Table of Contents

简介

- MongoDB 是一个基于分布式 文件存储的NoSQL数据库
- 由C++语言编写，运行稳定，性能高
- 旨在为 WEB 应用提供可扩展的高性能数据存储解决方案
- 查看官方网站

MongoDB特点

- 模式自由 :可以把不同结构的文档存储在同一个数据库里
- 面向集合的存储：适合存储 JSON风格文件的形式
- 完整的索引支持：对任何属性可索引
- 复制和高可用性：支持服务器之间的数据复制，支持主-从模式及服务器之间的相互复制。复制的主要目的是提供冗余及自动故障转移
- 自动分片：支持云级别的伸缩性：自动分片功能支持水平的数据库集群，可动态添加额外的机器
- 丰富的查询：支持丰富的查询表达方式，查询指令使用JSON形式的标记，可轻易查询文档中的内嵌的对象及数组
- 快速就地更新：查询优化器会分析查询表达式，并生成一个高效的查询计划
- 高效的传统存储方式：支持二进制数据及大型对象（如照片或图片）

基本操作

- MongoDB将数据存储为一个文档，数据结构由键值(key=>value)对组成
- MongoDB文档类似于JSON对象，字段值可以包含其他文档、数组、文档数组

名词

SQL术语/概念	**MongoDB**术语/概念	解释/说明
database	database	数据库
table	collection	数据库表/集合
row	document	数据记录行/文档
column	field	数据字段/域
index	index	索引
table joins		表连接,MongoDB不支持
primary key	primary key	主键,MongoDB自动将_id字段设置为主键

- 三元素：数据库，集合，文档
  - 集合就是关系数据库中的表
  - 文档对应着关系数据库中的行
- 文档，就是一个对象，由键值对构成，是json的扩展Bson形式

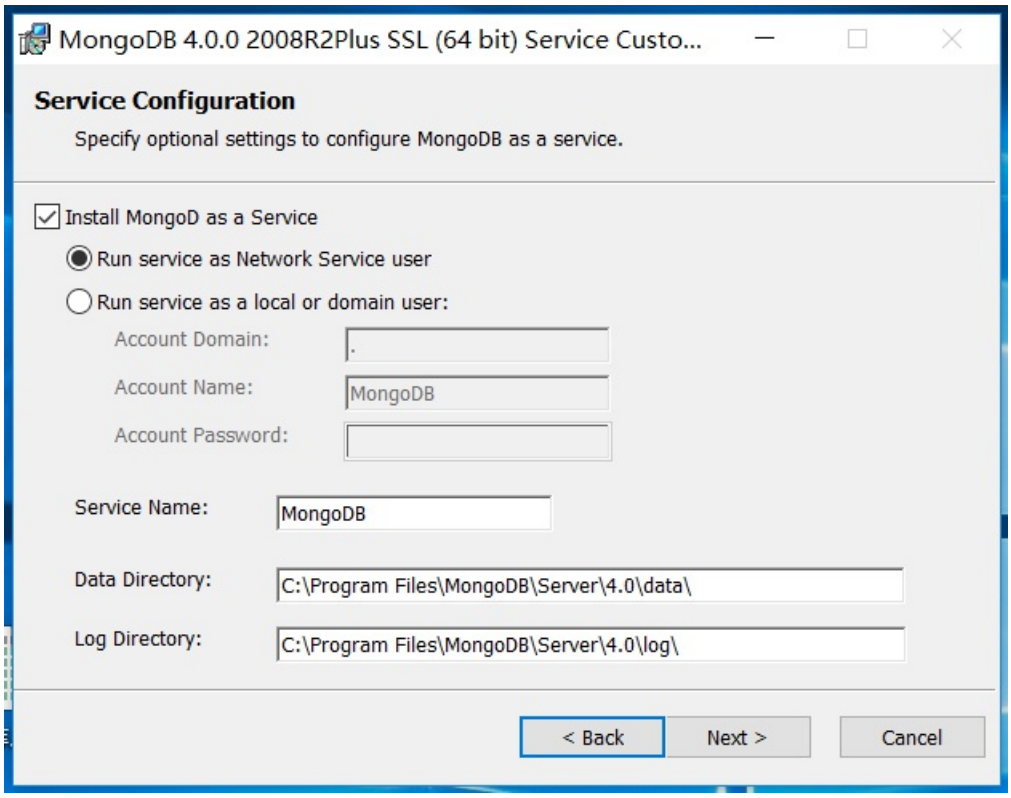
```
{'name':'guojing','gender':'男'}
```

- 集合：类似于关系数据库中的表，储存多个文档，结构不固定，如可以存储如下文档在一个集合中

```
{'name':'guojing','gender':'男'}
{'name':'huangrong','age':18}
{'book':'shuihuzhuan','heros':'108'}
```

- 数据库：是一个集合的物理容器，一个数据库中可以包含多个文档
- 一个服务器通常有多个数据库

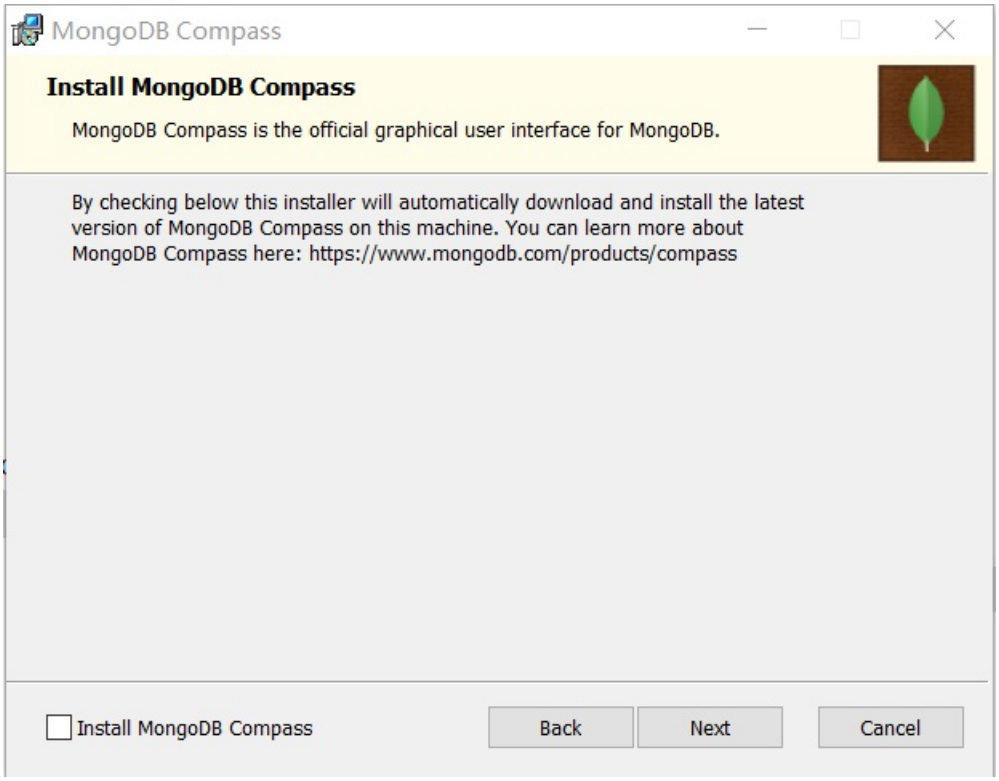
Windows环境安装



配置为服务

配置数据库路径

配置日志路径



取消install MongoDB Compass加快安装

安装完成测试 http://localhost:27017/

Ubuntu安装

- 在线安装

sudo apt-get install mongodb

- 版本号

mongo -version

- 服务启动停止重启

service mongodb stop

service mongodb start

service mongodb restart

- 查看是否启动

pgrep mongo

- 卸载**mongodb**

sudo apt-get --purge remove mongodb mongodb-server mongodb-clients

数据库切换

- 查看当前数据库名称

```
db
```

- 查看所有数据库名称
- 列出所有在物理上存在的数据库

```
show dbs
```

- 切换数据库
- 如果数据库不存在，则指向数据库，但不创建，直到插入数据或创建集合时数据库才被创建

```
use 数据库名称
```

- 默认的数据库为test，如果你没有创建新的数据库，集合将存放在test数据库中

数据库删除

- 删除当前指向的数据库
- 如果数据库不存在，则什么也不做

```
db.dropDatabase()
```

集合创建

- 语法

```
db.createCollection(name, options)
```

- name是要创建的集合的名称
- options是一个文档，用于指定集合的配置
- 选项参数是可选的，所以只需要到指定的集合名称。以下是可以使用的选项列表：
- 例1：不限制集合大小

```
db.createCollection("stu")
```

- 例2：限制集合大小，后面学会插入语句后可以查看效果
- 参数capped：默认值为false表示不设置上限，值为true表示设置上限
- 参数size：当capped值为true时，需要指定此参数，表示上限大小，当文档达到上限时，会将之前的数据覆盖，单位为字节

```
db.createCollection("sub", { capped : true, size : 10 } )
```

查看当前数据库的集合

- 语法

```
show collections
```

删除

- 语法

```
db.集合名称.drop()
```

数据类型

- 下表为MongoDB中常用的几种数据类型：
- Object ID：文档ID
- String：字符串，最常用，必须是有效的UTF-8
- Boolean：存储一个布尔值，true或false
- Integer：整数可以是32位或64位，这取决于服务器
- Double：存储浮点值
- Arrays：数组或列表，多个值存储到一个键
- Object：用于嵌入式的文档，即一个值为一个文档
- Null：存储Null值
- Timestamp：时间戳
- Date：存储当前日期或时间的UNIX时间格式

object id

- 每个文档都有一个属性，为_id，保证每个文档的唯一性
- 可以自己去设置_id插入文档
- 如果没有提供，那么MongoDB为每个文档提供了一个独特的_id，类型为objectID
- objectID是一个12字节的十六进制数
  - 前4个字节为当前时间戳
  - 接下来3个字节的机器ID
  - 接下来的2个字节中MongoDB的服务进程id
  - 最后3个字节是简单的增量值

插入

- 语法

```
db.集合名称.insert(document)
```

- 插入文档时，如果不指定_id参数，MongoDB会为文档分配一个唯一的ObjectId
- 例1

```
db.stu.insert({name:'gj',gender:1})
```

- 例2

```
s1={_id:'20160101',name:'hr'}
s1.gender=0
db.stu.insert(s1)
```

简单查询

- 语法

```
db.集合名称.find()
```

更新

- 语法

```
db.集合名称.update(<query>,<update>, {multi: <boolean>})
```

- 参数query:查询条件，类似sql语句update中where部分
- 参数update:更新操作符，类似sql语句update中set部分
- 参数multi:可选，默认是false，表示只更新找到的第一条记录，值为true表示把满足条件的文档全部更新
- 例3：全文档更新

```
db.stu.update({name:'hr'},{name:'mnc'})
```

- 例4：指定属性更新，通过操作符$set

```
db.stu.insert({name:'hr',gender:0})
db.stu.update({name:'hr'},{$set:{name:'hys'}})
```

- 例5：修改多条匹配到的数据

```
db.stu.update({},{$set:{gender:0}},{multi:true})
```

保存

- 语法

```
db.集合名称.save(document)
```

- 如果文档的_id已经存在则修改，如果文档的_id不存在则添加
- 例6

```
db.stu.save({_id:'20160102','name':'yk',gender:1})
```

- 例7

```
db.stu.save({_id:'20160102','name':'wyk'})
```

删除

- 语法

```
db.集合名称.remove(<query>, { justOne: <boolean>   })
```

- 参数query:可选，删除的文档的条件
- 参数justOne:可选，如果设为true或1，则只删除一条，默认false，表示删除多条
- 例8：只删除匹配到的第一条

```
db.stu.remove({gender:0},{justOne:true})
```

- 例9：全部删除

```
db.stu.remove({})
```

数据查询

基本查询

- 方法find()：查询

```
db.集合名称.find({条件文档})
```

- 方法findOne()：查询，只返回第一个

```
db.集合名称.findOne({条件文档})
```

- 方法pretty()：将结果格式化

```
db.集合名称.find({条件文档}).pretty()
```

比较运算符

- 等于，默认是等于判断，没有运算符
- 小于$lt
- 小于或等于$lte
- 大于$gt
- 大于或等于$gte
- 不等于$ne
- 例1：查询名称等于'gj'的学生

```
db.stu.find({name:'gj'})
```

- 例2：查询年龄大于或等于18的学生

```
db.stu.find({age:{$gte:18}})
```

逻辑运算符

- 查询时可以有多个条件，多个条件之间需要通过逻辑运算符连接
- 逻辑与：默认是逻辑与的关系
- 例3：查询年龄大于或等于18，并且性别为1的学生

```
db.stu.find({age:{$gte:18},gender:1})
```

- 逻辑或：使用$or
- 例4：查询年龄大于18，或性别为0的学生

```
db.stu.find({$or:[{age:{$gt:18}},{gender:1}]})
```

- and和or一起使用
- 例5：查询年龄大于18或性别为0的学生，并且学生的姓名为gj

```
db.stu.find({$or:[{age:{$gte:18}},{gender:1}],name:'gj'})
```

范围运算符

- 使用"$in"，"$nin" 判断是否在某个范围内
- 例6：查询年龄为18、28的学生

```
db.stu.find({age:{$in:[18,28]}})
```

支持正则表达式

- 使用//或$regex编写正则表达式
- 例7：查询姓黄的学生

```
db.stu.find({name:/^黄/})
db.stu.find({name:{$regex:'^黄'}})
```

自定义查询

- 使用$where后面写一个函数，返回满足条件的数据
- 例7：查询年龄大于30的学生

```
db.stu.find({$where:function(){return this.age>20}})
```

Limit

- 方法limit()：用于读取指定数量的文档
- 语法：

```
db.集合名称.find().limit(NUMBER)
```

- 参数NUMBER表示要获取文档的条数
- 如果没有指定参数则显示集合中的所有文档
- 例1：查询2条学生信息

```
db.stu.find().limit(2)
```

skip

- 方法skip()：用于跳过指定数量的文档
- 语法：

```
db.集合名称.find().skip(NUMBER)
```

- 参数NUMBER表示跳过的记录条数，默认值为0
- 例2：查询从第3条开始的学生信息

```
db.stu.find().skip(2)
```

一起使用

- 方法limit()和skip()可以一起使用，不分先后顺序
- 创建数据集

```
for(i=0;i<15;i++){db.t1.insert({_id:i})}
```

- 查询第5至8条数据

```
db.stu.find().limit(4).skip(5)
或
db.stu.find().skip(5).limit(4)
```

投影

- 在查询到的返回结果中，只选择必要的字段，而不是选择一个文档的整个字段
- 如：一个文档有5个字段，需要显示只有3个，投影其中3个字段即可
- 语法：
- 参数为字段与值，值为1表示显示，值为0不显示

```
db.集合名称.find({},{字段名称:1,...})
```

- 对于需要显示的字段，设置为1即可，不设置即为不显示
- 特殊：对于_id列默认是显示的，如果不显示需要明确设置为0
- 例1

```
db.stu.find({},{name:1,gender:1})
```

- 例2

```
db.stu.find({},{_id:0,name:1,gender:1})
```

排序

- 方法sort()，用于对结果集进行排序
- 语法

```
db.集合名称.find().sort({字段:1,...})
```

- 参数1为升序排列
- 参数-1为降序排列
- 例1：根据性别降序，再根据年龄升序

```
db.stu.find().sort({gender:-1,age:1})
```

统计个数

- 方法count()用于统计结果集中文档条数
- 语法

```
db.集合名称.find({条件}).count()
```

- 也可以与为

```
db.集合名称.count({条件})
```

- 例1：统计男生人数

```
db.stu.find({gender:1}).count()
```

- 例2：统计年龄大于20的男生人数

```
db.stu.count({age:{$gt:20},gender:1})
```

消除重复

- 方法distinct()对数据进行去重
- 语法

```
db.集合名称.distinct('去重字段',{条件})
```

- 例1:查找年龄大于18的性别（去重）

```
db.stu.distinct('gender',{age:{$gt:18}})
```

备份

- 语法

```
mongodump -h dbhost -d dbname -o dbdirectory
```

- -h：服务器地址，也可以指定端口号
- -d：需要备份的数据库名称
- -o：备份的数据存放位置，此目录中存放着备份出来的数据
- 例1

```
sudo mkdir test1bak
sudo mongodump -h 192.168.196.128:27017 -d test1 -o ~/Desktop/test1bak
```

恢复

- 语法

```
mongorestore -h dbhost -d dbname --dir dbdirectory
```

- -h：服务器地址
- -d：需要恢复的数据库实例
- --dir：备份数据所在位置
- 例2

```
mongorestore -h 192.168.196.128:27017 -d test2 --dir ~/Desktop/test1bak/test1
```

与python交互

- 点击查看 官方文档
- 安装python包

```
pip install pymongo
```

- 引入包pymongo

```
import pymongo
```

- 连接，创建客户端

```
client=pymongo.MongoClient("localhost", 27017)
```

- 获得数据库test1

```
db=client.test1
```

- 获得集合stu

```
stu = db.stu
```

- 添加文档

```
s1={name:'gj',age:18}
s1_id = stu.insert_one(s1).inserted_id
```

- 查找一个文档

```
s2=stu.find_one()
```

- 查找多个文档1

```
for cur in stu.find():
    print cur
```

- 查找多个文档2

```
cur=stu.find()
cur.next()
cur.next()
cur.next()
```

- 获取文档个数

```
print stu.count()
```