大纲：

1、MongoDb的体系结构 2、MongoDb安装配置与基础命令 3、MongoDB CRUD与全文索引

数据脚本.txt

一、MongoDb的体系结构

概要：

1. NoSql的概念
2. NoSql的应用场景
3. MongoDb的逻辑组成

1、NoSql的概念

> NoSQL(NoSQL = Not Only SQL)，意即“不仅仅是 [SQL](https://baike.baidu.com/item/SQL) ”，互联网的早期我们的数据大多以关系型数据库来存储的。其特点是规范的数据结构（预定义模式）、强一至性、表与表之间通过外键进行关联，这些特征使我们对数据的管理更加清晰和严谨，但随着互联网的发展数据成爆炸式的增长我们对数据库需要更好的灵活性和更快的速度。这就是NoSql可以做到的。它不需要预先定义模式，没有主外键关联、支持分片、支持复本。

NoSql的分类： 键值(Key-Value)存储数据库 这一类数据库主要会使用到一个哈希表，这个表中有一个特定的键和一个指针指向特定的数据。Key/value模型对于IT系统来说的优势在于简单、易部署。但是如果DBA只对部分值进行查询或更新的时候，Key/value就显得效率低下了。举例如：Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB. ** **列存储数据库。 这部分数据库通常是用来应对分布式存储的海量数据。键仍然存在，但是它们的特点是指向了多个列。这些列是由列家族来安排的。如：Cassandra, HBase, Riak. 文档型数据库 文档型数据库的灵感是来自于Lotus Notes办公软件的，而且它同第一种键值存储相类似。该类型的数据模型是版本化的文档，半结构化的文档以特定的格式存储，比如JSON。文档型数据库可 以看作是键值数据库的升级版，允许之间嵌套键值。而且文档型数据库比键值数据库的查询效率更高。如：CouchDB, MongoDb. 国内也有文档型数据库SequoiaDB，已经开源。 图形(Graph)数据库 图形结构的数据库同其他行列以及刚性结构的SQL数据库不同，它是使用灵活的图形模型，并且能够扩展到多个服务器上。NoSQL数据库没有标准的查询语言(SQL)，因此进行数据库查询需要制定数据模型。许多NoSQL数据库都有REST式的数据接口或者查询API。如：Neo4J, InfoGrid, Infinite Graph.

2、NoSql的应用场景

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；

- [ ] **基于豆瓣电影举例说明NoSQL的应用场景**
  - [ ] 电影基本信息分析
  - [ ] 电影与明星关系存储

3、MongoDb的逻辑组成

体系结构： 图片

逻辑结构与关系数据库的对比：

关系型数据库	MongoDb
database(数据库)	database（数据库）
table （表）	collection（ 集合）
row（ 行）	document（ BSON 文档）
column （列）	field （字段）
index（唯一索引、主键索引）	index （全文索引）
join （主外键关联）	embedded Document (嵌套文档)
primary key(指定1至N个列做主键)	primary key (指定_id field做为主键)
aggreation(groupy)	aggreation (pipeline mapReduce)

二、MongoDb安装配置与基础命令

概要：

1. mongoDb版本说明
2. mongoDb启动参数说明
3. 客户端Shell 的使用及参数说明
4. 数据库与集合的基础操作
5. mongoDb社区版说明

下载地址：https://www.mongodb.com/download-center/community


```
#下载
wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-4.0.5.tgz
# 解压
tar -zxvf mongodb-linux-x86_64-4.0.5.tgz
```

2.mongoDb启动参数说明

mongoDb 由C++编写，下载下来的包可以直接启动

```
#创建数据库目录
mkdir -p /data/mongo
# 启动mongo
./bin/mongod --dbpath=/data/mongo/
```

常规参数

参数	说明
dbpath	数据库目录，默认/data/db
bind_ip	监听IP地址，默认全部可以访问
port	监听的端口，默认27017
logpath	日志路径
logappend	是否追加日志
auth	是开启用户密码登陆
fork	是否已后台启动的方式登陆
config	指定配置文件

配置文件示例

```
vim mongo.conf
```

内容：

```

dbpath=/data/mongo/
port=27017
bind_ip=0.0.0.0
fork=true
logpath = /data/mongo/mongodb.log
logappend = true
auth=false
```

已配置文件方式启动

```
./bin/mongod -f mongo.conf
```

3.客户端Shell 的使用及参数说明

```
#启动客户端 连接 本机的地的默认端口
./bin/mongo
# 指定IP和端口
./bin/mongo --host=127.0.0.1 --port=27017
```

mongo shell 是一个js 控台，可以执行js 相关运算如:

```
> 1+1
2
> var i=123;
> print(i)
123
>
```

4.数据库与集合的基础操作

```
#查看数据库
show dbs;
#切换数据库
use luban;
#创建数据库与集合，在插入数据时会自动 创建数据库与集和
db.friend.insertOne({name:"wukong", sex:"man"});
#查看集合
show tables;
show collections;
#删除集合
db.friend.drop();
#删除数据库
db.dropDatabase();
```

三、MongoDB CRUD与全文索引

概要:

1. 数据的新增的方式
2. 数据的查询
3. 数据的修改删除
4. 全文索引查询

1. 数据的新增的方式

关于Mongodb数据插入的说明

1. 数据库的新增不需要序先设计模型结构，插入数据时会自动创建。
2. 同一个集合中不同数据字段结构可以不一样

插入相关方法:

```
//插入单条
db.friend.insertOne({name:"wukong", sex:"man"});
// 插入多条
db.friend.insertMany([
{name:"wukong",sex:"man"},{name:"diaocan",sex:"woman",age:18,birthday:new Date("1995-
11-02")},{name:"zixiao",sex:"woman"}
]);
// 指定ID
 db.friend.insert([
        {_id:1,name:"wokong",sex:"man",age:1},
        {_id:2,name:"diaocan",sex:"women",birthday:new Date("1988-11-    11")}
  ])
```

2、数据的查询

概要:

1. 基于条件的基础查询
2. $and、$or、$in、$gt、$gte、$lt、$lte 运算符
3. 基于 sort skip limit 方法实现排序与分页
4. 嵌套查询
5. 数组查询
6. 数组嵌套查询

基础查询:

```
#基于ID查找
db.emp.find({_id:1101})
#基于属性查找
db.emp.find({"name":"鲁班"})
# && 运算 与大于 运算
db.emp.find({"job":"讲师","salary":{$gt:8000}})
# in 运算
 db.emp.find({"job":{$in:["讲师","客服部"]}})
# or 运算
db.emp.find({$or:[{job:"讲师"  },{job:"客服部"}] })
```

排序与分页:

// sort skip limit

```
db.emp.find().sort({dep:1,salary:-1}).skip(5).limit(2)
```

嵌套查询:

```
# 错误示例：无结果
db.student.find({grade:{redis:87,dubbo:90 });
#错误示例：无结果
db.student.find({grade:{redis:87,dubbo:90,zookeper:85} })

# 基于复合属性查找 时必须包含其所有的值 并且顺序一至
db.student.find({grade:{redis:87,zookeper:85,dubbo:90} })

#基于复合属性当中的指定值 查找。注：名称必须用双引号
db.student.find({"grade.redis":87});

db.student.find({"grade.redis":{"$gt":80}});
```

数组查询：

```
db.subject.insertMany([
{_id:"001",name:"陈霸天",subjects:["redis","zookeper","dubbo"]},
{_id:"002",name:"张明明",subjects:["redis","Java","mySql"]},
{_id:"003",name:"肖炎炎",subjects:["mySql","zookeper","bootstrap"]},
{_id:"004",name:"李鬼才",subjects:["Java","dubbo","Java"]},
])
```

```
#无结果
db.subject.find({subjects:["redis","zookeper"]})
#无结果
db.subject.find({subjects:["zookeper","redis","dubbo"]})
# 与嵌套查询一样，必须是所有的值 并且顺序一至
db.subject.find({subjects:["redis","zookeper","dubbo"]})

# $all 匹配数组中包含该两项的值。注：顺序不作要求
db.subject.find({subjects:{"$all": ["redis","zookeper"]}})
注：
# 简化数组查询
db.subject.find({subjects:"redis"})
# 简化数组查询 ，匹配数组中存在任意一值。与$all相对应
db.subject.find({subjects:{$in: ["redis","zookeper"]}})
```

数组嵌套查询：

```
#基础查询 ，必须查询全部，且顺序一至
db.subject2.find({subjects:{name:"redis",hour:12} })
#指定查询第一个数组 课时大于12
db.subject2.find({"subjects.0.hour":{$gt:12}})
#查询任科目 课时大于12
db.subject2.find({"subjects.hour":{$gt:12}})
# $elemMatch 元素匹配，指定属性满足，且不要求顺序一至
db.subject2.find({subjects:{$elemMatch:{name:"redis",hour:12}}})

# 数组中任意元素匹配 不限定在同一个对象当中
db.subject2.find({"subjects.name":"mysql","subjects.hour":120})
```

修改

```
#设置值
db.emp.update({_id:1101} ,{ $set:{salary:10300}  })
#自增
db.emp.update({_id:1101} ,{ $inc:{salary:200}})

#基于条件 更新多条数据
# 只会更新第一条
db.emp.update({"dep":"客服部"},{$inc:{salary:100}})
# 更新所有 匹配的条件
db.emp.updateMany({"dep":"客服部"},{$inc:{salary:100}})
```

3、数据的修改与删除

修改

```
#设置值
db.emp.update({_id:1101} ,{ $set:{salary:10300}  })
#自增
db.emp.update({_id:1101} ,{ $inc:{salary:200}})

#基于条件 更新多条数据
# 只会更新第一条
db.emp.update({"dep":"客服部"},{$inc:{salary:100}})
# 更新所有 匹配的条件
db.emp.updateMany({"dep":"客服部"},{$inc:{salary:100}})
```

删除:

```
// 基于查找删除
db.emp.deleteOne({_id:1101})
// 删除整个集合
db.project.drop()
// 删除库
db.dropDatabase()
```

4、全文索引

索引的创建

```
db.project.createIndex({name:"text",description:"text"})
```

基于索引分词进行查询

```
db.project.find({$text:{$search:"java jquery"}})
```

基于索引 短语

```
db.project.find({$text:{$search:"\"Apache ZooKeeper\""}})
```

过滤指定单词

```
db.project.find({$text:{$search:"java apache -阿里"}})
```

查看执行计划

```
db.project.find({$text:{$search:"java -阿里"}}).explain("executionStats")
```