# sMongoDB简介

1. MongoDB是为快速开发互联网Web应用 而设计的数据库系统
2. MongoDB的设计目标是极简、灵活、作为 Web应用栈的一部分。
3. MongoDB的数据模型是面向文档的，所谓 文档是一种类似于JSON的结构，简单理解 MongoDB这个数据库中存的是各种各样的 JSON。（BSON）

# MongoDB安装：windows

1. **注意**

MongoDB的版本偶数版本为稳定版，奇数 版本为开发版。例如：3.2、3.4为稳定版。

MongoDB对于32位系统支持不佳，所以 3.2版本以后没有再对32位系统的支持。

1. 官网地址

<https://docs.mongodb.com/manual/>

<https://www.mongodb.com/try/download/community?tck=docs_server>

国内镜像地址：

https://www.mongodb.org/dl/win32/

1. Windows安装步骤（4.2）

* 双击msiw文件进行安装。选择custom,选择安装路径，其他不用管
* 安装完毕后，在安装目录的bin下执行mongod.exe，启动mongo
* 在安装目录的bin下执行mongo.exe，连接mongo
* 可以修改安装目录的bin下的mongod.cfg配置mongo
* 注意：4.2版本官网说明，data、log等文件夹，安装时已经创建，无需手动创建。

# MongoDB安装：Liux

详见自己的博客。

# NoSQL Manager for MongoDB

1. 官网下载地址，选择免费版

<https://www.mongodbmanager.com/download>

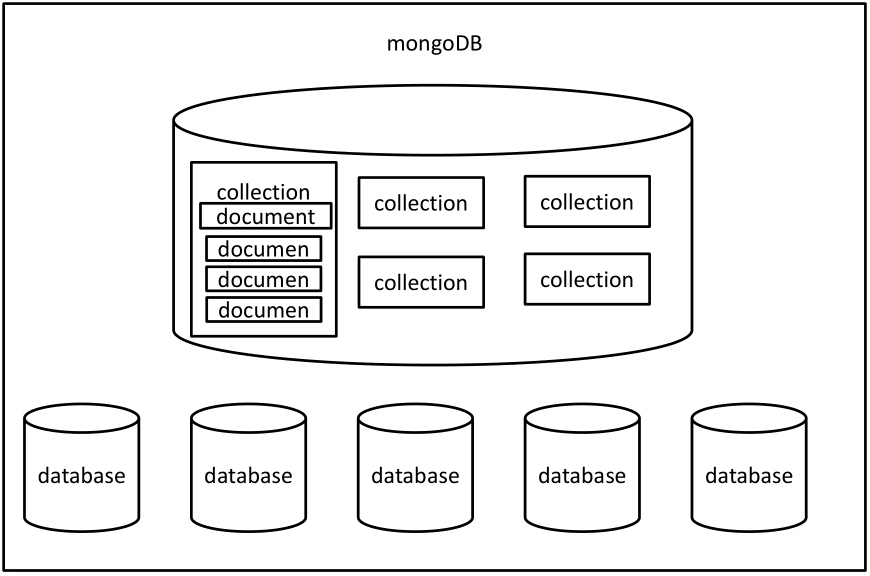
1. 默认安装即可
2. 连接mongoDB，不开启权限验证即可，直接连接。

# MongoDB基本使用

## 基本概念

* 数据库（database）
* 集合（collection）
* 文档（document）：

在MongoDB中，数据库和集合都不需要手动创建，当我们创建文档时，如果文档所在的集合或数据库不存在会自动创建数据库和集合



## 基本指令

1. show dbs（show databases）

显示当前的所有数据库

1. use 数据库名

进入到指定的数据库中,注意：如果没有数据库，也可以进入，但是mongo不会创建，而是在该数据库中插入文档，才会创建数据库、集合。

1. db

db表示的是当前所处的数据库

1. show collections

显示数据库中所有的集合

## 数据库的CRUD（参考官网）

### 插入

当我们向集合中插入文档时，如果不指定\_id属性，mongodb会自动为文档添加\_id，该属性是为了确保文档的唯一性。也可以自己维护（不建议）

db.<collection>.insert(doc)

* **例子(One)：**

向test数据库中的，students集合中插入一个新的学生对象

db.students.insert({name:"孙悟空",age:18,gender:"男"});

或者 db.students.insertOne({});

* **例子(Many)：**

向test数据库中的students集合插入多个学生数据

db.students.insert([{},{}]);

或者 db.students.insertMany({},{});

### 查询

1. db.collection.find()

* find()用来查询集合中所有符合条件的文档
* find()可以接收一个对象作为条件参数, {} 表示查询集合中所有的文档, {属性:值} 查询属性是指定值的文档.
* find()返回的是一个数组

**例子：在test库中查询students所有的记录**

db.stus.find({\_id:"hello"});

db.stus.find({age:16 , name:"白骨精"});

db.stus.find({age:28});

1. db.collection.findOne()

* 用来查询集合中符合条件的第一个文档
* findOne()返回的是一个文档对象
* 可以直接在查询结果上访问属性。

例子：db.stus.findOne({age:28});

1. db.collection.find({}).count()

* 查询所有结果的数量

例子：db.stus.find({}).count();

1. $in,类似于or

* 用来查询字段或条件的

例子：db.inventory.find({status: {$in: [ "A", "D" ]}});

1. Specify AND Conditions

* $lt

db.inventory.find( { status: "A", qty: { $lt: 30 } } )

1. Specify OR Conditions

* $or

db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )

1. Specify AND as well as OR Conditions

db.inventory.find( {

status: "A",

$or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]

} )

类似于：

SELECT \* FROM inventory WHERE status = "A" AND ( qty < 30 OR item LIKE "p%")

### 修改

1. db.collection.update(查询条件,新对象)

* update()默认情况下会使用新对象来替换旧的对象

db.stus.update({name:"沙和尚"},{age:28});

* 如果需要修改指定的属性，而不是替换需要使用“修改操作符”来完成修改

$set 可以用来修改文档中的指定属性

$unset 可以用来删除文档的指定属性

db.stus.update (

{"name" : "猪八戒"},

{

$set:{

address:"猪老庄"

}

}

);

db.stus.update(

{"\_id" : ObjectId("59c219689410bc1dbecc0709")},

{$unset:{

address:1

}}

)

* update()默认只会修改一个

1. db.collection.updateMany()

同时修改多个符合条件的文档

1. db.collection.updateOne()

修改一个符合条件的文档

1. db.collection.replaceOne()

替换一个文档

### 删除

1. db.collection.remove()

* 删除一个或多个，可以第二个参数传递一个true，则只会删除一个
* 如果传递一个空对象作为参数，则会删除所有的

1. db.collection.deleteOne()
2. db.collection.deleteMany()
3. db.collection.drop() 删除集合
4. db.dropDatabase() 删除数据库

注意：一般数据库中的数据都不会删除，所以删除的方法很少调用，一般会在数据中添加一个字段，来表示数据是否被删除

# 练习

## 练习1：

根据内嵌文档的属性查询数据；内嵌文档值为数组时添加元素/删除元素/不重复添加；批量插入（可以写脚本，先准备好数据，然后调用数据库一次插入）

|  |
| --- |
| //1.进入my\_test数据库  use my\_test  //2.向数据库的user集合中插入一个文档  db.users.insert({  username:"sunwukong"  });  //3.查询user集合中的文档  db.users.find();  //4.向数据库的user集合中插入一个文档  db.users.insert({  username:"zhubajie"  });    //5.查询数据库user集合中的文档  db.users.find();  //6.统计数据库user集合中的文档数量  db.users.find().count();  //7.查询数据库user集合中username为sunwukong的文档  db.users.find({username:"sunwukong"});  //8.向数据库user集合中的username为sunwukong的文档，添加一个address属性，属性值为huaguoshan  db.users.update({username:"sunwukong"},{$set:{address:"huaguoshan"}});  //9.使用{username:"tangseng"} 替换 username 为 zhubajie的文档  db.users.replaceOne({username:"zhubajie"},{username:"tangseng"});    //10.删除username为sunwukong的文档的address属性  db.users.update({username:"sunwukong"},{$unset:{address:1}});  //11.向username为sunwukong的文档中，添加一个hobby:{cities:["beijing","shanghai","shenzhen"] , movies:["sanguo","hero"]}  //MongoDB的文档的属性值也可以是一个文档，当一个文档的属性值是一个文档时，我们称这个文档叫做 内嵌文档  db.users.update({username:"sunwukong"},{$set:{hobby:{cities:["beijing","shanghai","shenzhen"] , movies:["sanguo","hero"]}}});  db.users.find();  //12.向username为tangseng的文档中，添加一个hobby:{movies:["A Chinese Odyssey","King of comedy"]}  db.users.update({username:"tangseng"},{$set:{hobby:{movies:["A Chinese Odyssey","King of comedy"]}}})  //13.查询喜欢电影hero的文档  //MongoDB支持直接通过内嵌文档的属性进行查询，如果要查询内嵌文档则可以通过.的形式来匹配  //如果要通过内嵌文档来对文档进行查询，此时属性名必须使用引号  db.users.find({'hobby.movies':"hero"});  //14.向tangseng中添加一个新的电影Interstellar  //$push 用于向数组中添加一个新的元素  //$addToSet 向数组中添加一个新元素 ， 如果数组中已经存在了该元素，则不会添加  db.users.update({username:"tangseng"},{$push:{"hobby.movies":"Interstellar"}});  db.users.update({username:"tangseng"},{$addToSet:{"hobby.movies":"Interstellar"}});  db.users.find();  //15.删除喜欢beijing的用户  db.users.remove({"hobby.cities":"beijing"});  //16.删除user集合  db.users.remove({});  db.users.drop();  show dbs;  //17.向numbers中插入20000条数据 7.2s  for(var i=1 ; i<=20000 ; i++){  db.numbers.insert({num:i});  }  db.numbers.find()  db.numbers.remove({});  //0.4s  var arr = [];  for(var i=1 ; i<=20000 ; i++){  arr.push({num:i});  }  db.numbers.insert(arr); |

## 练习2：

查询操作符（$lt,$gt……）;查询分页db.collection.find().sikp((页码 - 1)\*条数).limit(条数);

|  |
| --- |
| //18.查询numbers中num为500的文档  db.numbers.find({num:500})  //19.查询numbers中num大于5000的文档  db.numbers.find({num:{$gt:500}});  db.numbers.find({num:{$eq:500}});  //20.查询numbers中num小于30的文档  db.numbers.find({num:{$lt:30}});  //21.查询numbers中num大于40小于50的文档  db.numbers.find({num:{$gt:40 , $lt:50}});  //22.查询numbers中num大于19996的文档  db.numbers.find({num:{$gt:19996}});  //23.查看numbers集合中的前10条数据  db.numbers.find({num:{$lte:10}});  //limit()设置显示数据的上限  db.numbers.find().limit(10);  //在开发时，我们绝对不会执行不带条件的查询  db.numbers.find();  //24.查看numbers集合中的第11条到20条数据  /\*  分页 每页显示10条  1-10 0  11-20 10  21-30 20  。。。    skip((页码-1) \* 每页显示的条数).limit(每页显示的条数);    skip()用于跳过指定数量的数据    MongoDB会自动调整skip和limit的位置  \*/  db.numbers.find().skip(10).limit(10);  //25.查看numbers集合中的第21条到30条数据  db.numbers.find().skip(20).limit(10);  db.numbers.find().limit(10).skip(10); |

## 练习3：

更新操作符（$inc）；关联查询（写脚本）

|  |
| --- |
| //26.将dept和emp集合导入到数据库中  db.dept.find()  db.emp.find()  //27.查询工资小于2000的员工  db.emp.find({sal:{$lt:2000}});  //28.查询工资在1000-2000之间的员工  db.emp.find({sal:{$lt:2000 , $gt:1000}});  //29.查询工资小于1000或大于2500的员工  db.emp.find({$or:[{sal:{$lt:1000}} , {sal:{$gt:2500}}]});  //30.查询财务部的所有员工  //(depno)  var depno = db.dept.findOne({dname:"财务部"}).deptno;  db.emp.find({depno:depno});  //31.查询销售部的所有员工  var depno = db.dept.findOne({dname:"销售部"}).deptno;  db.emp.find({depno:depno});  //32.查询所有mgr为7698的所有员工  db.emp.find({mgr:7698})  //33.为所有薪资低于1000的员工增加工资400元  db.emp.updateMany({sal:{$lte:1000}} , {$inc:{sal:400}});  db.emp.find() |

# sort和投影

1. 查询文档时，默认情况是按照\_id的值进行排列（升序），sort()可以用来指定文档的排序的规则,sort()需要传递一个对象来指定排序规则 1表示升序 -1表示降序，limit skip sort 可以以任意的顺序进行调用

db.emp.find({}).sort({sal:1,empno:-1});

1. 在查询时，可以在第二个参数的位置来设置查询结果的 投影，用来设置只想查询的字段。（1:显示；0：不显示）

db.emp.find({},{ename:1 , \_id:0 , sal:1});

# 文档间关系

1. 一对一（one to one）

内嵌文档的方式设计

|  |
| --- |
| db.wifeAndHusband.insert([  {  name:"黄蓉",  husband:{  name:"郭靖"  }  },{  name:"潘金莲",  husband:{  name:"武大郎"  }  }  ]); |

1. 一对多(one to many) / (many to one)

//查找用户swk的订单

var user\_id = db.users.findOne({username:"zbj"}).\_id;

db.order.find({user\_id:user\_id});

在文档中增设属性来表达

|  |
| --- |
| db.users.insert([{  username:"swk"  },{  username:"zbj"  }]);  db.order.insert({    list:["牛肉","漫画"],  user\_id: ObjectId("59c47e35241d8d36a1d50de0")  }); |

1. 多对多(many to many)

文档中增设数组类型的属性，即可表达

|  |
| --- |
| db.teachers.insert([  {name:"洪七公"},  {name:"黄药师"},  {name:"龟仙人"}  ]);  db.stus.insert([  {  name:"郭靖",  tech\_ids:[  ObjectId("59c4806d241d8d36a1d50de4"),  ObjectId("59c4806d241d8d36a1d50de5")  ]  },{  name:"孙悟空",  tech\_ids:[  ObjectId("59c4806d241d8d36a1d50de4"),  ObjectId("59c4806d241d8d36a1d50de5"),  ObjectId("59c4806d241d8d36a1d50de6")  ]  }  ]) |