**Mongodb开发**

# 相关

## 主站

<https://www.mongodb.com/>

## 资源网站

<http://www.mongoing.com/> MongoDB中文社区 | 中文社区

<http://www.yiibai.com/mongodb/> MongoDB教程

<http://www.runoob.com/mongodb/mongodb-tutorial.html> MongoDB 教程

<http://mongoosejs.com/docs/middleware.html> **<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<**

## Ubuntu安装

安装后用：

启动服务 sudo service mongod start

链接服务器 mongo

**但是问题在于程序貌似无法读写（一直挂起） <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<**

## Win安装后还需要自定义启动

"D:\Program Files\MongoDB\Server\3.4\bin\mongod.exe" --dbpath "D:\Program Files\MongoDB\data" --port 27017



## 启动Express

<https://dashboard.daocloud.io/packages/18d186d7-ccfe-4221-943a-33f97c9723a7>

## 如何使用这个镜像管理我自己的 MongoDB 数据库

这个镜像在启动时依赖于下面几个环境变量来配置 MongoDB 数据库连接

* 「**MONGODB\_PORT\_27017\_TCP\_ADDR**」 MongoDB 数据库 IP
* 「**MONGODB\_PORT\_27017\_TCP\_PORT**」 MongoDB 数据库端口
* 「**MONGODB\_INSTANCE\_NAME**」 MongoDB 数据库名称
* 「**MONGODB\_USERNAME**」 MongoDB 数据库用户名
* 「**MONGODB\_PASSWORD**」 MongoDB 数据库密码

|  |
| --- |
| 创建*Dockerfile* 文件并进行构建自己的MongoDB-Express  FROM mongo-express  ENV MONGODB\_PORT\_27017\_TCP\_ADDR=127.0.0.1  ENV MONGODB\_PORT\_27017\_TCP\_PORT=27017  ENV MONGODB\_INSTANCE\_NAME=mean-development  ENV MONGODB\_USERNAME=admin  ENV MONGODB\_PASSWORD=pass  构建自己的mongo-express-mean  docker build -t mongo-express-mean .  运行  sudo docker run \  -p 8081:8081 \  --name db-express-mean \  -d mongo-express-mean  <https://dashboard.daocloud.io/packages/609cfb6f-416c-4608-82c1-90554106ea69>  $ docker run -it --rm \  --name mongo-express \  --link web\_db\_1:mongo \  -p 8081:8081 \  -e ME\_CONFIG\_OPTIONS\_EDITORTHEME="ambiance" \  -e ME\_CONFIG\_BASICAUTH\_USERNAME="user" \  -e ME\_CONFIG\_BASICAUTH\_PASSWORD="fairly long password" \  mongo-express |

sudo docker run \

-p 27017:27017 \

-p 8081:8081 \

--name db-express \

-v /home/demo/nodejs\_projects/docker\_meanjs/mongodb-express\_data:/data/db \

-d mongo-express

## MongoDB 基础（二）mongodb 与 T-SQL 对比

|  |  |
| --- | --- |
| **SQL Terms/Concepts** | [**MongoDB**](http://lib.csdn.net/base/mongodb)**Terms/Concepts** |
| database | database |
| table | Collection（集合） |
| row | document（文档） |
| column | Key（键） |
| Value | Value（值） |
| index | index |
| table joins | embedded documents and linking |
| primary key | primary key |
| Specify any unique column or column combination as primary key. | In MongoDB, the primary key is automatically set to the \_id field. |
| aggregation (e.g. group by) | aggregation pipeline |

|  |  |
| --- | --- |
| **Help** | **查看帮助（mongodb 中注意大小写）** |
| db.help() | help on db methods |
| db.mycoll.help() | help on collection methods |
| sh.help() | sharding helpers |
| rs.help() | replica set helpers |
| help admin | administrative help |
| help connect | connecting to a db help |
| help keys | key shortcuts |
| help misc | misc things to know |
| help mr | mapreduce |
| show dbs | show database names |
| show collections | show collections in current database |
| show users | show users in current database |
| show profile | show most recent system.profile entries with time >= 1ms |
| show logs | show the accessible logger names |
| show log [name] | prints out the last segment of log in memory, 'global' is default |
| use <db\_name> | 切换/创建数据库 |
| db.foo.find() | list objects in collection foo |
| db.foo.find( { a : 1 } ) | list objects in foo where a == 1 |
| it | result of the last line evaluated; use to further iterate |
| DBQuery.shellBatchSize = x | set default number of items to display on shell |
| exit | quit the mongo shell |

|  |  |
| --- | --- |
| [**数据库**](http://lib.csdn.net/base/mysql) | |
| show dbs | 查看所有数据库 |
| use mydb | 设置为当前数据库 |
| db | 查看当前数据库名称 |
| db.createCollection("tab")  db.tab.insert({"id":1}) | 创建collection  插入数据也会被创建 |
| db.copyDatabase("mydb", "newmydb", "127.0.0.1") | 复制为另一个新的数据库 |
| db.dropDatabase() | 删除数据库 |

|  |  |
| --- | --- |
| **Collection定义操作（集合必须以字母或下划线开头）** | |
| db.createCollection("tab",{size:1024,max:1000,capped:true})  db.tab.insert({"id":1}) | 创建collections 并做限制（如创建“tab”）  (插入数据也会创建collections) |
| show collections | 查看当前数据库所有collections |
| db.tab.drop() | 删除collections |
| db.tab.update({},{$rename:{"name":"newname"}},false,true) | 更改所有行的字段名name为newname |
| db.getCollection('tab').renameCollection('newtab')  db.newtab.renameCollection('tab') | 更改collection名称 |

|  |  |
| --- | --- |
| **Collection 数据操作** | |
| db.tab.save({"id":2})  db.tab.insert({"id":3})  db.tab.insert([{ size: "S", qty: 25 }, { size: "M", qty: 50 }])  db.tab.insert({ array :[{ size: "S", qty: 25 }, { size: "M", qty: 50 } ]}) | 插入数据到collections  （会自动生成唯一列 “\_id”）  插入多行（相当： insert into tab values(“S”,1), (“M”,50)）  字段中插入子集 |
| db.tab.update({"id":1},{$set:{"id":5}}) | 更改数据（id值由1更改为5） |
| db.tab.update({ id:2},{id:6, name:'E'}) | 将id=2的行更新为新的行 |
| db.tab.update({"id":1},{$unset:{ name:'120'}}) | 删除一个键值对（删除id=2中的字段name） |
| db.tab.update({"id":1},{$push:{ name:'C'}}) | 往id=1的行 字段为name数组中插入元素’C’ |
| db.tab.update({"id":1},{$pull:{ name:'C'}}) | 从id=1的行 字段为name数组中删除所有元素’C’ |
| db.tab.remove({"id":3})  db.tab.remove({"id":3},1)  db.tab.remove({}) | 删除id=3的所有记录  删除id=3的第一条记录  删除所有记录 |

|  |  |
| --- | --- |
| Collection查询数据操作 | |
| Select \* from tab | db.tab.find()  db.tab.find({}) |
| Select id from tab | db.tab.find({},{"id":1})   #({条件},{字段:0/1})  db.tab.find({},{"\_id":0,"id":1})  db.tab.aggregate({ $project : {id : 1 }});  db.tab.find({id:{$exists:1}}); |
| Select \* from tab where id=1 | db.tab.find( { id :1}) |
| Select id from tab where id=1 | db.tab.find({id :1},{"\_id":0,"id":1}) |
| Select \* from tab where id<>1 | db.tab.find({id:{$ne:1}}) |
| Select \* from tab where id>2 | db.tab.find( { id: { $gt: 2} } ) |
| Select \* from tab where id<3 | db.tab.find( { id: { $lt: 3} } ) |
| Select \* from tab where id>=2 | db.tab.find( { id: { $gte: 2 } } ) |
| Select \* from tab where id<=3 | db.tab.find( { id: { $lte: 3} } ) |
| Select \* from tab where id = 1 or id = 2 | db.tab.find({$or:[{id:3},{id:2}]}) |
| Select \* from tab where id < 2 or id >4 | db.tab.find({$or:[{id:{$gt:4}},{id:{$lt:2}}]}) |
| Select \* from tab where id in (1,2) | db.tab.find( { id: { $in: [ 1, 2 ] } } ) |
| Select \* from tab where id not in (2,3) | db.tab.find({id:{"$nin":[2,3]}}) |
| Select \* from tab where id between 2 and 3  Select \* from tab where id >= 2 and id <= 3 | db.tab.find({$and:[{id:{$gte:2}},{id:{$lte:5}}]}) |
| Select \* from tab where id = 1 and name = ‘kk’ | db.tab.find({id:2, name:'kk'}) |
| Select distinct id from tab | db.tab.distinct("id");  db.runCommand ( { distinct: "tab", key: "id" } ) |
| Select distinct id from tab where name = ‘A’ | db.runCommand({distinct:'tab',key:'id',query:{name:'A'}}) |
| Select \* from tab where name like ‘%A%’ | db.tab.find({ name:{$regex:"A"}})  db.tab.find({ name:/A/}) |
| Select \* from tab order by id asc | db.tab.find().sort({id:1}) |
| Select \* from tab order by id desc | db.tab.find().sort({id:-1}) |
| Select  top 5 \* from tab | db.tab.find().limit(5) |
| 跳过前m条记 | db.tab.find().skip(2)  db.tab.aggregate({ $skip : 2 }); |
| 跳过前m条记录，从m+1开始取n条 | db.tab.find().skip(2).limit(3) |
| 除了指定的列，其他列都显示 | db.tab.find({id:null})  db.tab.find({},{"\_id":0}) |
| 查找字段id为string类型的行(参考下表格) | db.tab.find({ id: {$type: 2}}); |
|  |  |
| select name,sum(id) as sumid  from tab  where id >0 group by name | db.tab.group({  key:{ "name":true}  # group by name  ,cond:{id:{ $gt:0}}   # where id >0  ,reduce:function(obj,prev) #聚合函数  { prev.sumid += obj.id; } #函数逻辑，累加id  , initial: {sumid: 0 }})  #初始化 |
| Select sum(\*) from tab | db.tab.group({key:{},cond:{}  ,reduce:function(obj,prev)  { prev.sumid += obj.id; },initial: {sumid: 0 }}); |
| Select sum(\*) as newcol from tab | db.tab.aggregate([{$group:{\_id:"$by\_user",newcol:{$sum:"$id"}}}]) |
| Select count(\*) from tab | db.tab.count()  或者 db.tab.find().count() |
| Select count(\*) from tab | db.tab.group({key:{},cond:{},  reduce:function(obj,prev)  { prev.sumid += 1; },initial: {sumid: 0 }}); |
| Select avg(\*) from tab | db.tab.aggregate([{$group:{\_id:"$by\_user",newcol:{$avg:"$id"}}}]) |
| Select max(\*) from tab | db.tab.find().sort({id:-1}).limit(1)  db.tab.aggregate([{$group:{\_id:"$by\_user",newcol:{$max:"$id"}}}]) |
| Select min(\*) from tab | db.tab.find().sort({id:1}).limit(1)  db.tab.aggregate([{$group:{\_id:"$by\_user",newcol:{$min:"$id"}}}]) |
|  |  |
| #元素查询  #db.tab.insert({ratings: [ 5, 8, 9 ] })  db.tab.find({ ratings: [ 5, 8, 9 ] } )   #查找匹配的数组  db.tab.find({ ratings: 5 })     #查找元素中含“5”的记录  db.tab.find({ ratings:{$elemMatch:{$gt:8,$lt:10}}})   #元素匹配查找  #内嵌文档  #db.tab.insert({producer:{company: 'ABC',address: 'Street'}})  #db.tab.insert({producer:[{ company: 'ABC',address: 'Street'},{ company: 'KK',address: 'Street2'}] })  db.tab.find({producer:{company: 'ABC',address: 'Street'}})  db.tab.find({'producer.company': 'ABC'})  db.tab.find( { 'producer.0.address': 'Street'} )  #字段'producer的第一个元素的address=’ Street’ | |

|  |  |
| --- | --- |
| **类型描述** | **类型值** |
| Double | 1 |
| String | 2 |
| Object | 3 |
| Array | 4 |
| Binary data | 5 |
| Object id | 7 |
| Boolean | 8 |
| Date | 9 |
| Null | 10 |
| Regular expression | 11 |
| [**JavaScript**](http://lib.csdn.net/base/javascript) code | 13 |
| Symbol | 14 |
| JavaScript code with scope | 15 |
| 32-bit integer | 16 |
| Timestamp | 17 |
| 64-bit integer | 18 |
| Min key | 255 |
| Max key | 127 |

# 开发

## CRUD

增加： insert， save， update

查询： find

更新： update， save

删除： remove

**> db.posts.insert({"title":"Second Post", "user": "alice"})**

**> db.posts.update({**

**"user": "alice"**

**}, {**

**"title": "Second Post",**

**"user": "alice"**

**}, {**

**upsert: true**

**})**

**> db.posts.update({**

**"user": "alice"**

**}, {**

**$set: {**

**"title": "Second Post"**

**}**

**}, {**

**multi: true**

**})**

**> db.posts.save({"title":"Second Post", "user": "alice"})**

**> db.posts.save({**

**"\_id": ObjectId("50691737d386d8fadbd6b01d"),**

**"title": "Second Post",**

**"user": "alice"**

**});**

**> db.posts.find({ "user": "alice", "commentsCount": { $gt: 10 } })**

**> db.posts.find( { $or: [{ "user": "alice" }, { "user": "bob" }] })**

**> db.posts.remove({ "user": "alice" }, true)**

**查看列表**

[**http://127.0.0.1:3000/users**](http://127.0.0.1:3000/users)

**创建**

**curl -X POST -H "Content-Type: application/json" -d '{"firstName":"First", "lastName":"Last","email":"user@example.com","username":"username","password":"password"}' localhost:3000/users**

**curl -X POST -H "Content-Type: application/json" -d '{"firstName":"First", "lastName":"Last","email":"user@example.com","username":"username","password":"password","role":"Admin"}' localhost:3000/users**

**更新**

**curl -X PUT -H "Content-Type: application/json" -d '{"lastName":"Updated"}' localhost:3000/users/[id]**

**curl -X PUT -H "Content-Type: application/json" -d '{"lastName":"Updated"}' localhost:3000/users/58e8ebec928829e422768bf5**

**删除**

**curl -X DELETE localhost:3000/users/[id]**

**curl -X DELETE localhost:3000/users/58e8cffe928829e422768bf2**

## 6 Rules of Thumb for MongoDB Schema Design

<https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1> **<<<<<<<<EN**

<http://www.jianshu.com/p/bb0caddff60a> **<<<<<<<<CN** MongoDB数据库设计中6条重要的经验法则，part 1

## 1 ：N关系建模时

1. 数据对象嵌入：N代表的数量级很小，且N代表的实体不需要单独操作时
2. 1对象中用Array存放N的引用：N代表的数量级比较大，或者N代表的实体需要单独进行操作时
3. 在N对象中添加一个引用到1对象：N代表的数量级非常大时

## M:N关系时

在两个不同的集合的对象中分别加入对方的引用

## 种树结构Model Tree Structures in MongoDB

<https://docs.mongodb.com/manual/tutorial/model-tree-structures/>

1. 引用子节点
2. 引用父节点
3. 引用父节点 和 祖先节点数组（根到当前节点的经过的节点）
4. 路径（根到当前节点的经过的节点）
5. Model Tree Structures with Nested Sets 内嵌集合（分 左 和 右 ？）