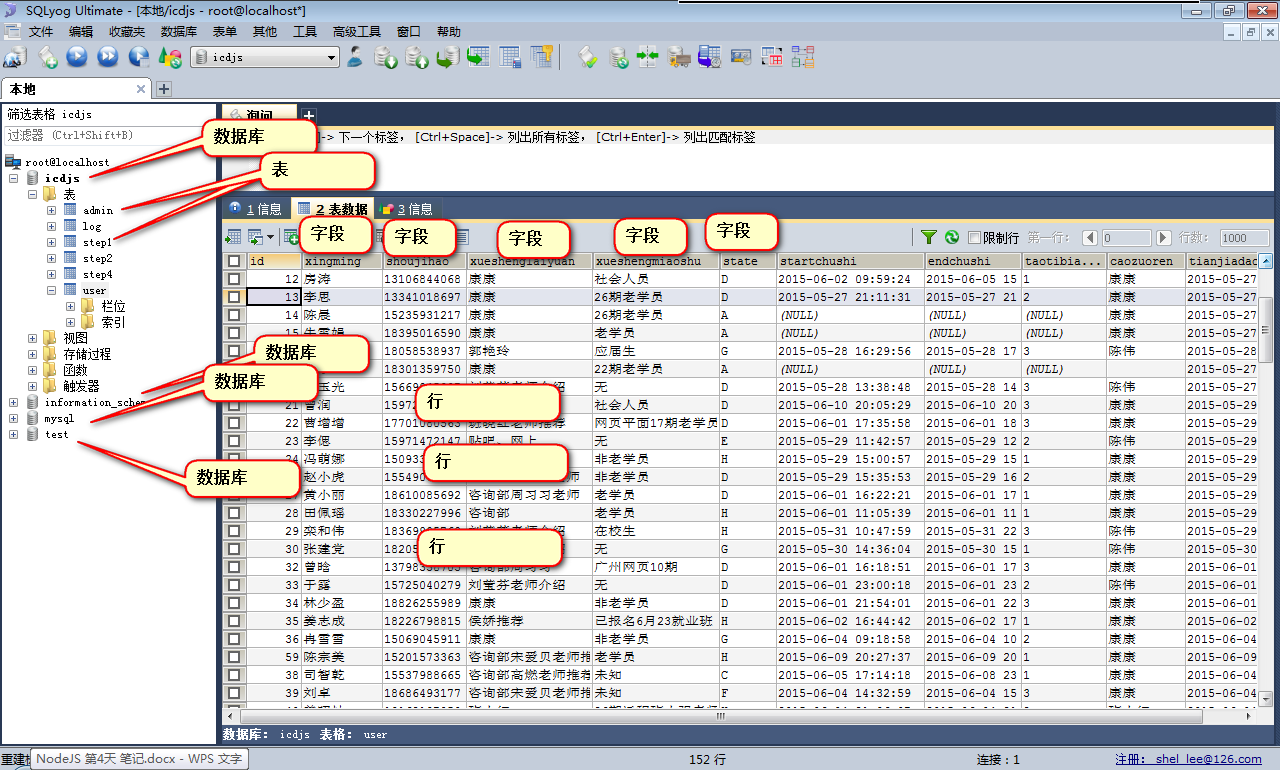
# 一、传统数据库技术回顾

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 xingbie = '女';

再比如，查询所有女生，并且年龄20~24之间，且在北京：

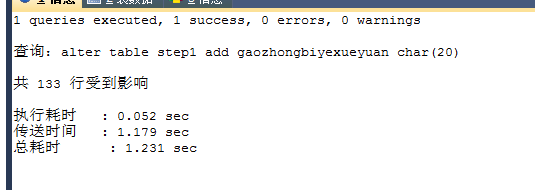
|  |
| --- |
| 1. SELECT \* FROM step1 WHERE xingbie = '女' AND nianling < 24 AND nianling >= 20 AND xianzaisuozaidi = '北京'; |

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。

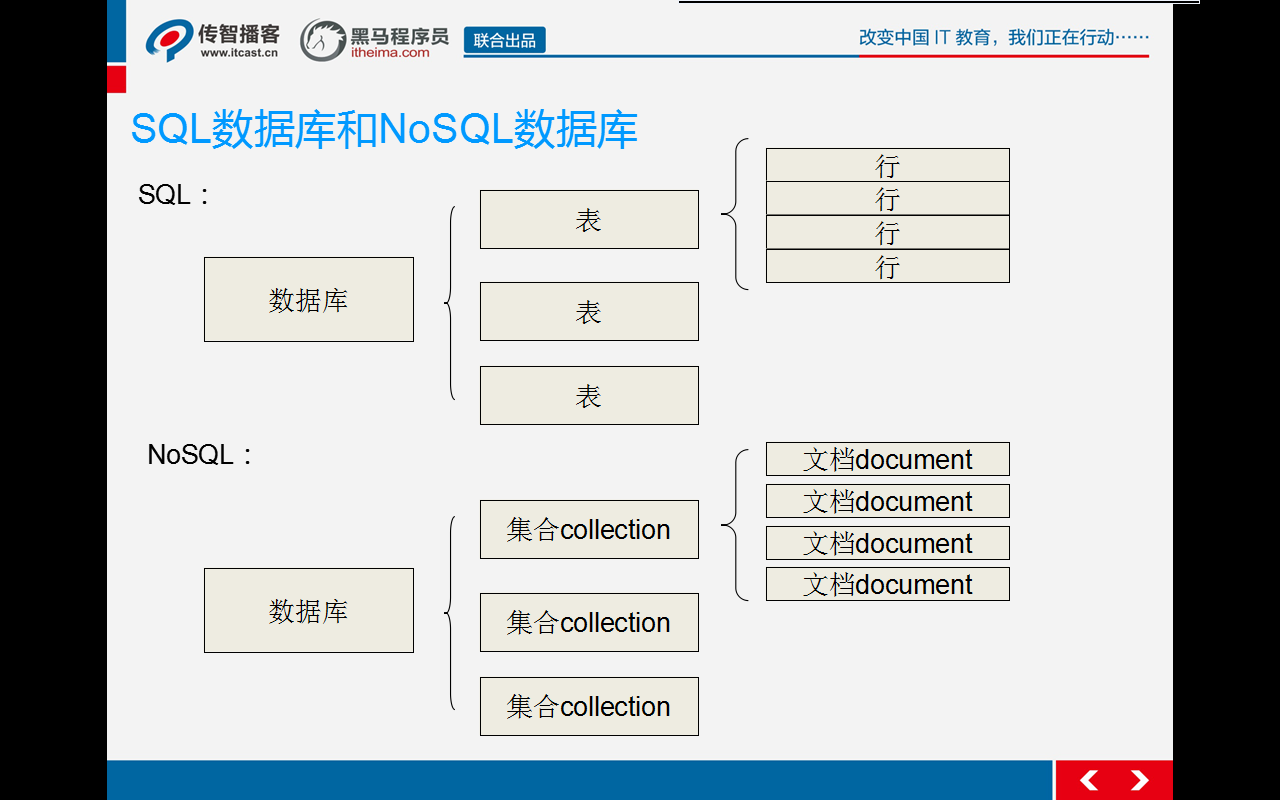
非结构型数据库NoSQL应运而生。

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

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。

我们看，有些系统，特别需要筛选。比如，筛选出所有女生大于20岁的。那么SQL型数据库，非常擅长！因为它有行、列的概念。

但是，有些系统，真的不需要进行那么多的筛选，比如站内信。站内信只需要存储就好了。不需要筛选。那么NoSQL的。

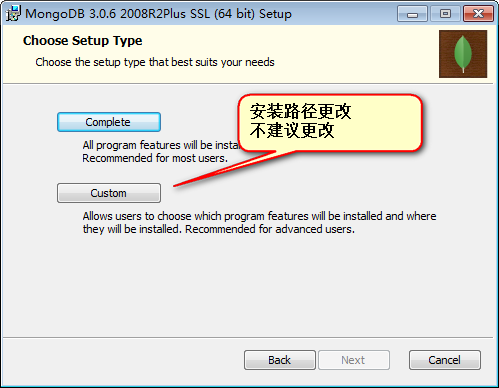
NoSQL不是银弹，没有资格挑战老牌数据库，还是特定情况下，是适合的。

# 三、MongoDB安装

官网：<https://www.mongodb.com/>

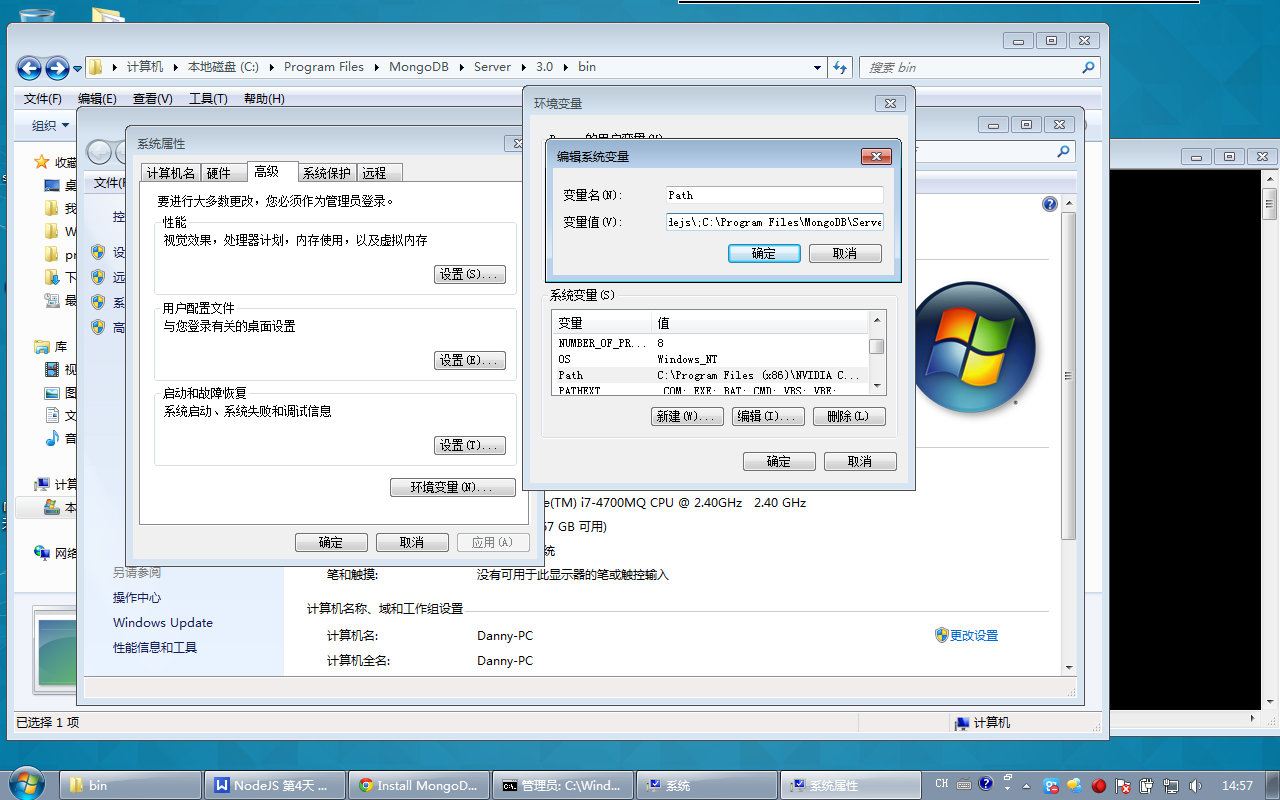
手册：<https://docs.mongodb.org/manual/>

win7系统需要安装补丁，KB2731284。



此时，我们看一下装好的文件夹：

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：

mongo 使用数据库

mongod 开机

mongoimport 导入数据

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：



student就是所谓的集合。集合中存储着很多json。

student是第一次使用，集合将自动创建。

# 四、数据库使用

要管理数据库，必须先开机，开机使用**mongod --dbpath c:\mongo**

管理数据库：mongo (一定要在新的cmd中输入)



清屏：

|  |
| --- |
| 1. cls |

查看所有数据库列表

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

使用数据库、创建数据库

|  |
| --- |
| 1. use itcast |

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：

|  |
| --- |
| 1. 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。

删除数据库，删除当前所在的数据库

|  |
| --- |
| 1. db.dropDatabase(); |

## 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.student.insert({"name":"xiaoming"});** |

我们不可能一条一条的insert。所以，我们希望用sublime在外部写好数据库的形式，然后导入数据库：

|  |
| --- |
| 1. mongoimport **--db test** **--collection restaurants** **--drop** **--file primer-dataset.json** |

**-db test 想往哪个数据库里面导入**

**--collection restaurants 想往哪个集合中导入**

**--drop 把集合清空**

**--file primer-dataset.json 哪个文件**

**这样，我们就能用sublime创建一个json文件，然后用mongoimport命令导入，这样学习数据库非常方便。**

## 4.2 查找数据

查找数据，用find。find中没有参数，那么将列出这个集合的所有文档：

|  |
| --- |
| 1. db.restaurants.find() |

精确匹配：

|  |
| --- |
| 1. db.student.find({"score**.**shuxue":70}); |

多个条件：

|  |
| --- |
| 1. db.student.find({"score.shuxue":70 , "age":12}) |

大于条件：

|  |
| --- |
| 1. db.student.find({"score.yuwen":**{$gt:50}**}); |

或者。寻找所有年龄是9岁，或者11岁的学生

|  |
| --- |
| 1. db.student.find({**$or**:[{"age":9},{"age":11}]}); |

查找完毕之后，打点调用sort，表示升降排序。

|  |
| --- |
| 1. db.restaurants.find().**sort(** { "borough": 1, "address.zipcode": 1 } **)** |

## 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：

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

## 4.4 删除数据

|  |
| --- |
| 1. db.restaurants.remove( { "borough": "Manhattan" } ) |

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 }** ) |

# 五、Node.js操作MongoDB

需要引包：

|  |
| --- |
| 1. npm install mongodb |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |