目录 CONTENTS

1 单机安装

2 快速入门

3 查询

4 更新

5 安全

6 其他命令

01 MongoDB安装

1. 官网下载安装介质：https://www.mongodb.com/download-center，选择适当的版本，这里以linux版本为例；
2. 解压到系统某路径， tar -xvzf mongodb-linux-x86_64-rhel70-3.4.10.tgz 并在安装目录创建data目录，以及logs目录和logs/mongodb.log文件
3. 使用vim在mongodb的bin目录创建mongodb的配置文件，如：vim bin/mongodb.conf，mongodb.conf内容请见下一页课件；
4. 编写shell脚本，命名为start-mongodb.sh，脚本内容如下：

   nohup ./mongod -f mongodb.conf &
5. 使用start-mongodb.sh启动mongodb实例，如：./start-mongodb
6. 使用mongoClient进行测试，通过restAPI地址测试（端口加1000）

02 配置文件

```
storage:
   dbPath: "/usr/local/apache/mongoDB/mongodb-linux-x86_64-rhel70-3.4.10/data"
systemLog:
   destination: file
   path: "/usr/local/apache//mongoDB/mongodb-linux-x86_64-rhel70-3.4.10/logs/mongodb.log"
net:
   port: 27022
   http:
      RESTInterfaceEnabled: true
processManagement:
   fork: false
```

mongoDB配置文件参数详解

C目录 ONTENTS

1 单机安装

2 快速入门

3 查询

4 更新

5 安全

6 其他命令

01 数据结构介绍

- **人员信息**

```
{
        "_id" : ObjectId("59f938235d93fc4af8a37114"),
        "username" :   “lison",
        "country" : "in11digo",
        "address" : {
                "aCode" : "邮编",
                "add" : "d11pff"
        },
        "favorites" : {
                "movies" : ["杀破狼2","1dushe","雷神1"],
                "cites" : ["1sh","1cs","1zz"]
        },
        "age" : 18 ,
        "salary" : NumberDecimal("2.099"),
        "lenght"  : 1.79
}
```

02 需求描述

- **新增2人**
- **删除**
  - ✓ delete from users where username = ‘lison’
  - ✓ delete from users where age >8 and age <25
- **修改**
  - ✓ update users set age=6 where username = lison'
  - ✓ update users set favorites.movies add "小电影2 ", "小电影3" where favorites.cites has "东莞"
- **查询**
  - ✓ select * from users where favorites.cites has "东莞"、"东京"
  - ✓ select * from users where username like '%s%' and (country= English or country= USA)

03 快速入门我们学习的目标是什么？

- ✓ 直观感受mongoDB的魅力
- ✓ mongo开发入门（原生、spring）
- ✓ 开发框架版本选择
- ✓ mongoDB数据类型全解析
- ✓ 对nosql的理念有初步的认识

04 直观感受mongoDB？客户端 & 图形化界面

- mongoDB自带的命令客户端 mongo
- mongoDB自带的图形化客户端：Compass Community
- 第三方mongoDB的图形化客户端：robomongo

产品	查看、新增和删除db和collections	对doccument的增删改查支持	构建和运行脚本	查看查询计划	管理索引	版本支持
Compass community	支持	支持，比较人性化	不支持	支持	支持	3.6
robomongo	支持	支持	支持	不支持	支持	3.4

05 原生java客户端

pom文件

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongo-java-driver</artifactId>
    <version>3.5.0</version>
</dependency>
```

Tips:

1. 3.5.0最新版本加入了对pojo的支持；
2. 3.5.0最新版本增强对json的支持；
3. mongodb原生客户端支持两种document和pojo模式的开发；

06 Spring-data-mongodb客户端

- **pom文件**

```
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-mongodb</artifactId>
    <version>1.10.9.RELEASE</version>
</dependency>
```

Tips:

1. spring-data-mongodb的最新版本是2.x.x，如果是spring为5.0版本以上的才推荐使用；
2. spring-data-mongodb的1.10.9版本基于spring4.3.x开发，但是默认依赖的mongodb驱动为2.14.3，可以将mongodb的驱动设置为3.5.0的版本，兼容性待进一步测试；
3. spring-data-mongodb的1.10.9可能会支持3.5.0版本 mongodb驱动；
4. spring-data-mongodb一般使用pojo的方式开发；

07 开发框架版本选择

- java 驱动与mongoDB兼容性

Java Driver Version	MongoDB 2.6	MongoDB 3.0	MongoDB 3.2	MongoDB 3.4	MongoDB 3.6
Version 3.6	✓	✓	✓	✓	✓
Version 3.5	✓	✓	✓	✓	
Version 3.4	✓	✓	✓	✓	
Version 3.3	✓	✓	✓		
Version 3.2	✓	✓	✓		
Version 3.1	✓	✓			
Version 3.0	✓	✓			

08 开发框架版本选择

- **java 驱动与jdk的兼容性**

Java Driver Version	Java 5	Java 6	Java 7	Java 8
Version 3.6		✓	✓	✓
Version 3.5		✓	✓	✓
Version 3.4		✓	✓	✓
Version 3.3		✓	✓	✓
Version 3.2		✓	✓	✓
Version 3.1		✓	✓	✓
Version 3.0		✓	✓	✓

Tips：java驱动优先使用3.5以上版本

09 开发框架版本选择

- **spring data mongo 与java mongo驱动兼容性**

spring mongodb 版本	spring版本支持	jdk版本支持	mongodb server支持	默认的mongoDB java驱动版本
Spring Data MongoDB 1.x	4.3.13.RELEASE 以上	jdk 1.6以上	2.6版本以上，3.4以下	2.14.3
Spring Data MongoDB 2.x	5.0.2.RELEASE 以上	jdk 1.8以上	2.6版本以上，3.6	3.5.0

Tips：Spring Data MongoDB 1.x 与 2.x之间区别比较大，请点击我

想要使用**spring mongoDB 2.x的新API，同时想使用3.5.0以上版本的java驱动？**

- ✓ spring-data-mongodb的1.10.9版本
- ✓ mongodb 3.5
- ✓ spring 4.3以上

10 mongoDB数据类型

数据类型	示例	说明
null	{"key":null}	null表示空值或者不存在该字段
布尔	{"key","true"} {"key","false"}	布尔类型表示真或者假
32位整数	{"key":8}	存储32位整数，但再shell界面显示会被自动转成64位浮点数
64位整数	{"key":{"floatApprox":8}}	存储64位整数，floatApprox意思是使用64位浮点数近似表示一个64位整数
64位浮点数	{"key":8.21}	存储64位整数，shell客户端显示的数字都是这种类型；
字符串	{"key":"value"} {"key":"8"}	UTF-8格式
对象ID	{"key":ObjectId()}	12字节的唯一ID
日期	{"key":new Date()}	
代码	{"key":function(){}}	
二进制数据		主要存储文件
未定义	{"key":undefined}	值没有定义，null和undefined是不同的
数组	{"key":[16,15,17]}	集合或者列表
内嵌文档	{"user":{"name":"lison"}}	子对象
Decimal128	{"price":NumberDecimal("2.099")}	3.4版本新增的数据类型，无精度问题

目录 CONTENTS

1 单机安装

2 快速入门

3 **查询**

4 更新

6 安全

7 其他命令

01 查询概要

- MongoDB 查询数据的语法格式如下：

```
db.collection.find(query, projection)
```

- ✓ **query** ：可选，使用查询操作符指定查询条件
- ✓ **projection** ：可选，使用投影操作符指定返回的键。查询时返回文档中所有键值， 只需省略该参数即可（默认省略）。

需要以易读的方式来读取数据，可以使用 pretty() 方法：

02 查询选择器

运算符类型	运算符	描述
范围	$eq	等于
	$lt	大于
	$gt	小于
	$lte	小于等于
	$gte	大于等于
	$in	判断元素是否在指定的集合范围里
	$all	判断数组中是否包含某几个元素,无关顺序
	$nin	判断元素是否不在指定的集合范围里
布尔运算	$ne	不等于
	$not	不匹配结果
	$or	有一个条件成立则匹配
	$nor	所有条件都不匹配
	$and	所有条件都必须匹配
	$exists	判断元素是否存在
其他	.	子文档匹配
	$regex	正则表达式匹配

03 查询选择

映射

- ✓ 字段选择：db.users.find({},{'username':1})
- ✓ 字段排除：db.users.find({},{'username':0})
- ✓ 数组子元素选择：db.users.find({},{'favorites.movies':{'$slice':[1,2]},'favorites.cites':1})

  $slice可以取两个元素数组,分别表示跳过和限制的条数；

排序

- ✓ sort()：db.orders.find().sort({'orderTime':1,'price':1})

  1：升序　2：降序

跳过和限制

- ✓ skip(n)：跳过n条数据
- ✓ limit(n)：限制n条数据

  e.g：db.orders.find().sort({'orderTime':-1}).limit(5).skip(5)

查询唯一值

- ✓ distinct()：查询指定字段的唯一值，e.g：db.users.distinct（“age”）

04 Java客户端解析

原生java驱动

- ✓ MongoClient → MongoDatabase →MongoCollection
  - MongoClient被设计成线程安全、可以被多线程共享的。通常访问数据库集群的应用只需要一个实例
  - 如果需要使用pojo对象读写，需要将PojoCodecProvider注入到client中
- ✓ 查询和更新的API类

  查询器：com.mongodb.client.model.Filters

  更新器：com.mongodb.client.model.Updates

05 Spring mongodb解析

Xml配置文件

```
<!-- mongodb连接池配置 -->
<!-- mongodb连接池配置 -->
<mongo:mongo-client host="192.168.1.129" port="27017">
    <mongo:client-options
          write-concern="ACKNOWLEDGED"
          connections-per-host="100"
          threads-allowed-to-block-for-connection-multiplier="5"
          max-wait-time="120000"
      connect-timeout="10000"/>
</mongo:mongo-client>>

<!-- mongodb数据库工厂配置 -->
<mongo:db-factory dbname="lison" mongo-ref="mongo" />

<!-- mongodb模板配置 -->
<bean id="anotherMongoTemplate"
class="org.springframework.data.mongodb.core.MongoTemplate">
    <constructor-arg name="mongoDbFactory" ref="mongoDbFactory" />
    <property name="writeResultChecking" value="EXCEPTION"></property>
</bean>
```

06 Spring mongodb解析

Spring mongodb开发

- ✓ 模板模式，基于MongoOperations进行操作，基于pojo的操作，配合@document注解开发；
- ✓ 查询和更新的API类

  查询器：org.springframework.data.mongodb.core.query.Query

  更新器：org.springframework.data.mongodb.core.query.Update

07 Mongodb连接池配置

参数名	默认值	说明
writeConcern	ACKNOWLEDGED	写入安全机制，是一种客户端设置，用于控制写入安全的级别: ACKNOWLEDGED 默认选项，数据写入到Primary就向客户端发送确认 0 Unacknowledged 对客户端的写入不需要发送任何确认，适用于性能要求高，但不关注正确性的场景; 1 W1 数据写入后，会等待集群中1台发送确认 2 W2 数据写入后，会等待集群中两台台发送确认 3 W3 数据写入后，会等待集群中3台台发送确认 JOURNALED 确保所有数据提交到 journal file MAJORITY 等待集群中大多数服务器提交后确认;
codecRegistry	MongoClient.getDefaultCodecRegistry()	编解码类，实现Codec接口
minConnectionsPerHost		最小连接数，connections-per-host
connectionsPerHost	100	最大连接数
threadsAllowedToBlockForConnectionMultiplier	5	此参数跟connectionsPerHost的乘机为一个线程变为可用的最大阻塞数，超过此乘机数之后的所有线程将及时获取一个异常
maxWaitTime	1000 * 60 * 2	一个线程等待链接可用的最大等待毫秒数，0表示不等待
maxConnectionIdleTime		设置池连接的最大空闲时间，0表示没有限制
maxConnectionLifeTime		设置池连接的最大使用时间，0表示没有限制
connectTimeout	1000*10	连接超时时间
alwaysUseMBeans	false	是否打开JMX监控

08 Mongodb连接池配置

参数名	默认值	说明
heartbeatFrequency	10000	设置心跳频率。 这是驱动程序尝试确定群集中每个服务器的当前状态的频率。
minHeartbeatFrequency	500	设置最低心跳频率。 如果驱动程序必须经常重新检查服务器的可用性，那么至少要等上一次检查以避免浪费。
heartbeatConnectTimeout	20000	心跳检测连接超时时间
heartbeatSocketTimeout	20000	心跳检测Socket超时时间

C目录 ONTENTS

1 单机安装

2 快速入门

3 查询

4 **更新**

5 索引

6 安全

7 其他命令

01 更新操作概要

- 性能更好
- 原子性操作

更新的方法

- 替换更新
- 操作符更新

update() 方法用于更新已存在的文档。语法格式如下：

db.collection.update(<query>, <update>, { upsert: <boolean>, multi: <boolean>, writeConcern: <document> })

参数说明：

- **query :** update的查询条件，类似sql update查询内where后面的；
- **update :** update的对象和一些更新的操作符（如$,$inc...）等，也可以理解为sql update查询内set后面的
- **upsert :** 可选，这个参数的意思是，如果不存在update的记录，是否插入objNew,true为插入，默认是false，不插入。
- **multi :** 可选，mongodb 默认是false,只更新找到的第一条记录，如果这个参数为true,就把按条件查出来多条记录全部更新。
- **writeConcern :**可选，写安全配置。

02 更新选择器

类型	运算符	描述
操作符	$inc	指定值加n
	$set	更新指定字段
	$unset	将指定字段删除
	$rename	更新字段名称
数组操作符	$	定位到某一个元素
	$push	添加值到数组中
	$addToSet	添加值到数组中，有重复则不处理
	$pop	删除数组第一个或者最后一个
	$pull	从数组中删除匹配查询条件的值
	$pullAll	从数组中删除多个值
数组运算修饰符	$each	与$push和$addToSet一起使用来操作多个值
	$slice	与$push和$each一起使用来操作用来缩小更新后数组的大小
	$sort	与$push、$each和$slice一起使用来对数组进行排序
隔离运算符	$isolated	隔离其他操作，不允许其他操作交叉更新，不能再分片中使用

03 更新的原子性

1. mongodb的更新都是原子的，mongodb所有的写操作都是有锁的。mongoDB 2.2之前锁级别为实例级别，mongoDB 2.2以后版本锁级别为数据库级别，mongoDB 3.0中，WiredTiger的锁级别是集合级别；
2. findAndModify命令：在同一往返过程中原子更新文档并返回它；
3. $isolated命令：mongoDB对写锁进行了优化，对长时间的写操作会为其他的读写操作让路，使用$isolated可以避免这种让路，隔离其他操作，不允许其他操作交叉更新，不能再分片中使用；

C目录 ONTENTS

1 单机安装

2 快速入门

3 查询

4 更新

5 **安全**

6 其他命令

01 Role-Based Access Control 基于角色的控制

角色类型	类型说明	角色名称	说明
数据库一般角色（Database User Roles）	每个数据库都包含的一般角色；	read	提供读取所有非系统集合和部分系统集合的数据的能力，系统集合包括：system.indexes，system.js和system.namespaces集合。
		readWrite	提供read角色的所有权限以及修改所有非系统集合和system.js集合上的数据的能力。
数据库管理角色（Database Administration Roles）	每个数据库都包含的吧数据库管理角色；	dbAdmin	提供执行管理任务的能力，如与模式相关的任务，索引，收集统计信息。 此角色不授予用户和角色管理的权限。
		userAdmin	提供在当前数据库上创建和修改角色和用户的能力。
		dbOwner	提供对数据库执行任何管理操作的能力。 此角色结合了readWrite，dbAdmin和userAdmin角色授予的权限。
集群管理角色（Cluster Administration Roles）	用于管理整个数据库集群系统而不是特定数据库的角色。 这些角色包括但不限于副本集和分片群集管理功能。	clusterManager	在集群上提供管理和监视操作。 具有此角色的用户可以分别访问在分片和复制中使用的config和local数据库。
		clusterMonitor	为监控工具（如MongoDB Cloud Manager和Ops Manager监控代理）提供只读访问权限。
		hostManager	提供监视和管理服务器的能力。
		clusterAdmin	提供权限最高的群集管理访问。 此角色结合了由clusterManager，clusterMonitor和hostManager角色授予的权限。 此外，该角色还提供了dropDatabase操作。

02 Role-Based Access Control 基于角色的控制

角色类型	类型说明	角色名称	说明
备份和恢复角色（Backup and Restoration Roles）	用于专门的备份和恢复的角色	backup	提供备份数据所需的权限。 此角色提供足够的权限来使用MongoDB Cloud Manager备份代理，Ops Manager备份代理或使用mongodump。
		restore	提供使用mongorestore恢复数据所需的权限
全数据库角色（All-Database Roles）	适用于除mongod实例中的local和config之外的所有数据库：	readAnyDatabase	提供与读取相同的只读权限，除了适用于群集中除本地和配置数据库以外的所有权限。 该角色还提供了整个集群上的listDatabases操作。
		readWriteAnyDatabase	提供与readWrite相同的读取和写入权限，除了它适用于群集中除本地和配置数据库以外的所有数据。 该角色还提供了整个集群上的listDatabases操作。
		userAdminAnyDatabase	提供与userAdmin相同的用户管理操作访问权限，除了适用于群集中除本地数据库和配置数据库外的所有数据。
		dbAdminAnyDatabase	提供与dbAdmin相同的数据库管理操作访问权限，除了它适用于除集群中的本地数据库和配置数据库以外的所有数据库管理操作。 该角色还提供了整个集群上的listDatabases操作。
超级角色（Superuser Roles）	所有资源的完整权限	root	提供对readWriteAnyDatabase，dbAdminAnyDatabase，userAdminAnyDatabase，clusterAdmin，还原和备份相结合的操作和所有资源的访问。

03 客户端授权

shell脚本创建用户

- ✓ db.createUser({'user':'boss', 'pwd':'boss', 'roles':[{'role':'userAdminAnyDatabase', 'db':'admin'}]})
- ✓ db.createUser({'user':'lison','pwd':'lison','roles':[{'role':'readWrite','db':'lison'}]})

Tips:

1. 服务器启动需要加上auth参数连接服务器才需要验证

   如：./mongod -f mongodb.conf --auth
2. 切换到数据库上，才能给当前数据库创建用户；

04 客户端授权

Java客户端安全认证

MongoCredential类包括每个受支持的身份验证机制的静态工厂方法。

```
public static MongoCredential createCredential(final String userName,
                                               final String database,
                                               final char[] password)
```

C目录 ONTENTS

1 单机安装

2 快速入门

3 查询

4 更新

6 安全

7 其他命令

01 其他常用命令

- ✓ show dbs ：显示数据库列表
- ✓ show collections ：显示集合列表
- ✓ db.stats() ：显示数据库信息
- ✓ db.help()，db.collection.help()：内置帮助，显示各种方法的说明；