**一、基本使用**

（1）使用数据库

```
use mydb;
```

（2）创建集合（相当于关系数据中的表）

```
db.createCollection(" dept")
```

（3）插入数据

```
db.dept.insert({"deptno":10,"dname":"财务部","loc":"北京"})
```

（4）查看数据

```
db.dept.find()
```

（6）删除数据

```
db.dept.remove({"_id" : ObjectId("5e58e5f97831d42f91ff903d")})
```

（7）更新数据

```
db.dept.update({"_id" : ObjectId("5e58e7527831d42f91ff903f")},deptData)
```

（8）删除集合

```
db.dept.drop();
```

（9）删除数据库

```
db.dropDatabase();
```

**二、读写操作（1）**

**数据增加**

（1）在 infos 集合中增加一条 url 为 www.ljc.cn 的数据

```
db.infos.insert({"url":"www.ljc.cn"})
```

（2）在 infos 集合中增加数组数据，数组有两个 url 分别为 www.ljc.cn 和 www.ljclg.cn 的数据

```
db.infos.insert([{"url":"www.ljc.cn"},{"url":"www.ljclg.cn"}])
```

（3）在 infos 集合中增加 10000 个 url 分别为 ljc-1、ljc-2、ljc-3……的数据

```
for( var x=0;x<10000;x++ ){db.infos.insert({"url":"ljc-"+x})}
```

**数据查询**

（1）查出集合 infos 中的所有数据

```
db.infos.find()
```

（2）查出 url 为 www.ljc.cn 的数据

```
db.infos.find({"url":"www.ljc.cn"});
```

（3）查出 url 为 www.ljc.cn 的数据，但不显示“_id”

```
db.infos.find({"url":" www.ljc.cn"},{"_id":0});
```

**运算符： 大于($gt)、小于($lt)、大于等于($gte)、小于等于($lte)、不等于($ne)、等于（key : value 的形式）**

（4）查询姓名是张三的学生信息

```
db.students.find({"name":"张三"}).pretty()
```

（5）查询性别是男的学生信息

```
db.students.find({"sex":"男"}).pretty()
```

（6）查询年龄大于 19 岁的学生

```
db.students.find({"age":{"$gt":19}}).pretty()
```

（7）查询成绩大于 60 分的学生
db.students.find({"score":{"$gt":60}}).pretty()
（8）查询姓名不是王五的
db.students.find({"name":{"$ne":"王五"}}).pretty()
**运算符：与($and)、或（$or）、非($not、$nor)**
（9）查询年龄在 19-20 岁的学生信息
db.students.find({"age":{"$gte":19,"$lte":20}}).pretty()
（10）查询年龄大于 19 岁，或者成绩大于 90 分的学生信息
db.students.find({"$or": [{"age":{"$gt":19}},{"score":{"$gt":90}}]}).pretty()
（11）查询年龄小于等于 19 岁，并且成绩小于等于 90 分的学生信息
db.students.find({"$and":[{"age":{"$lte":19}},{"score":{"$gte":90}}]}).pretty()
**运算符：$in 和 $nin**
（12）查询姓名是“张三”、“李四”、“王五”的信息
db.student.find({"name":{"$in":["张三","李四","王五"]}}).pretty();
（13）查询姓名不是“张三”、“李四”、“王五”的信息
db.student.find({"name":{"$nin":["张三","李四","王五"]}}).pretty();

**三、读写操作（2）**
（1）查询姓名带谷字的同学信息
db.students.find({"name":/谷/}).pretty()
或者 db.students.find({"name":{"$regex":/谷/,"$options":"s"}}).pretty()
（2）查询姓名含有字符 A 或者 a 的同学信息
db.students.find({"name":/a/i}).pretty()
**升序（1），降序（-1）**
（3）按照成绩进行升序排序
db.students.find().sort({"score": 1}).pretty()
分页显示：skip(n) :表示跨过多少数据行，limit(n) :取出的数据行的个数限制
（4）分页显示（第一页 5 条数据)
db.students.find().sort({"score": -1}).skip(0).limit(5).pretty()
（5）分页显示（第二页 5 条数据)
db.students.find().sort({"score": -1}).skip(5).limit(5).pretty()
**数组的查询选择器运算符：$all、$size、$slice、$elemMatch**
（6）查询同时参加语文和数学课程的学生
db.students.find({"course":{"$all":["语文","数学"]}}).pretty()
（7）查询数组中第二个内容为数学的信息
db.students.find({"course.1":"数学"}).pretty()
**数组可以使用 key.index 的方式定义索引，默认索引下标从 0 开始**
（8）查询只参加两门课的学生
db.students.find({"course":{"$size":2}}).pretty()
（9）返回年龄为 19 岁所有学生的信息，但是要求只显示两门参加课程
显示前两门信息 db.students.find({"age":19},{"course":{"$slice":2}}).pretty()
显示后两门信息 db.students.find({"age":19},{"course":{"$slice":-2}}).pretty()
显示中间的部分 db.students.find({"age":19},{"course":{"$slice":[1,2]}}).pretty()

**第一个数据表示跳过的数量，第二个数据表示返回的数量**

（10）查询出年龄大于等于 19 岁，父母有人是局长的学生信息

db.students.find({"$and":[{"age":{"$gte":19}},{"parents":{"$elemMatch":{"job":"局长"}}}]}).pretty()

**运算符：$exists　判断某个字段是否存在，如果设置为 true 表示存在**

（11）查询具有 parents 成员的数据

db.students.find({"parents":{"$exists":true}}).pretty()

（12）查询不具有 course 成员的数据

db.students.find({"course":{"$exists":false}}).pretty()

## 四、修改操作

### 数据更新(update)

运算符：$set、$inc、$unset

（1）更新存在的数据：将年龄是 19 岁的人成绩都更新为 100 分

db.students.updateMany({"age":19},{"$set":{"score":100}})

（2）更新不存在的数据：将年龄是 30 岁的人姓名更新为“不存在”

**db.students.update({"age":30},{"$set":{"name":"不存在"}},true)**

**加减乘除：$add（+），$subtract（-），$multiply（*），$divide 操作符（/）**

**$inc: 操作数字字段的数据内容**

（3）将所有年龄为 19 岁的成绩一律减少 30 分，年龄减小 1 岁

db.students.updateMany({"age":19},{"$inc":{"score":-30,"age":-1}})

（4）将所有年龄是 20 岁的人的成绩修改为 99

db.students.updateMany({"age":20},{"$set":{"score":99}})

**$unset: 删除某个属性及其内容**

（5）删除“张三”的年龄与成绩信息

db.students.updateMany({"name":"张三"},{"$unset": {"age": 1,"score": 1}})

**运算符：$push、$pushAll、$addToSet**

（6）向“张三”添加课程信息“语文”

db.students.updateOne({"name": "张三"},{"$push": {"course": "语文"}})

（7）向“谷大神-E”里面的课程追加一个“美术”

db.students.updateOne({"name":"谷大神-E"},{"$addToSet": {"course":"美术"}})

（8）向“王五”的信息里面添加多个课程内容：美术、音乐、体育

db.students.updateOne({"name":"王五"},{"$push":{"course":{"$each":["美术","音乐","体育"]}}})

（9）向“王五”的信息添加新的内容：course 为舞蹈

db.students.updateOne({"name":"王五"},{"$addToSet": {"course":"舞蹈"}})

**运算符：　$pop、$pull、$pullAll**

**$pop: 删除数组内的数据：**

**{$pop: {field: value}},value 为-1 表示删除第一个，value 为 1 表示删除最后一个**

（10）删除“王五”的第一个课程
db.students.updateOne({"name":"王五"},{"$pop":{"course":-1}})
（11）删除“王五”的最后一个课程
db.students.updateOne({"name":"王五"},{"$pop":{"course":1}})
（12）删除“王五”的音乐课程信息
db.students.updateOne({"name": "王五"},{"$pull":{"course":"音乐"}})
（13）删除“谷大神-A”的三门课程
db.students.updateOne({"name":"谷大神-A"},{"$pullAll":{"course":["语文","数学","英语"]}})
运算符： $rename
（14）将“张三”name 成员名称修改为“姓名”
db.students.updateOne({"name":"张三"},{"$rename":{"name":"姓名"}})

**2 数据删除（remove）**

（1）删除所有姓名里面带“谷”的信息
db.students.remove({"name" : /谷/})
（2）删除姓名带“漓江”的信息，要求只删除一个
db.students.remove({"name" : /漓江/}, true)

**五、聚合,使用 emp**

（1）查询每个职位的人数
db.emp.aggregate([{"$group":{"_id":"$job","sum":{"$sum":1}}}])
（2）查询每个职位的总工资
db.emp.aggregate([{"$group":{"_id":"$job","总工资":{"$sum":"$salary"}}}])
（3）查询每个职位的总工资 ， 平均工资
db.emp.aggregate([{"$group":{"_id":"$job","sum":{"$sum":"$salary"},"avg":{"$avg":"$salary"}}}]
)
（4）查询每个职位的最高工资 ， 最低工资
db.emp.aggregate([{"$group":{"_id":"$job","max":{"$max":"$salary"},"min":{"$min":"$salary"}}}]
)
（5）查询出每个职位的工资数据
db.emp.aggregate([{"$group":{"_id":"$job",
"人数":{"$sum":1},"sum":{"$sum":"$salary"},"avg":{"$avg":"$salary"},
"max":{"$max":"$salary"},"min":{"$min":"$salary"}}}])
（6）查询每个职位的人员
db.emp.aggregate([{"$group":{"_id":"$job","name":{"$push":"$name"}}}])
（7）查询每个职位的人员 只保留第一个名称
db.emp.aggregate([{"$group":{"_id":"$job","first":{"$first":"$name"}}}])
（8）查询每个职位的人员 只保留最后一个名称
db.emp.aggregate([{"$group":{"_id":"$job","last":{"$last":"$name"}}}])
（9）只显示 name，job 成员，不显示“_id”

db.emp.aggregate([{"$project":{"_id":0,"name":1,"job":1}}])
（10）查询每个人员的年薪，显示 name，job 和年薪，不显示“_id”
**db.emp.aggregate([{"$project":{"_id":0,"name":1,"job":1,"年薪":{"$multiply":["$salary",12]}}}])**
（11）（未成功）找出工资大于 2000 的所有雇员姓名、年龄、工资
db.emp.aggregate([{"$match":{"salary":{"$gt":2000}}},{"$project":{"_id":0,"name":1,"age":1,"salary":1}}])

## 六、索引

（1）查看 students 集合的索引
db.students.getIndexes()
（2）在 age 成员上，设置一个降序索引
db.students.createIndex({"age":-1})
（3）查询 age 等于 19 岁的数据，并对查询过程在索引上做一个分析
db.students.find({"age":19}).explain()
（4）查询 score 大于 60 分的数据，并对查询过程在索引上做一个分析
db.students.find({"score":{"$gt":90}}).explain()
（5）查询 age 等于 19 岁 或者 score 大于 60 分的数据，并对查询过程在索引上做一个分析
db.students.find({"$or":[{"age":9}, {"score":{"$gt":60}}]}).explain()
（6）创建一个 age 和 score 的复合索引，两者均为降序
db.students.ensureIndex({"age":-1,"score":-1},{name:"age_-1_score_-1_index"})
（7）删除一个索引
db.students.dropIndex({"age" : -1})
（8）删除全部索引
db.students.dropIndexes()
（9）为 name 创建一个唯一索引
db.students.createIndex({"name":1},{"unique":true})
（10）在 phones 集合中，为 time 创建过期索引，升序，10 秒过期
db.phones.createIndex({"time":1,},{expireAfterSeconds:10})
（11）在 shop 集合中，为 loc 设置 2d 索引
db.shop.createIndex({"loc": "2d"})
（12）在 shop 集合中，查询坐标[11,11]附近最大距离为 5 范围内的数据
db.shop.find({"loc": {"$near" :[11,11], "$maxDistance": 5}})
（13）在 shop 集合中，查询坐标[9,9]和[11,11]范围内的数据
db.shop.find({"loc": {"$geoWithin" : {"$box": [[9,9],[11,11]]}}})
（14）在 shop 集合中，查询以坐标[10,10]为圆心，2 为半径范围内的数据
db.shop.find({"loc": {"$geoWithin" : {"$center": [[10,10],2]}}})

**mongodb 导入.csv 文件**
**bin 目录下执行：mongodbimport --type csv --headerline --file 文件路径（.csv）**