基于MongoDB的非关系型数据库实验

非关系型数据库 简介

非关系型数据库也叫**NoSQL数据库**，全称是“**Not Only SQL**”。
非关系型数据库提出另一种理念，例如，**以键值对存储**，且**结构不固定**，每一个元组可以有不一样的字段，每个元组可以根据需要增加一些自己的键值对，这样就不会局限于固定的结构，可以减少一些时间和空间的开销。使用这种方式，用户可以根据需要去添加自己需要的字段，这样，为了获取用户的不同信息，不需要像关系型数据库中，要对多表进行关联查询。仅需要**根据id取出相应的value**就可以完成查询。

关系型数据库与非关系型数据库的区别：

关系型数据库通过外键关联来建立表与表之间的关系，非关系型数据库通常指数据以对象的形式存储在数据库中，而**对象之间的关系通过每个对象自身的属性来决定。**

NoSQL数据库的特点：

- **模式自由**
  - 不需要定义表结构，数据表中的每条记录都可能有不同的属性和格式。
- **逆规范化**
  - 不遵循范式要求，去掉完整性约束，减少表之间的依赖
- **弹性可扩展**
  - 可在系统运行的过程中，动态的删除和增加节点。
- **多副本异步复制**
  - 数据快速写入一个节点，其余节点通过读取写入的日志来实现异步复制。
- **弱事务**
  - 不能完全满足事务的ACID特性，但是可以保证事务的最终一致性。

什么时候用NoSQL数据库：

- 数据库表schema经常变化
- 数据库表字段是复杂数据类型
- 高并发数据库请求
- 海量数据的分布式存储

常见的关系型数据库有哪些：

- MySQL
- Oracle
- DB2

- Microsoft SQL Server

  ……

关系型数据库

关系型数据库的特性

1. 关系型数据库，是指采用了关系模型来组织数据的数据库；
2. 关系型数据库的最大特点就是事务的一致性；
3. 简单来说，关系模型指的就是二维表格模型，而一个关系型数据库就是由二维表及其之间的联系所组成的一个数据组织。

关系型数据库的优点

1. 容易理解：二维表结构是非常贴近逻辑世界一个概念，关系模型相对网状、层次等其他模型来说更容易理解；
2. 使用方便：通用的SQL语言使得操作关系型数据库非常方便；
3. 易于维护：丰富的完整性(实体完整性、参照完整性和用户定义的完整性)大大减低了数据冗余和数据不一致的概率；
4. 支持SQL，可用于复杂的查询。

关系型数据库的缺点

1. 为了维护一致性所付出的巨大代价就是其读写性能比较差；
2. 固定的表结构；
3. 高并发读写需求；
4. 海量数据的高效率读写；

常见的NoSQL数据库有哪些：

- MongoDB
- NoSql
- Redis
- Memcached
- HBase

  ……

非关系型数据库

非关系型数据库的特性

1. 使用键值对存储数据；
2. 分布式；

3. 一般不支持ACID特性；
4. 非关系型数据库严格上不是一种数据库，应该是一种数据结构化存储方法的集合。

非关系型数据库的优点

1. 无需经过SQL层的解析，读写性能很高；
2. 存储数据的格式：NoSQL的存储格式是key,value形式、文档形式、图片形式等等，文档形式、图片形式等等，而关系型数据库则只支持基础类型。

非关系型数据库的缺点

1. 不提供sql支持，学习和使用成本较高；
2. 无事务处理，附加功能bi和报表等支持也不好；

MongoDB

MongoDB简介

MongoDB.inc 公司研发的一款免费开源的跨平台 NoSQL 数据库，命名源于英文单词 humongous，意思是「巨大无比」，可见开发组对 MongoDB 的定位。与关系型数据库不同，MongoDB 的数据以类似于 JSON 格式的二进制文档存储：

```
{
    name: "Zhangsan",
    age: 18,
    hobbies: ["Steam", "Guitar"]
}
```

- 功能强大、使用灵活、性能卓越且易于扩展的数据库。

文档型的数据存储方式有几个重要好处：

- 文档的数据类型可以对应到语言的数据类型，如数组类型（Array）和对象类型（Object）；
- 文档可以嵌套，有时关系型数据库涉及几个表的操作，在 MongoDB 中一次就能完成，可以减少昂贵的连接花销；
- 文档不对数据结构加以限制，不同的数据结构可以存储在同一张表。

MongoDB 的文档数据模型和索引系统能有效提升数据库性能；复制集功能提供数据冗余，自动化容灾容错，提升数据库可用性；分片技术能够分散单服务器的读写压力，提高并发能力，提升数据库的可拓展性。MongoDB 高性能，高可用性、可扩展性等特点，使其至 2009 年发布以来，逐渐被认可，并被越来越多的用于生产环境中。AWS、GCP、阿里云等云平台都提供了十分便捷的 MongoDB 云服务。

官方网站：https://www.mongodb.org
开源项目：http://github.com/mongodb

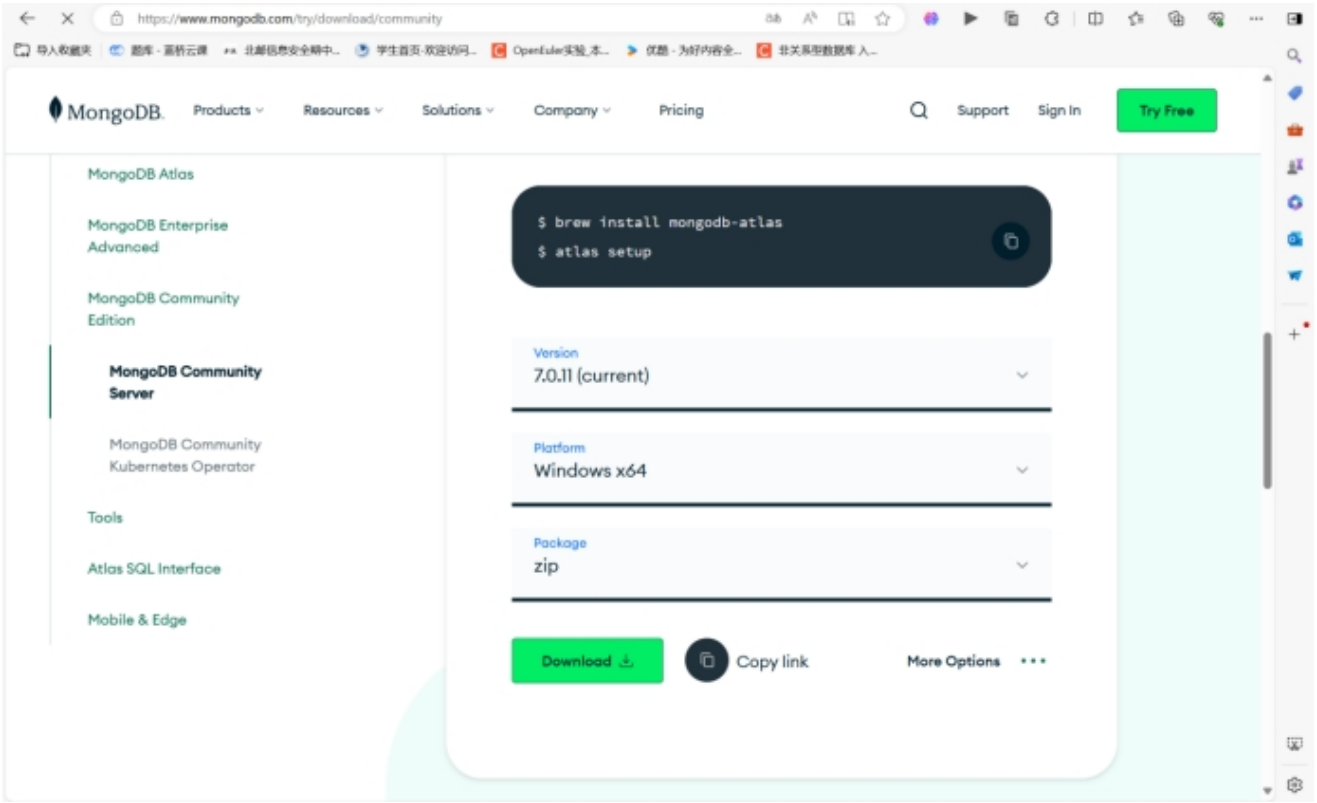
MongoDB特点

- 面向集合存储，易存储对象类型的数据
- 模式自由
- 支持动态查询
- 支持完全索引，包含内部对象
- 支持查询
- 支持复制和故障恢复
- 使用高效的二进制数据存储，包括大型对象（如视频等）
- 自动处理碎片，以支持云计算层次的扩展性
- 支持RUBY，PYTHON，JAVA，C++，PHP等多种语言
- 文件存储格式为BSON（一种JSON的扩展）
- 可通过网络访问

MongoDB安装

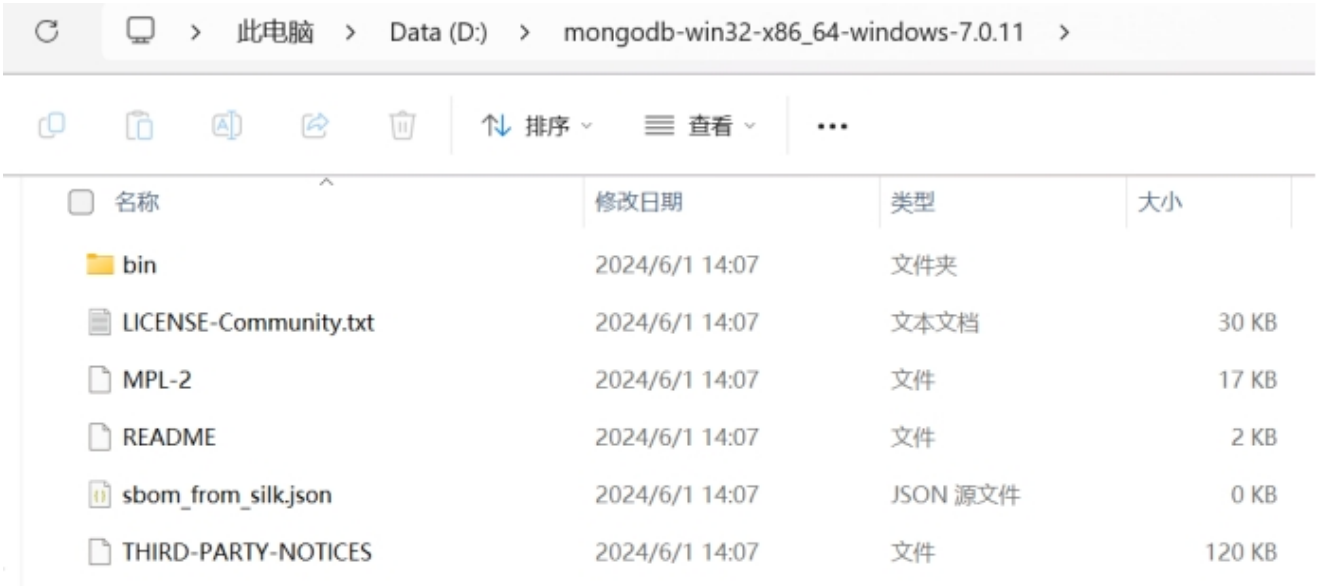
1、下载压缩包

下载地址：https://www.mongodb.com/try/download/community
这里以zip的格式进行下载：

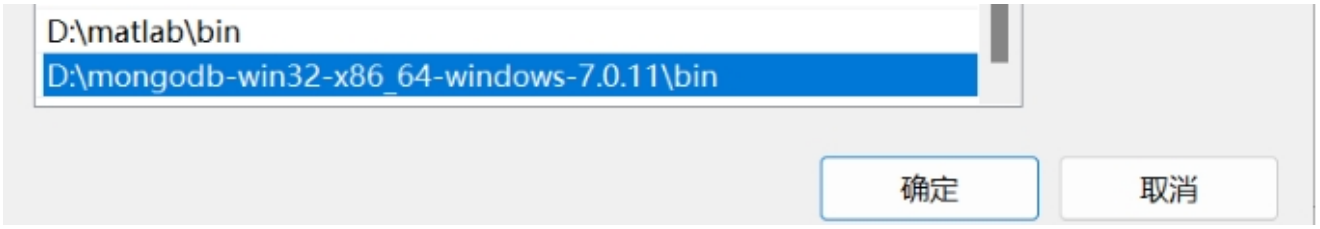


2、解压

下载完成后得到压缩包，解压；其中的bin目录就存放着mongodb相关的命令。



3、编辑环境变量

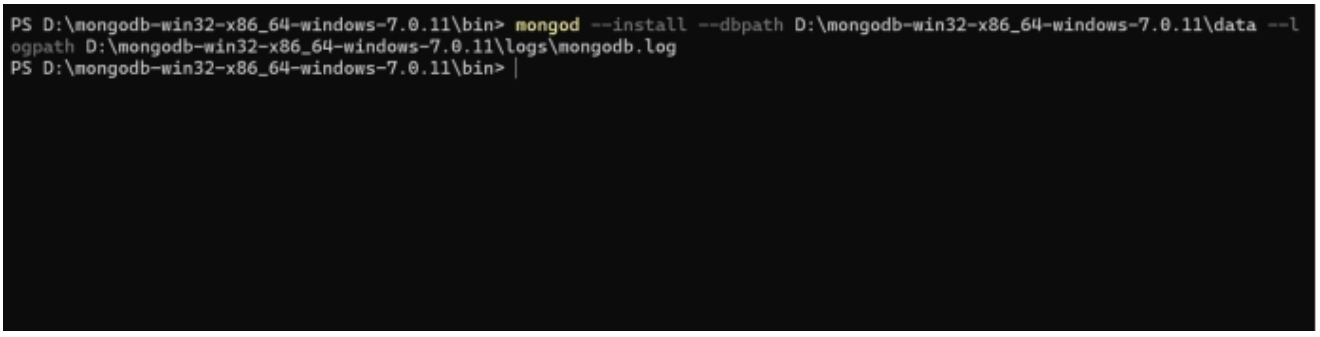


4、安装服务

首先要在安装目录里创建两个目录：

- 数据目录：data
- 日志目录：logs
  然后以管理员模式，切换到安装目录下的bin目录运行以下格式命令来指定mongdb的数据及日志目录。

```
mongod --install --dbpath D:\mongodb-win32-x86_64-windows-7.0.11\data --
logpath D:\mongodb-win32-x86_64-windows-7.0.11\logs\mongodb.log
```

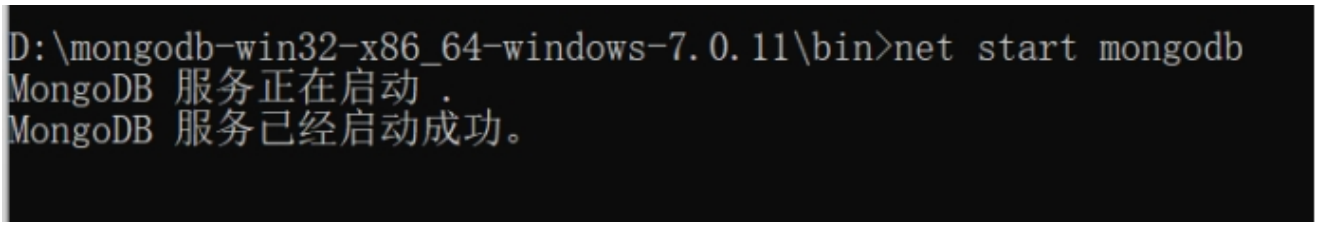


没有任何报错和提示，则代表MongoDB服务创建成功。

5、启动服务

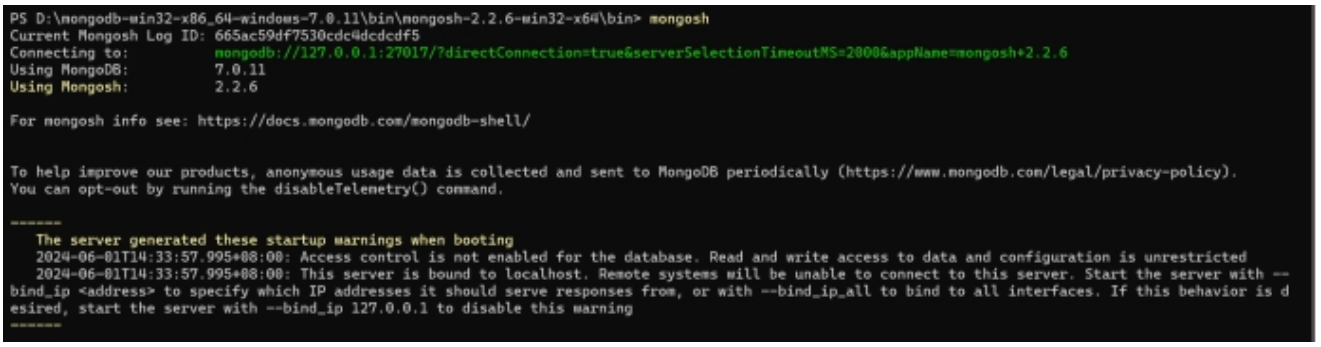
输入以下命令启动服务：

```
net start mongodb
```



6、登录
输入：

```
mongosh
```



MongoDB的CRUD 操作

概念:
集合==表, 文档==数据

命令关键词:

- `Show dbs`： 查看数据库
- `Show collectios`： 查看集合
- `Create collection`： 创建一个集合
- `use`： 切换数据库
- `insert`： 插入数据
- `Find`： 查找数据
- `Update`： 修改数据
- `Remove`： 删除数据

在进行增查改删操作之前，先了解下**常用的 shell 操作**:

- `db`： 显示当前所在数据库，默认为 `test`
- `show dbs`： 列出可用数据库
- `show tables show collections`： 列出数据库中可用集合
- `use <database>`： 用于切换数据库

MongoDB 预设有两个数据库，**admin** 和 **local**，admin 用来存放系统数据，local 用来存放该实例数据，在副本集中，一个实例的 local 数据库对于其它实例是不可见的。使用 `use` 命令切换数据库：

```
> use admin
> use local
> use newDatabase
```

可以 use 一个不存在的数据库，当你存入新数据时，mongoDB 会创建这个数据库：

```
> use newDatabase
> db.newCollection.insert({x:1})
WriteResult({ "nInserted" : 1 })
```

以上命令向数据库中插入一个文档，返回 1 表示插入成功，mongoDB 自动创建 newCollection 集合和数据库 newDatabase。下面将开始进行正式的**增删改查操作**。

MongoDB创建和删除数据库

MongoDB中使用**use关键字**来创建一个数据库：

```
use chenshifeng;#创建了一个数据库
db;#查看当前的数据库
db.dropDatabase();
```

增：插入文档

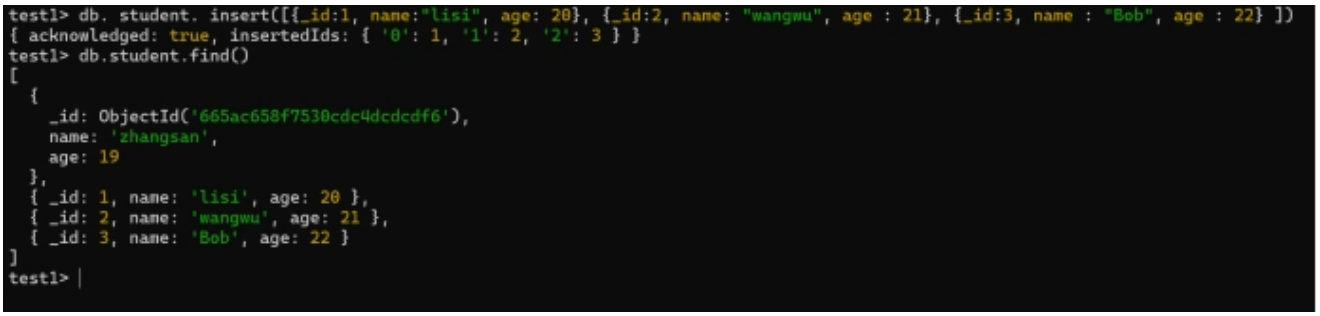
1. **插入一条数据**

```
test1> db.student.insert({name:"zhangsan",age:19})
test1> db.student.find()
```

```
test> use test1
switched to db test1
test1> db.student.insert({name:"zhangsan",age:19})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('665ac658f7530cdc4dcdcdf6') }
}
test1> db.student.find()
[
  {
    _id: ObjectId('665ac658f7530cdc4dcdcdf6'),
    name: 'zhangsan',
    age: 19
  }
]
test1>
```

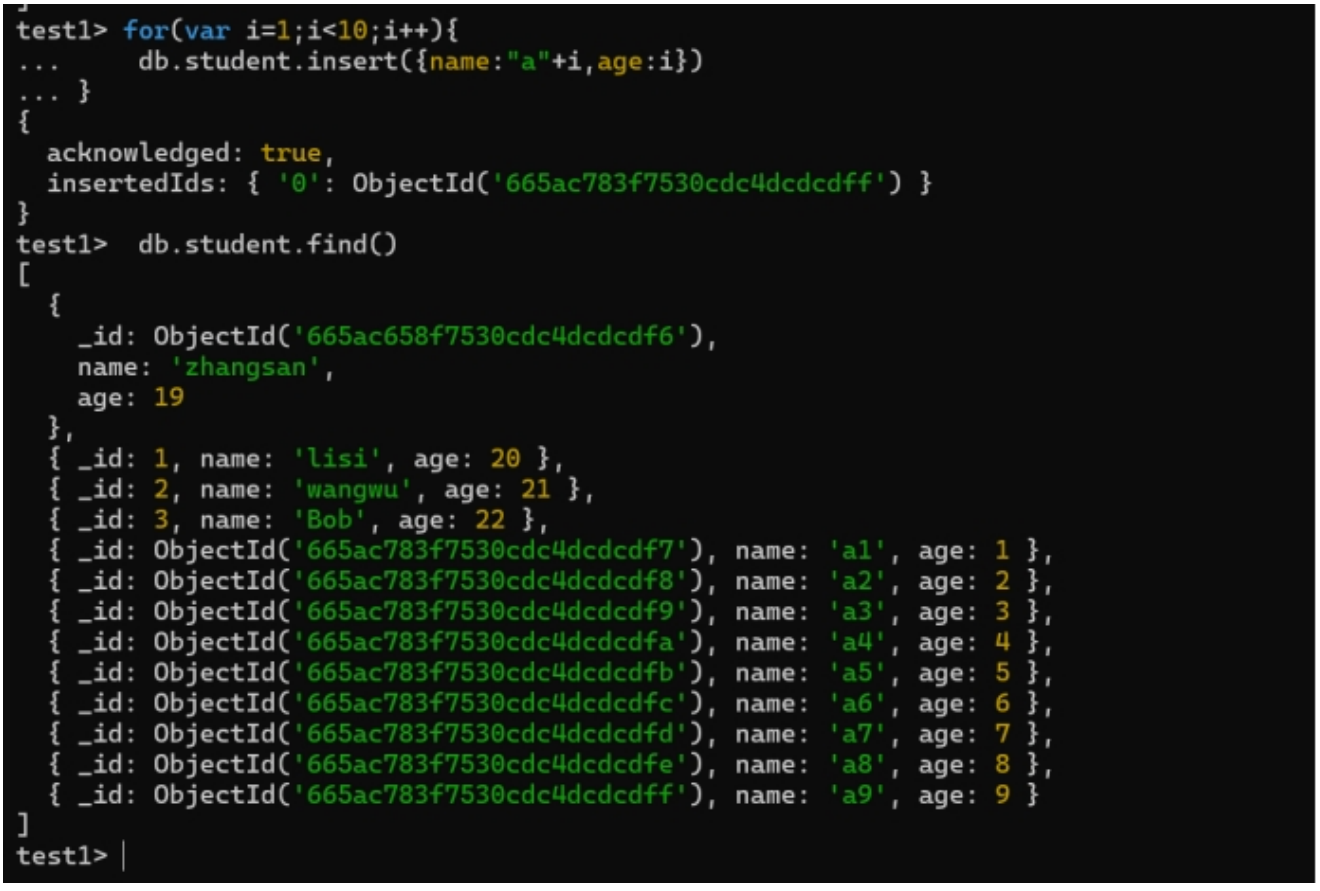
2. 一次插入多条数据并指定 _id

```
test1> db.student.insert([{_id:1, name:"lisi", age: 20}, {_id:2, name:
"wangwu", age : 21}, {_id:3, name : "Bob", age : 22} ])
test1> db.student.find()
```



3. 利用for循环插入数据

```
test1> for(var i=1;i<10;i++){
...      db.student.insert({name:"a"+i,age:i})
... }
test1>  db.student.find()
```



删：删除文档

1. **删除name为zhangsan的数据**

```
test1> db.student.remove({name:"zhangsan"},true)
test1> db.student.find()
```

```
test1> db.student.remove({name:"zhangsan"},true)
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
test1> db.student.find()
[
  { _id: 1, name: 'lisi', age: 20 },
  { _id: 2, name: 'wangwu', age: 21 },
  { _id: 3, name: 'Bob', age: 22 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf7'), name: 'a1', age: 1 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf8'), name: 'a2', age: 2 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf9'), name: 'a3', age: 3 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfa'), name: 'a4', age: 4 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfb'), name: 'a5', age: 5 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfc'), name: 'a6', age: 6 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfd'), name: 'a7', age: 7 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfe'), name: 'a8', age: 8 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdff'), name: 'a9', age: 9 }
]
test1>
```

改：修改文档

1. **给name为Bob的年龄增加2岁**

```
test1> db.student.update({name:"Bob"},{$inc:{age:2}})
test1> db.student.find()
```

```
test1> db.student.update({name:"Bob"},{$inc:{age:2}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test1> db.student.find()
[
  { _id: 1, name: 'lisi', age: 20 },
  { _id: 2, name: 'wangwu', age: 21 },
  { _id: 3, name: 'Bob', age: 24 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf7'), name: 'a1', age: 1 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf8'), name: 'a2', age: 2 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf9'), name: 'a3', age: 3 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfa'), name: 'a4', age: 4 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfb'), name: 'a5', age: 5 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfc'), name: 'a6', age: 6 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfd'), name: 'a7', age: 7 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfe'), name: 'a8', age: 8 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdff'), name: 'a9', age: 9 }
]
test1>
```

2. **将name为a3的修改为a33**

```
test1> db.student.update({name:"a3"},{$set:{name:"a33"}})
test1> db.student.find()
```

```
test1> db.student.update({name:"a3"},{$set:{name:"a33"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test1> db.student.find()
[
  { _id: 1, name: 'lisi', age: 20 },
  { _id: 2, name: 'wangwu', age: 21 },
  { _id: 3, name: 'Bob', age: 24 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf7'), name: 'a1', age: 1 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf8'), name: 'a2', age: 2 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf9'), name: 'a33', age: 3 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfa'), name: 'a4', age: 4 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfb'), name: 'a5', age: 5 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfc'), name: 'a6', age: 6 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfd'), name: 'a7', age: 7 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfe'), name: 'a8', age: 8 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdff'), name: 'a9', age: 9 }
]
test1>
```

查：查询文档

1. 查询指定列的所有数据

(1) 只查询name列：

```
test1>  db.student.find({},{name:1})
```

```
test1> db.student.find({},{name:1})
[
  { _id: 1, name: 'lisi' },
  { _id: 2, name: 'wangwu' },
  { _id: 3, name: 'Bob' },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf7'), name: 'a1' },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf8'), name: 'a2' },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf9'), name: 'a33' },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfa'), name: 'a4' },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfb'), name: 'a5' },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfc'), name: 'a6' },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfd'), name: 'a7' },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfe'), name: 'a8' },
  { _id: ObjectId('665ac783f7530cdc4dcdcdff'), name: 'a9' }
]
```

(2) 查询除name以外的列：

```
test1> db.student.find({},{name:0})
```

```
test1>  db.student.find({},{name:0})
[
  { _id: 1, age: 20 },
  { _id: 2, age: 21 },
  { _id: 3, age: 24 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf7'), age: 1 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf8'), age: 2 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdf9'), age: 3 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfa'), age: 4 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfb'), age: 5 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfc'), age: 6 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfd'), age: 7 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfe'), age: 8 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdff'), age: 9 }
]
test1>
```

2. 查询指定条件的数据

(1)查询age>21的数据：

```
test1> db.student.find({age:{$gt:21}})
```

```
test1> db.student.find({age:{$gt:21}})
[ { _id: 3, name: 'Bob', age: 24 } ]
```

(2)查询age为7/8/9的数据：

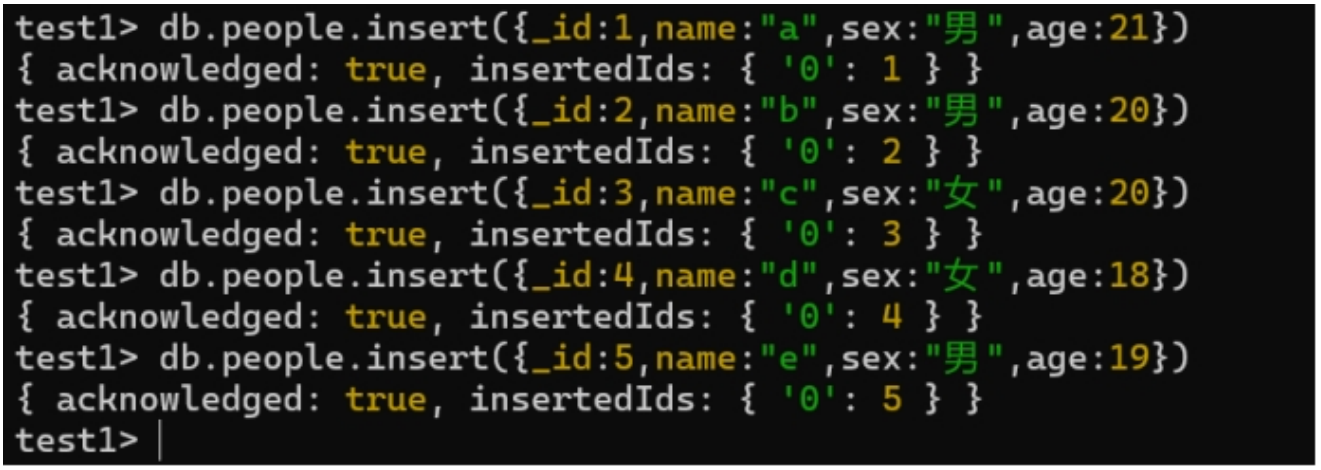
```
test1> db.student.find({age:{$in:[7,8,9]}})
```

```
test1> db.student.find({age:{$in:[7,8,9]}})
[
  { _id: ObjectId('665ac783f7530cdc4dcdcdfd'), name: 'a7', age: 7 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdfe'), name: 'a8', age: 8 },
  { _id: ObjectId('665ac783f7530cdc4dcdcdff'), name: 'a9', age: 9 }
]
test1>
```

3. 聚合查询

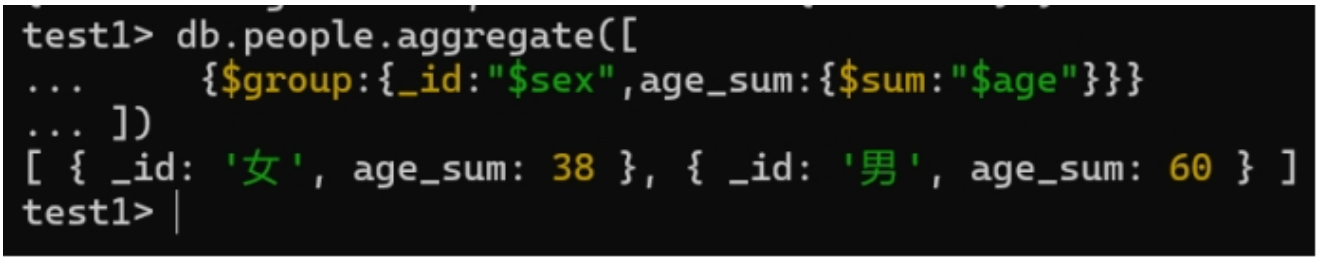
数据准备：

```
test1> db.people.insert({_id:1,name:"a",sex:"男",age:21})
test1> db.people.insert({_id:2,name:"b",sex:"男",age:20})
test1> db.people.insert({_id:3,name:"c",sex:"女",age:20})
test1> db.people.insert({_id:4,name:"d",sex:"女",age:18})
test1> db.people.insert({_id:5,name:"e",sex:"男",age:19})
```



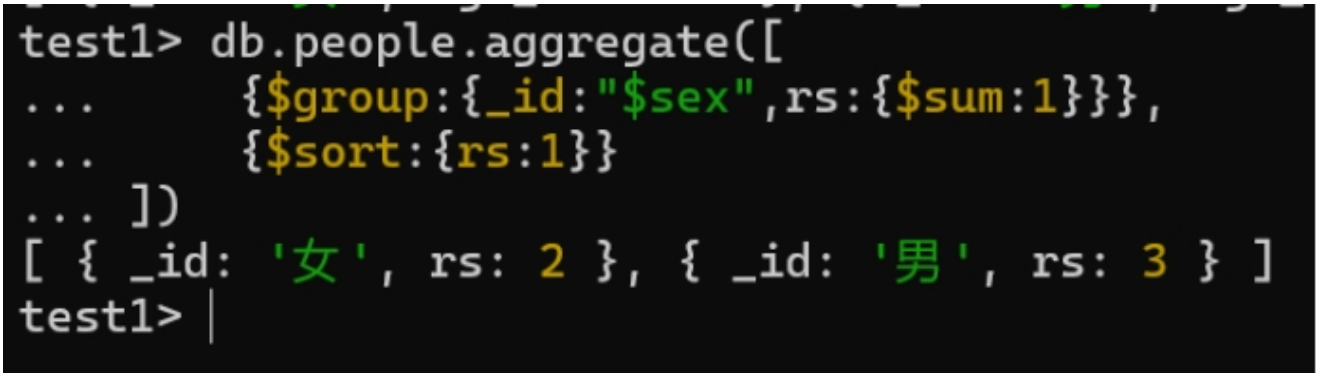
1. 统计男生、女生的总年龄：

```
test1> db.people.aggregate([
{$group:{_id:"$sex",age_sum:{$sum:"$age"}}}
])
```



2. 查询男生、女生人数，按人数升序：

```
db.people.aggregate([
{$group:{_id:"$sex",rs:{$sum:1}}},
{$sort:{rs:1}}
])
```



开启profile

MongoDB的profile就和mysql的慢查询类似，用于记录执行时间超过多少的语句。

```
db.getProfilingLevel()#获取profile级别
db.setProfilingLevel(1,2000)# 设置profile级别
```

profile级别有三种：

1. 不开启
2. 记录慢命令，默认为大于100ms
3. 记录所有命令

查询profile记录：

- `db.system.profile.find();`
- `ts`： 该命令在何时执行
- `op`：操作类型
- `query`：本命令的详细信息
- `responseLength`：返回结果集的大小
- `ntoreturn`：本次查询实际返回的结果集
- `millis`：该命令执行耗时，以毫秒记

创建索引

```
db.collections.ensureIndex({xx:1})#创建单列索引
db.collections.ensureIndex({xx:1,xx:1})#创建多列索引
db.collections.ensureIndex({xx:1},{“unique”:true})#创建唯一索引
```

查看、删除索引

```
db.system.indexes.find();#查看索引
db.collections.getIndexes();#查看当前集合中的索引
#Mongodb中使用dropIdenx来删除索引：
```

```
db.collections.dropIndex({xx:1});#删除指定索引
db.user.dropIndexes();#删除所有的索引
```

explain

使用explain可以解析查询语句

```
db.collection.find({xx:xx}).explain();
```

Explain说明：

- `cursor`：返回游标类型(BasicCursor 或 BtreeCursor)
- `nscanned`：被扫描的文档数量
- `n`：返回的文档数量
- `millis`：耗时(毫秒)
- `indexBounds`：所使用的索引
- `isMultiKey`:是否使用了多键索引
- `scanAndOrder`:是否在内存中对结果集进行了排序
- `indexOnly`:是否只使用索引就能完成查询（覆盖索引）

相比传统关系型数据库，MongoDB 的 CURD 操作更像是编写程序，更符合开发人员的直觉，不过 MongoDB 同样也支持 SQL 语言。MongoDB 的 CURD 引擎配合索引技术、数据聚合技术和 JavaScript 引擎，赋予 MongoDB 用户更强大的操纵数据的能力。

关闭实例

关闭 mongoDB 服务：

```
> use admin
> db.shutdownServer()
```

使用 exit 或 Ctrl + C 断开连接:

```
> exit
```

总结

上述，我们启动了一个不安全的 MongoDB 实例，应用到生产环境中，还需要了解 MongoDB 的安全机制，了解如何建立索引提升数据库的读写速度。随着数据的增长，需要了解副本集技术如何将多个实例部署到不同的服务器、了解分片技术对数据库进行横向扩展。为保证服务器性能和安全，需要了解如何对 MongoDB 进行测试和监控…

参考：

https://blog.csdn.net/u012431703/article/details/94393585

https://blog.csdn.net/qq_45173404/article/details/114260970