# Windows下安装

## 新建

|  |  |  |
| --- | --- | --- |
| 文件或目录名称 | 类型 |  |
| data | 数据目录 |  |
| log/log.log | 日志文件 |  |
| mongo.config | 配置文件 |  |
|  |  |  |

## mongo.config内容

|  |
| --- |
| ##数据目录  dbpath=C:/Program Files/MongoDB/Server/3.0/data  ##日志文件  logpath=C:/Program Files/MongoDB/Server/3.0/log/log.log |

## 执行mongod.exe命令指定mongo.config文件

到mongo的bin目录下执行命令

|  |
| --- |
| mongod.exe --config “E:\ruanjian\MongoDB\mongo.config” |

## 执行mongo.exe命令

会默认连接到test库

# 简介

## 与rdbms的对应关系

|  |  |
| --- | --- |
| **RDBMS** | **MongoDB** |
| Database | Database |
| Table | Collection |
| Tuple/Row | Document |
| column | Field |
| Table Join | Embedded Documents |
| Primary Key | Primary Key (Default key \_id provided by mongodb itself) |
| **数据库服务器和客户端** | |
| Mysqld/Oracle | mongod |
| mysql/sqlplus | mongo |

## 示例文档

\_id是一个12字节的十六进制数，保证每一份文件的唯一性。您可以提供\_id同时插入文档。如果没有提供，那么MongoDB的每个文档提供了一个独特的ID。这12个字节，前4个字节为当前时间戳，未来3个字节的机器ID，接下来的2个字节的进程id MongoDB的服务器及剩余3个字节是简单的增量值。

|  |
| --- |
| {  \_id: ObjectId(7df78ad8902c)  title: 'MongoDB Overview',  description: 'MongoDB is no sql database',  by: 'yiibai.com',  url: 'http://www.yiibai.com',  tags: ['mongodb', 'database', 'NoSQL'],  likes: 100,  comments: [  {  user:'user1',  message: 'My first comment',  dateCreated: new Date(2011,1,20,2,15),  like: 0  },  {  user:'user2',  message: 'My second comments',  dateCreated: new Date(2011,1,25,7,45),  like: 5  }  ]  } |

## MongoDB比RDBMS的优势

* 架构：MongoDB是文档型数据库，其中一个集合保存不同的不同的文件。字段的数量，内容和该文件的大小可以是不同于从一个文件复制到另一个；
* 一个单一的对象是结构清晰；
* 没有复杂的连接；
* 深查询能力。 MongoDB支持动态查询使用基于文档的查询语言，如SQL几乎一样强大的文件；
* 调优；
* 易于规模化：MongoDB是易于扩展；
* 不需要数据库对象的应用程序对象转换/映射；
* 使用内部存储器存储（窗口）工作组，从而实现更快的数据存取；

## 为什么要使用MongoDB

* JSON风格文件的形式，面向文档存储：数据存储；
* 对任何属性可索引；
* 复制和高可用性；
* 自动分片；
* 丰富的查询；
* 快速就地更新；

## 应该在哪里使用MongoDB

* 大数据；
* 内容管理和交付；
* 移动和社交基础设施；
* 用户数据管理；
* 数据平台；

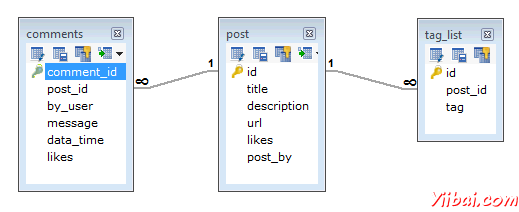
## 数据模型

假设一个客户端需要一个数据库设计，他的博客网站看看RDBMS 和 MongoDB 架构设计之间的差异。网站有以下要求。

* 每一个岗位都有独特的标题，描述和网址。
* 每一个岗位可以有一个或多个标签。
* 每一个岗位都有其出版商总数喜欢的名称。
* 每一个岗位有意见连同他们的名字，消息，数据的时间和喜欢的用户。
* 在每个岗位上，可以是零个或多个注释。

### 上述要求在RDBMS模式设计

将有至少三个表。



### 在MongoDB 模式设计将发布一个集合

并具有以下结构：

|  |
| --- |
| {  \_id: POST\_ID  title: TITLE\_OF\_POST,  description: POST\_DESCRIPTION,  by: POST\_BY,  url: URL\_OF\_POST,  tags: [TAG1, TAG2, TAG3],  likes: TOTAL\_LIKES,  comments: [  {  user:'COMMENT\_BY',  message: TEXT,  dateCreated: DATE\_TIME,  like: LIKES  },  {  user:'COMMENT\_BY',  message: TEXT,  dateCreated: DATE\_TIME,  like: LIKES  }  ]  } |

# 常用命令

## 创建、使用据库

use

例如：use mydb

## 删除数据库

db.dropDatabase()

## 显示数据库

show dbs

## 创建集合（表）

db.createCollection(name, options)

name：要创建的集合名称

options：如下：

|  |  |  |
| --- | --- | --- |
| 命令 | 类型 | 备注 |
| capped | Boolean | （可选）如果为true，则启用封顶集合。封顶集合是固定大小的集合，会自动覆盖最早的条目，当它达到其最大大小。如果指定true，则需要也指定尺寸参数。 |
| autoIndexID | Boolean | （可选）如果为true，自动创建\_id字段的索引  默认值是false。 |
| size | number | （可选）指定最大大小字节封顶集合。如果封顶如果是 true，那么你还需要指定这个字段。 |
| max | number | （可选）指定封顶集合允许文件的最大数量。 |
|  |  |  |

* 例如：

db.createCollection("mycollection")

db.createCollection("mycol", { capped : true, autoIndexID : true, size : 6142800, max : 10000 } )

db.lzg.insert({"name" : "yiibai"}) 直接创建lzg集合

## 显示集合（表）

show collections

## 删除集合（表）

db.COLLECTION\_NAME.drop()

## 插入文档（记录）

* 新加一个文档

db.COLLECTION\_NAME.insert(document)

db.COLLECTION\_NAME.insert([document1,document2,document3,……]) //插入多个

## 修改文档（记录）

db.COLLECTION\_NAME.update(where\_sql , UPDATED\_DATA);

* 例如：

mycol集合内容如下：

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}  { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}  { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} |

把title为MongoDB Overview的文档改为New MongoDB Tutorial，如下：

|  |
| --- |
| db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}}) |

**注：MongoDB默认将只更新单一的文件，来更新多个你需要设置参数置'multi' 为true，如下：**

|  |
| --- |
| db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}},{multi:true}) |

## 插入、修改文档（记录）

* 新加或更新一个文档

db.lzg.save({"\_id" : ObjectId("55379c89b680e19981df28c0"), "name" : "lzgsss" }

\_id存在时，就会做更新操作；

\_id不存在时，就会做新加操作；

## 删除文档（记录）

db.COLLECTION\_NAME.remove(DELETION\_CRITERIA)

db.COLLECTION\_NAME.remove(DELETION\_CRITERIA,1) //只删除第一条记录

db.COLLECTION\_NAME.remove() //删除所有，相当于SQL的truncate命令

* 如：

|  |
| --- |
| db.mycol.remove({'title':'MongoDB Overview'}) |

删除title为'MongoDB Overview'的文档；

## 查询文档（记录）

db.collection\_name.find();

db.collection\_name.findOne(); //返回一个文档

db.collection\_name.find().pretty(); //格式化的方式

### 比较符（=,!=,>,>=,<,<=）

|  |  |  |  |
| --- | --- | --- | --- |
| **操作** | **语法** | **例子** | **RDBMS 等同** |
| 等于 | {<key>:<value>} | db.mycol.find({"by":"tutorials point"}).pretty() | where by = 'tutorials point' |
| 小于 | {<key>:{$lt:<value>}} | db.mycol.find({"likes":{$lt:50}}).pretty() | where likes < 50 |
| 小于等于 | {<key>:{$lte:<value>}} | db.mycol.find({"likes":{$lte:50}}).pretty() | where likes <= 50 |
| 大于 | {<key>:{$gt:<value>}} | db.mycol.find({"likes":{$gt:50}}).pretty() | where likes > 50 |
| 大于等于 | {<key>:{$gte:<value>}} | db.mycol.find({"likes":{$gte:50}}).pretty() | where likes >= 50 |
| 不等于 | {<key>:{$ne:<value>}} | db.mycol.find({"likes":{$ne:50}}).pretty() | where likes != 50 |

### and符

db.mycol.find({key1:value1, key2:value2}).pretty()；

### or符

db.mycol.find(

{

$or: [

{key1: value1}, {key2:value2}

]

}

).pretty();

### and 和 or一起使用

db.mycol.find("age": {$gt:10}, $or: [{"name": "yiibai"}, {"title": "MongoDB Overview"}] }).pretty();

等效于 SQL where子句 为 'where age>10 AND (name = 'yiibai' OR title = 'MongoDB Overview')'

|  |  |  |
| --- | --- | --- |
| 命令 | 描述 | 备注 |
| db.stats() | 统计信息 |  |
| db | 显示当前使用的数据库 |  |
| use | 创建并使用数据库，无此数据库时创建，有则使用 | use mydb |
| db.dropDatabase() | 删除某数据库 | use mydb  db.dropDatabase() |
| show dbs | 显示当前所有的数据库 |  |
|  |  |  |
|  |  |  |
|  |  |  |

### 投影（查询个别属性）

db.COLLECTION\_NAME.find({},{KEY:1})

mongodb 投影意思是只选择必要的数据而不是选择一个文件的数据的整个。如果一个文档有5个字段，需要显示只有3个，然后选择其中只有3个字段。

MongoDB 的find()方法，在 MongoDB查询 文档解释接受第二个可选参数是要检索的字段列表。在MongoDB中，当执行find()方法，那么它会显示一个文档所有字段。要限制这一点，需要设置的字段列表值1或0。 1用来显示字段而0是用来隐藏字段。

* 如：

集合myycol具有以下的数据

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}  { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}  { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Yiibai Overview"} |

db.mycol.find({},{"title":1,\_id:0});

### 限制记录（分页）

* db.COLLECTION\_NAME.find().limit(NUMBER) //只显示符合条件中的几条记录
* db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER) // skip() 也接受数字类型的参数，并使用跳过的文档数

要限制 MongoDB 中的记录，需要使用 limit() 方法。 limit() 方法接受一个数字型的参数，这是要显示的文档数。

### 排序

* db.COLLECTION\_NAME.find().sort({KEY:1})

要在 MongoDB 中的文档进行排序，需要使用sort()方法。 sort() 方法接受一个文档，其中包含的字段列表连同他们的排序顺序。要指定排序顺序1和-1。 1用于升序排列，而-1用于降序。

### 索引

* db.COLLECTION\_NAME.ensureIndex({KEY:1})

这里关键是要在其中创建索引，1是按升序排列的字段名称。要创建降序索引，需要使用-1。

如果没有索引，MongoDB 必须扫描每一个文档的集合，要选择那些文档相匹配的查询语句。这种扫描的效率非常低，会要求 mongod 做大数据量的处理。

索引是一种特殊的数据结构，存储设置在一个易于遍历形式的数据的一小部分。索引存储一个特定的字段或一组字段的值，在索引中指定的值的字段排列的。

其它参数

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **描述** |
| background | Boolean | 在后台建立索引，以便建立索引并不能阻止其他数据库活动。指定true建立在后台。默认值是 false. |
| unique | Boolean | 创建唯一索引，以便收集不会接受插入索引键或键匹配现有的值存储在索引文档。指定创建唯一索引。默认值是 false. |
| name | string | 索引的名称。如果未指定，MongoDB中都生成一个索引名索引字段的名称和排序顺序串联. |
| dropDups | Boolean | 创建一个唯一索引的字段，可能有重复。 MongoDB的索引只有第一次出现的一个键，从集合中删除的所有文件包含该键的后续出现的。指定创建唯一索引。默认值是 false. |
| sparse | Boolean | 如果为true，指数只引用文档指定的字段。这些索引使用更少的空间，但在某些情况下，特别是各种不同的表现。默认值是 false. |
| expireAfterSeconds | integer | 指定一个值，以秒为TTL控制多久MongoDB的文档保留在此集合. |
| v | index version | 索引版本号。默认的索引版本取决于mongodb 运行的版本在创建索引时. |
| weights | document | 权重是从1到99999范围内的数，表示该字段的意义，相对于其他的索引字段分数. |
| default\_language | string | 对于文本索引时，决定停止词和词干分析器和标记生成规则列表的语言。默认值是 english. |
| language\_override | string | 对于文本索引时，指定的名称在文档中包含覆盖默认的语言，语言字段中。默认值是语言。 |

### 聚合

db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

|  |  |  |
| --- | --- | --- |
| **表达式** | **描述** | **实例** |
| $sum | 总结从集合中的所有文件所定义的值. | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : "$likes"}}}]) |
| $avg | 从所有文档集合中所有给定值计算的平均. | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$avg : "$likes"}}}]) |
| $min | 获取集合中的所有文件中的相应值最小. | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$min : "$likes"}}}]) |
| $max | 获取集合中的所有文件中的相应值的最大. | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$max : "$likes"}}}]) |
| $push | 值插入到一个数组生成文档中. | db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$push: "$url"}}}]) |
| $addToSet | 值插入到一个数组中所得到的文档，但不会创建重复. | db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$addToSet : "$url"}}}]) |
| $first | 根据分组从源文档中获取的第一个文档。通常情况下，这才有意义，连同以前的一些应用 “$sort”-stage. | db.mycol.aggregate([{$group : {\_id : "$by\_user", first\_url : {$first : "$url"}}}]) |
| $last | 根据分组从源文档中获取最后的文档。通常，这才有意义，连同以前的一些应用 “$sort”-stage. | db.mycol.aggregate([{$group : {\_id : "$by\_user", last\_url : {$last : "$url"}}}]) |

* 如：

|  |
| --- |
| {  \_id: ObjectId(7df78ad8902c)  title: 'MongoDB Overview',  description: 'MongoDB is no sql database',  by\_user: 'yiibai point',  url: 'http://www.yiibai.com',  tags: ['mongodb', 'database', 'NoSQL'],  likes: 100  },  {  \_id: ObjectId(7df78ad8902d)  title: 'NoSQL Overview',  description: 'No sql database is very fast',  by\_user: 'yiibai point',  url: 'http://www.yiibai.com',  tags: ['mongodb', 'database', 'NoSQL'],  likes: 10  },  {  \_id: ObjectId(7df78ad8902e)  title: 'Neo4j Overview',  description: 'Neo4j is no sql database',  by\_user: 'Neo4j',  url: 'http://www.neo4j.com',  tags: ['neo4j', 'database', 'NoSQL'],  likes: 750  } |

根据by\_user字段查询数量

|  |
| --- |
| db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : 1}}}])  {  "result" : [  {  "\_id" : "yiibai point",  "num\_tutorial" : 2  },  {  "\_id" : "yiibai point",  "num\_tutorial" : 1  }  ],  "ok" : 1  } |

* 相当于SQL：

select by\_user, count(\*) from mycol group by by\_user

与java

# 数据类型

String : 这是最常用的数据类型来存储数据。在MongoDB中的字符串必须是有效的UTF-8。

Integer : 这种类型是用来存储一个数值。整数可以是32位或64位，这取决于您的服务器。

Boolean : 此类型用于存储一个布尔值 (true/ false) 。

Double : 这种类型是用来存储浮点值。

Min/ Max keys : 这种类型被用来对BSON元素的最低和最高值比较。

Arrays : 使用此类型的数组或列表或多个值存储到一个键。

Timestamp : 时间戳。这可以方便记录时的文件已被修改或添加。

Object : 此数据类型用于嵌入式的文件。

Null : 这种类型是用来存储一个Null值。

Symbol : 此数据类型用于字符串相同，但它通常是保留给特定符号类型的语言使用。

Date : 此数据类型用于存储当前日期或时间的UNIX时间格式。可以指定自己的日期和时间，日期和年，月，日到创建对象。

Object ID : 此数据类型用于存储文档的ID。

Binary data : 此数据类型用于存储二进制数据。

Code : 此数据类型用于存储到文档中的JavaScript代码。

Regular expression : 此数据类型用于存储正则表达式