MongoDB使用

# 目录

[目录 1](#_Toc477784271)

[一、传统数据库技术回顾 2](#_Toc477784272)

[二、NoSQL 4](#_Toc477784273)

[三、MongoDB介绍安装 6](#_Toc477784274)

[四、数据库使用 9](#_Toc477784275)

[4.1 插入数据 10](#_Toc477784276)

[4.2 查找数据 10](#_Toc477784277)

[4.3 修改数据 14](#_Toc477784278)

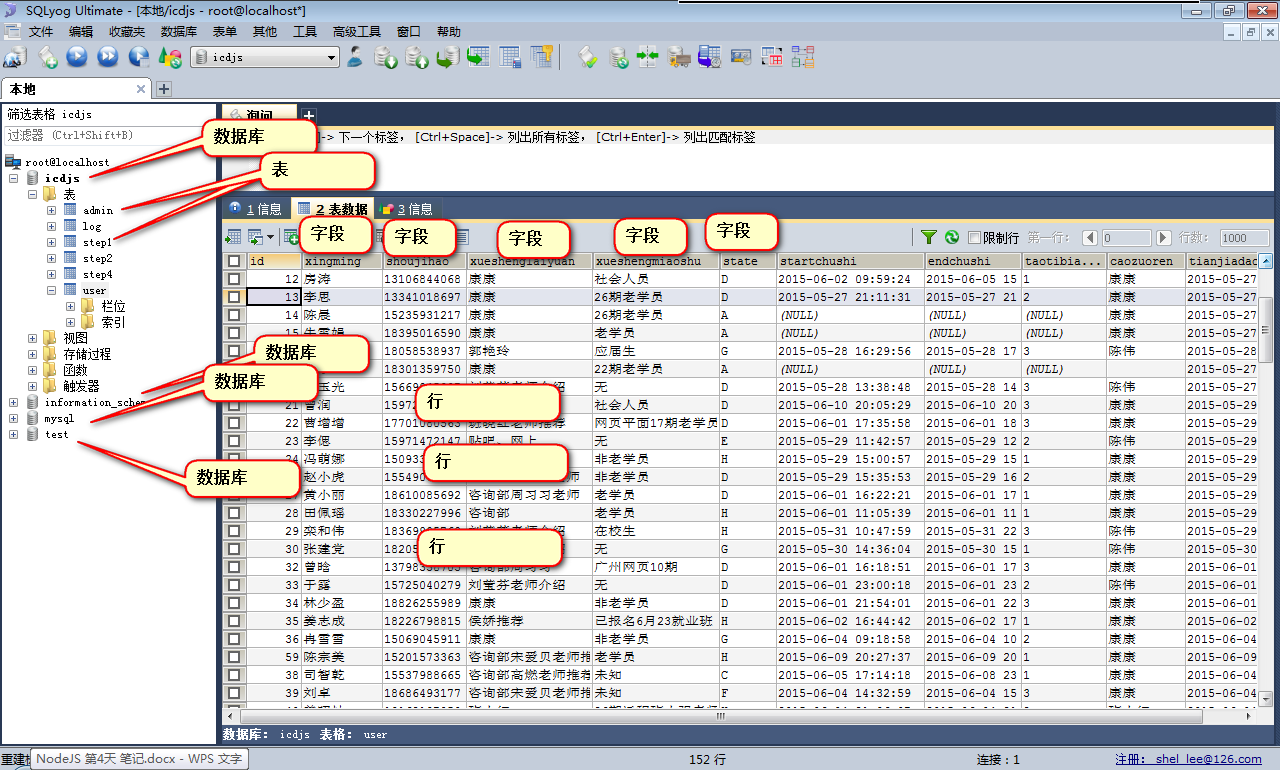
[4.4 删除数据 15](#_Toc477784279)

# 一、传统数据库技术回顾

数据库就是存储数据的，那么存储数据就用txt就行了啊，为什么要有数据库？

理由之1： 数据库有行、列的概念，数据有关系，数据不是散的。

老牌数据库，比如MySQL、SQL Server、Oracle、Access。这些数据库，我们管他们叫做结构型数据库。为什么？因为每个表中，都有明确的字段，每行记录，都有这些字段。不能有的行有，有的行没有。



理由二：数据库能够提供非常方便的接口，让增删改查操作变得简单

我们的老牌数据库，都无一例外的使用SQL语言，管理数据库。

SQL就是structure query language。

比如，查询所有女生： SELECT \* FROM step1 WHERE sex= '女';

再比如，查询所有女生，并且年龄20~24之间，且在北京：

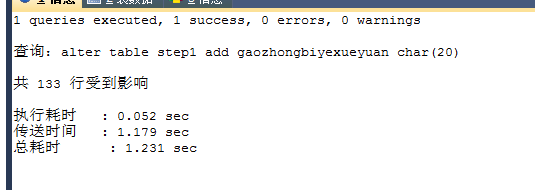
|  |
| --- |
| 1. SELECT \* FROM step1 WHERE xingbie = '女' AND nianling < 24 AND nianling >= 20 AND xianzaisuozaidi = '北京'; |

理由之三：数据库不能自己玩儿，要给向PHP、.net、jsp等语言提供接口。

用php这些语言，能够向数据库之中增删改查。

老牌数据库，都是结构型数据库，现在出了什么问题？

比如，我们现在想往一个已经有1000条数据的数据库中增加一个字段“高中信息”。



之前已经存在的数据，实际上不需要增加这个字段。因为这些用户已经填写完毕表单了，不需要再手机高中信息了。我们的意图就是在今后注册的用户，需要填写高中信息。但是，我们刚才说了，所谓的字段，是表的一个结构。所有的行都必须拥有，不能有的行有这个字段，有的行没有这个字段。

可想而知，大数据时代，数据库中有100万条数据都算少的。我们如果要动字段，时间太长。

**所以，字段这个东西，太不灵活。**

**数据不灵活。一个字段，需要是同样类型的数据。**不能一行记录是文本，一行记录是数字。

非结构型数据库NoSQL应运而生。

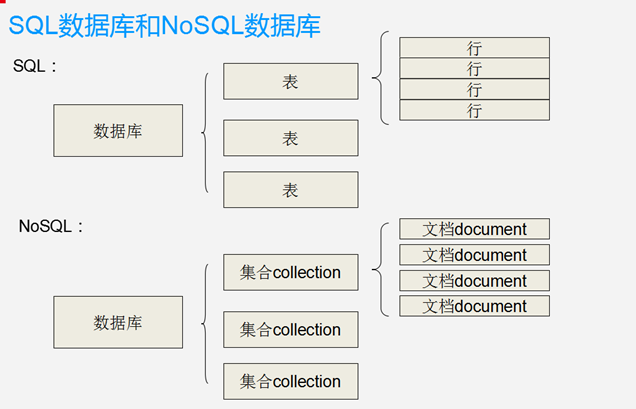
NoSQL是个怪胎，无法挑战老牌数据库，但是在大数据时代有自己的意义。

# 二、NoSQL

NoSQL(NoSQL = Not Only SQL )，意即“不仅仅是[SQL](http://baike.baidu.com/view/34.htm" \t "_blank)”，是一项全新的**[数据库](http://lib.csdn.net/base/mysql" \t "_blank" \o "MySQL知识库)**革命性运动，早期就有人提出，发展至2009年趋势越发高涨。NoSQL的拥护者们提倡运用非关系型的数据存储，相对于铺天盖地的[关系型数据库](http://baike.baidu.com/view/495393.htm" \t "_blank)运用，这一概念无疑是一种全新的思维的注入。

非结构型数据库。**没有行、列的概念。用JSON来存储数据。**

**集合就相当于“表”，文档就相当于“行”。**



文档就是JSON，上下文语境中，也是JavaScript范畴，所以我们的数据库也是JS范畴的东西，JS全栈。



因此，我们总结NoSQL数据库在以下的这几种情况下比较适用：1、数据模型比较简单；2、需要灵活性更强的IT系统；3、对数据库性能要求较高；4、不需要高度的数据一致性；5、对于给定key，比较容易映射复杂值的环境。

# 三、MongoDB介绍安装

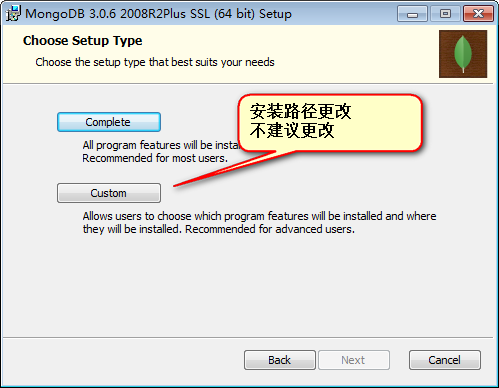
**MongoDB属于NoSql的一种，且是属于NoSql中的基于分布式文件存储的文档型数据库。由C++语言编写，旨在为WEB应用提供可扩展的高性能数据存储解决方案。**

**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。它支持的数据结构非常松散，是类似json的bson（是一种类json的一种二进制形式的存储格式，简称Binary JSON）格式，因此可以存储比较复杂的数据类型。**Mongo最大的特点是他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，而且还支持对数据建立索引。**

官网：<https://www.mongodb.com/>

手册：<https://docs.mongodb.org/manual/>

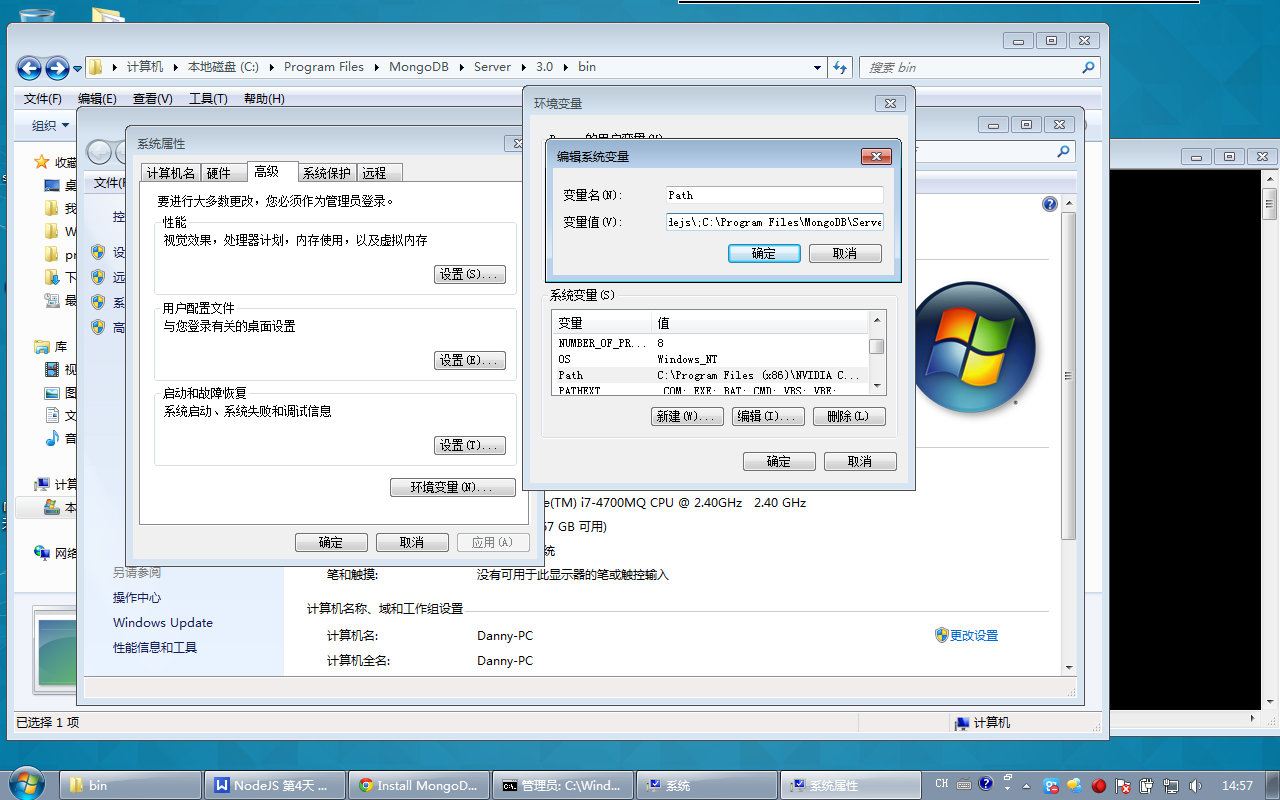
win7系统需要安装补丁，KB2731284。



此时，我们看一下装好的文件夹：

**1.下一步下一步安装**

2. C:\Program Files\MongoDB\Server\3.0\bin 加入到系统的path环境变量中



那么我们就能在系统的任何盘符，使用mongo命令了：

4.新建一个文件夹 不能有中文和空格

服务端：mongod 开启数据库服务 mongod --dbpath C:\mongodb

客户端：mongo 使用数据库

开机命令：



--dbpath就是选择数据库文档所在的文件夹。

也就是说，mongoDB中，真的有物理文件，对应一个个数据库。U盘可以拷走。

一定要保持，开机这个CMD不能动了，不能关，不能ctrl+c。 一旦这个cmd有问题了，数据库就自动关闭了。

所以，**应该再开一个cmd**。输入



那么，运行环境就是mongo语法了。

列出所有数据库：

|  |
| --- |
| 1. show dbs |

使用某个数据库

|  |
| --- |
| 1. use 数据库名字 |

如果想新建数据库，也是use。use一个不存在的，就是新建。

查看当前所在数据库

|  |
| --- |
| 1. db |

插入数据：

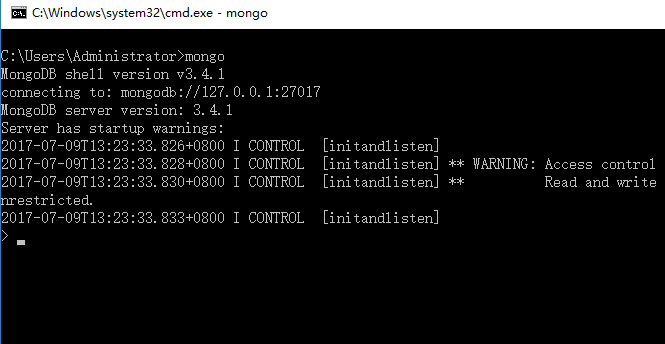


# 四、数据库使用

要管理数据库，必须先开启服务，开启服务使用**mongod --dbpath c:\mongo**



**管理数据库：**mongo (一定要在新的cmd中输入)



清屏：

|  |
| --- |
| cls |

查看所有数据库列表

|  |
| --- |
| show dbs |

使用数据库、创建数据库

|  |
| --- |
| use student |

如果真的想把这个数据库创建成功，那么必须插入一个数据。

数据库中不能直接插入数据，只能往集合(collections)中插入数据。不需要创建集合，只需要写点语法：

|  |
| --- |
| db.student.insert({“name”:”xiaoming”}); |

db.student 系统发现student是一个陌生的集合名字，所以就自动创建了集合。

显示当前的数据集合（mysql中叫表）

|  |
| --- |
| show collections |

删除数据库，删除当前所在的数据库

|  |
| --- |
| db.dropDatabase(); |

删除集合，删除指定的集合 删除表

|  |
| --- |
| 删除集合 db.COLLECTION\_NAME.drop() |

## 4.1 插入数据

插入数据，随着数据的插入，数据库创建成功了，集合也创建成功了。

|  |
| --- |
| 1. **db.表名.insert({"name":"zhangsan"}); student集合名称（表）** |

## 4.2 查找数据

**1、查询所有记录**

db.userInfo.find();

相当于：select\* from userInfo;

**2、查询去掉后的当前聚集集合中的某列的重复数据**

db.userInfo.distinct("name");

会过滤掉name中的相同数据

相当于：select distict name from userInfo;

**3、查询age = 22的记录**

db.userInfo.find({"age": 22});

相当于： select \* from userInfo where age = 22;

**4、查询age > 22的记录**

db.userInfo.find({age: {$gt: 22}});

相当于：select \* from userInfo where age >22;

**5、查询age < 22的记录**

db.userInfo.find({age: {$lt: 22}});

相当于：select \* from userInfo where age <22;

**6、查询age >= 25的记录**

db.userInfo.find({age: {$gte: 25}});

相当于：select \* from userInfo where age >= 25;

**7、查询age <= 25的记录**

db.userInfo.find({age: {$lte: 25}});

**8、查询age >= 23 并且 age <= 26 注意书写格式**

db.userInfo.find({age: {$gte: 23, $lte: 26}});

**9、查询name中包含 mongo的数据 模糊查询用于搜索**

db.userInfo.find({name: /mongo/});

//相当于%%

select \* from userInfo where name like ‘%mongo%’;

**10、查询name中以mongo开头的**

db.userInfo.find({name: /^mongo/});

select \* from userInfo where name like ‘mongo%’;

**11、查询指定列name、age数据**

db.userInfo.find({}, {name: 1, age: 1});

相当于：select name, age from userInfo;

当然name也可以用true或false,当用ture的情况下河name:1效果一样，如果用false就是排除name，显示name以外的列信息。

**12、查询指定列name、age数据, age > 25**

db.userInfo.find({age: {$gt: 25}}, {name: 1, age: 1});

相当于：select name, age from userInfo where age >25;

**13、按照年龄排序 1升序 -1降序**

升序：db.userInfo.find().sort({age: 1});

降序：db.userInfo.find().sort({age: -1});

**14、查询name = zhangsan, age = 22的数据**

db.userInfo.find({name: 'zhangsan', age: 22});

相当于：select \* from userInfo where name = ‘zhangsan’ and age = ‘22’;

**15、查询前5条数据**

db.userInfo.find().limit(5);

相当于：selecttop 5 \* from userInfo;

**16、查询10条以后的数据**

db.userInfo.find().skip(10);

相当于：select \* from userInfo where id not in (

selecttop 10 \* from userInfo

);

**17、查询在5-10之间的数据**

db.userInfo.find().limit(10).skip(5);

可用于分页，limit是pageSize，skip是第几页\*pageSize

**18、or与 查询**

db.userInfo.find({$or: [{age: 22}, {age: 25}]});

相当于：select \* from userInfo where age = 22 or age = 25;

**19、查询第一条数据**

db.userInfo.findOne();

相当于：selecttop 1 \* from userInfo;

db.userInfo.find().limit(1);

**20、查询某个结果集的记录条数 统计数量**

db.userInfo.find({age: {$gte: 25}}).count();

相当于：select count(\*) from userInfo where age >= 20;

如果要返回限制之后的记录数量，要使用count(true)或者count(非0)

db.users.find().skip(10).limit(5).count(true);

**21、按照某列进行排序**

db.userInfo.find({sex: {$exists: true}}).count();

相当于：select count(sex) from userInfo;

## 4.3 修改数据

修改里面还有查询条件。你要该谁，要告诉mongo。

查找名字叫做小明的，把年龄更改为16岁：

|  |
| --- |
| 1. db.student.update({"name":"小明"},{$set:{"age":16}}); |

查找数学成绩是70，把年龄更改为33岁：

|  |
| --- |
| 1. db.student.update({"score.shuxue":70},{$set:{"age":33}}); |

更改所有匹配项目："

By default, the update() method updates a single document. To update multiple documents, use the multi option in the update() method.

|  |
| --- |
| 1. db.student.update({"sex":"男"},{$set:{"age":33}},{multi: true}); |

完整替换，不出现$set关键字了： 注意

|  |
| --- |
| 1. db.student.update({"name":"小明"},{"name":"大明","age":16}); |

db.users.update({name: 'Lisi'}, {$inc: {age: 50}}, false, true);

相当于：update users set age = age + 50 where name = ‘Lisi’;

db.users.update({name: 'Lisi'}, {$inc: {age: 50}, $set: {name: 'hoho'}}, false, true);

相当于：update users set age = age + 50, name = ‘hoho’ where name = ‘Lisi’;

## 4.4 删除数据

|  |
| --- |
| 1. db.collectionsNames.remove( { "borough": "Manhattan" } ) 2. db.users.remove({age: 132}); |

By default, the remove() method removes all documents that match the remove condition. Use the justOne option to limit the remove operation to only one of the matching documents.

|  |
| --- |
| 1. db.restaurants.remove( { "borough": "Queens" }, **{ justOne: true }** ) |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |