Boolean	**3.0+版本已废弃。**在建立唯一索引时是否删除重复记录,指定 true 创建唯一索引。默认值为 **false**.
sparse	Boolean	对文档中不存在的字段数据不启用索引；这个参数需要特别注意，如果设置为true的话，在索引字段中不会查询出不包含对应字段的文档.。默认值为 **false**.
expireAfterSeconds	integer	指定一个以秒为单位的数值，完成 TTL设定，设定集合的生存时间。
v	index version	索引的版本号。默认的索引版本取决于mongod创建索引时运行的版本。
weights	document	索引权重值，数值在 1 到 99,999 之间，表示该索引相对于其他索引字段的得分权重。
default_language	string	对于文本索引，该参数决定了停用词及词干和词器的规则的列表。 默认为英语
language_override	string	对于文本索引，该参数指定了包含在文档中的字段名，语言覆盖默认的language，默认值为 language.

其他操作

1、查看集合索引
db.col.getIndexes()

2、查看集合索引大小
db.col.totalIndexSize()

3、删除集合所有索引
db.col.dropIndexes()

4、删除集合指定索引
db.col.dropIndex(“索引名称”)

MongoDB聚合

MongoDB中聚合(aggregate)主要用于处理数据(诸如统计平均值,求和等)，并返回计算后的数据结果。有点类似sql语句中的 count(*)。

aggregate() 方法的基本语法格式如下所示：

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

实例操作：

```
> db.col.aggregate([{$group:{_id:'$by',num_tutorial:{$sum:1}}}])
{ "_id" : "菜鸟教程", "num_tutorial" : 2 }
{ "_id" : "lyh", "num_tutorial" : 1 }
```

这里使用到$group，属于MongoDB中的聚合管道概念，在下面介绍下管道的概念

管道

MongoDB的聚合管道将文档在一个管道处理完后将结果传递给下一个管道处理，管道可以是重复的。
下面介绍几个常用的操作：

- $project：修改输入文档的结构，可以用来重命名、增加或删除域，也可以用于创建计算结果以及嵌套文档
- $match：用于过滤数据，只输出符合条件的文档。使用起来同标准查询操作
- $limit：用来限制返回文档的个数
- $skip：在聚合管道中跳过指定数量的文档，并返回余下的文档
- $group：将集合中的文档分组，可用于统计结果，同Mysql中的group by
- $sort：将输入文档排序后输出
- $geoNear：输出接近某一地理位置的有序文档
- $unwind：将文档中的某个数组类型字段拆分成多条，每条包含数组中的一个值

管道操作符实例

1.$project实例

```
> db.col.aggregate( { $project : { title : 1, likes : 1, _id : 0 } } )
{ "title" : "mongoDB", "likes" : 100 }
{ "title" : "PHP 教程", "likes" : 200 }
{ "title" : "MongoDB 教程", "likes" : 100 }
```

2.$match实例

```
> db.col.aggregate( [ { $match : { likes : { $gte : 100 , $lt : 200 } }
 }, { $group : { _id : '$title', count: { $sum :1} } } ] )
{ "_id" : "MongoDB 教程", "count" : 1 }
{ "_id" : "mongoDB", "count" : 1 }
```

3.$skip实例

```
> db.col.aggregate( { $project : { title : 1, likes : 1, _id : 0 }},{ $sk
ip :2} )
{ "title" : "MongoDB 教程", "likes" : 100 }
```

Mongo方法与特殊操作符

$type

$type操作符是基于BSON类型来检索集合中匹配的数据类型，并返回结果。

如下：

```
> db.test.find({'qwe':{$type:'string'}})
{ "_id" : ObjectId("5c9b423405b418e98b4285b7"), "qwe" : "1" }
{ "_id" : ObjectId("5ca2cc73056f79a88b52e4dc"), "qwe" : "2" }
```

limit方法

limit()方法接受一个数字参数，该参数指定从MongoDB中读取的记录条数。

limit()方法基本语法如下所示：

```
>db.COLLECTION_NAME.find().limit(NUMBER)
```

实例如下：

```
> db.test.find({'qwe':{$type:'string'}})
{ "_id" : ObjectId("5c9b423405b418e98b4285b7"), "qwe" : "1" }
{ "_id" : ObjectId("5ca2cc73056f79a88b52e4dc"), "qwe" : "2" }

> db.test.find({'qwe':{$type:'string'}}).limit(1)
{ "_id" : ObjectId("5c9b423405b418e98b4285b7"), "qwe" : "1" }
```

skip方法

skip()方法来跳过指定数量的数据，skip方法同样接受一个数字参数作为跳过的记录条数。

skip() 方法脚本语法格式如下：

```
>db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)
```

实例如下：

```
> db.test.find({'qwe':{$type:'string'}})
{ "_id" : ObjectId("5c9b423405b418e98b4285b7"), "qwe" : "1" }
{ "_id" : ObjectId("5ca2cc73056f79a88b52e4dc"), "qwe" : "2" }
```

```
> db.test.find({'qwe':{$type:'string'}}).limit(1)
{ "_id" : ObjectId("5c9b423405b418e98b4285b7"), "qwe" : "1" }
> db.test.find({'qwe':{$type:'string'}}).limit(1).skip(1)
{ "_id" : ObjectId("5ca2cc73056f79a88b52e4dc"), "qwe" : "2" }
```

sort方法

sort() 方法可以通过参数指定排序的字段，并使用 1 和 -1 来指定排序的方式，其中 1 为升序排列，而 -1 是用于降序排列。

sort()方法基本语法如下所示：

```
>db.COLLECTION_NAME.find().sort({KEY:1})
```

实例如下：

```
> db.col.find().sort({'by':-1}).pretty()
{
    "_id" : ObjectId("56066542ade2f21f36b0313a"),
    "title" : "PHP 教程",
    "description" : "PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。",
    "by" : "菜鸟教程",
    "url" : "http://www.runoob.com",
    "tags" : [
        "php"
    ],
    "likes" : 200
}
{
    "_id" : ObjectId("5606654fade2f21f36b0313c"),
    "title" : "MongoDB 教程",
    "description" : "MongoDB 是一个 Nosql 数据库",
    "by" : "菜鸟教程",
    "url" : "http://www.runoob.com",
    "tags" : [
        "mongodb"
    ],
    "likes" : 100
}
{
    "_id" : ObjectId("5c9c9699dc1d85c788c52e96"),
    "title" : "mongoDB",
    "description" : "MongoDB 是一个 Nosql 数据库",
    "by" : "lyh",
    "tags" : [
        "mongodb",
        "database",
        "NoSQL"
    ],
```

```
    "likes" : 100
}
```