黑马程序员
www.itheima.com

MongoDB数据库

Mongodb数据库课程概要

1.认识NoSQL

2.安装与配置

3.mongodb的基本操作

4.mongodb的查询操作

5.mongodb的聚合操作

6.mongodb的索引

7.备份与恢复

8.Python与mongodb的交互

https://my.oschina.net/qinlinwang/blog/82295

1.认识NoSQL和MongoDB

1.0 数据库的分类

1.数据库的分类:

SQL数据库(关系型数据库)

Oracle

MySQL

SQL Server

NoSQL数据库(非关系型数据库)

mongodb

redis

hbase

关系型和非关系型的介绍

```
{    customer_id : 1,
     first_name : "Mark",
     last_name : "Smith",
     city : "San Francisco",
     phones: [ {
          type : "work",
          number: "1-800-555-1212"
     },
     {    type : "home",
          number: "1-800-555-1313",
          DNC: true
     },
     {    type : "home",
          number: "1-800-555-1414",
          DNC: true
     }
   ]
}
```

关系型和非关系型的介绍

Relational

关系数据库很强大，但是它并不能很好的应付所有的应用场景。
MySQL的扩展性差，大数据下IO压力大， 表结构更改困难

MongoDB

易扩展，大数据量高性能，灵活的数据模型，高可用

1.认识NoSQL和MongoDB

1.0 数据库的分类

2.关系性数据库在当前时代的缺点:

1)数据格式单一

2)高并发读写性能低

3)可扩展性低

1.认识NoSQL和MongoDB

1.1 NoSQL

1.1.1 什么是NoSQL

指的是非关系型的数据库，是相对于关系型数据库的统称，主要是解决大规模数据存储的挑战

1.1.2 为什么要使用NoSQL

易扩展

读写速度快

高性能和高可扩展

非结构化与不可预知数据(数据模型灵活)

1.认识NoSQL和mongodb

1.1 NoSQL

1.1.3 NoSQL数据库的分类:

键值型	redis
文档型	mongodb
列存储	HBase
图形	

1.1.4 如何选择SQL还是NoSQL

SQL着重于关系，NoSQL着重于存储

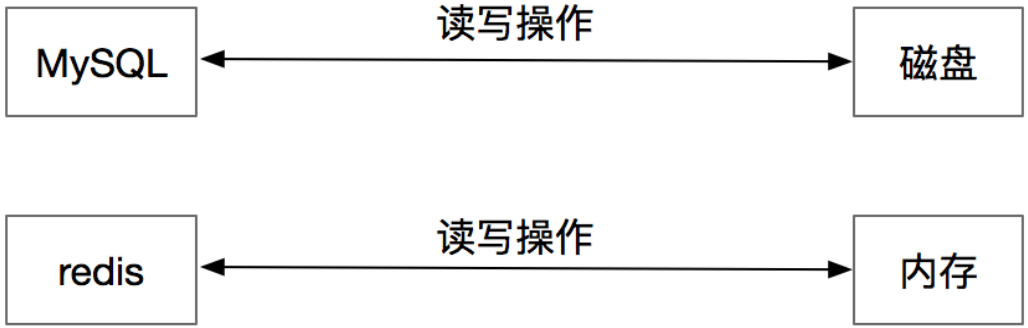
1.认识NoSQL和MongoDB

1.2 MongoDB

1.2.1 什么是MongoDB

可扩展的高性能，开源，面向文档(分布式文件存储)的NoSQL型的数据存库。

1.2.2 数据库存储方式的比较



内存用于缓存操作

mongoDB<----------->内存<----------->磁盘

1.认识NoSQL和MongoDB

1.2 MongoDB

1.2.2 为什么要使用MongoDB

高可扩展性

高性能存储

使用简单

部署简单

1.认识NoSQL和MongoDB

1.2 MongoDB

1.2.4 MongoDB中的重要指示点

①MongoDB中的三要素

数据库

集合 ------- 表

文档 ------ 行

②MongoDB中的数据存储是一Bson的形式存储的，Bson是二进制的json,所以看上去记录的形式类似于json数据

③MongoDB中集合中的数据不同于SQL型数据库中的数据，MongoDB中文档结构可以不同，因此扩展性非常好

1.认识NoSQL和MongoDB

1.2 MongoDB

1.2.3 MongoDB的主要应用场景

1.网站数据

网站实时操作比如插入，更新和查询

2.缓存

性能高

3.大量，低价值数据的存储

日志监控数据、爬虫数据

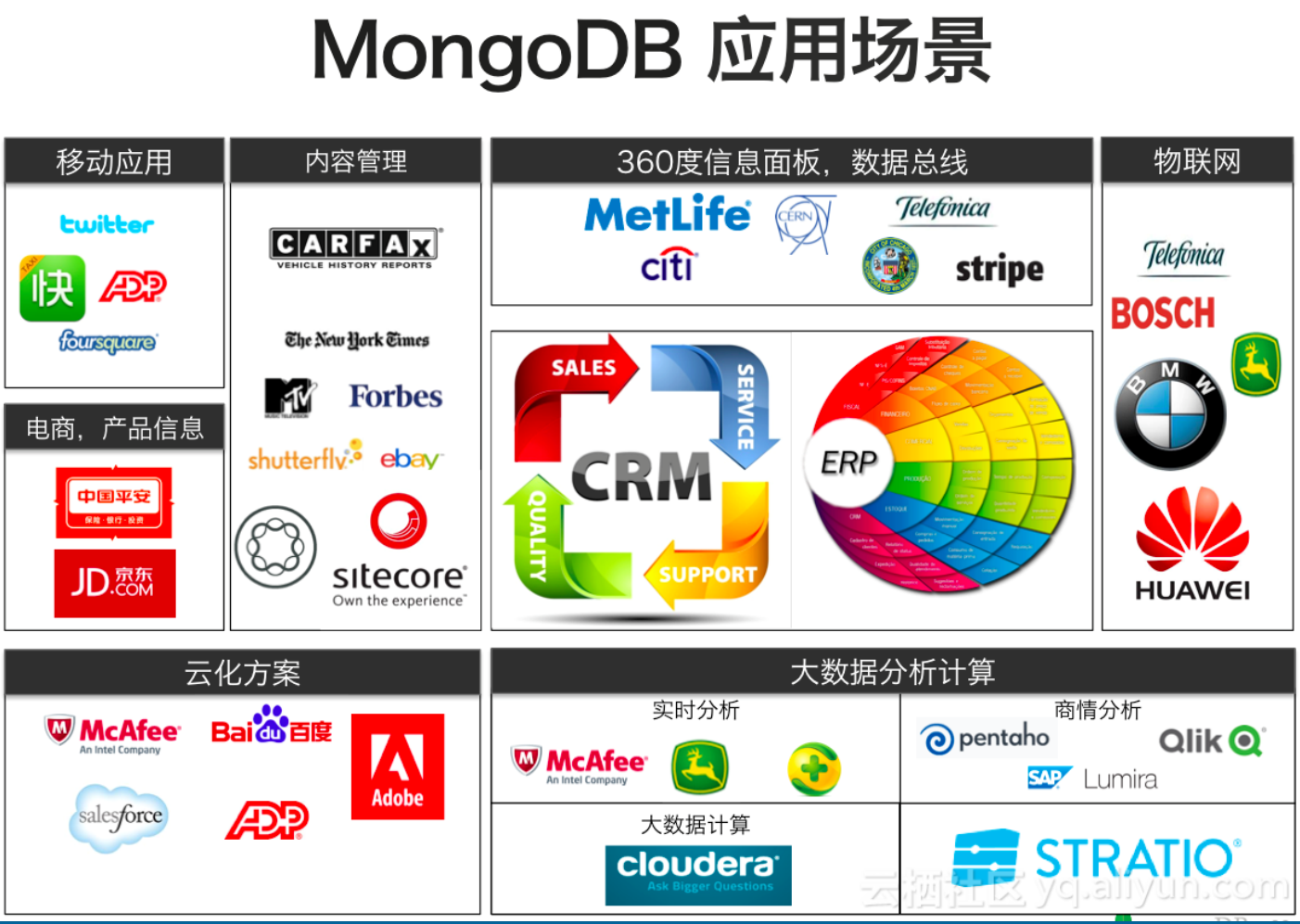
4.集群

5. json格式的数据

mongodb采用BSON(binary JSON)数据格式

1.认识NoSQL和MongoDB

1.2 MongoDB



2.MongoDB的安装配置

2.1 版本的选择

奇偶版本

奇数为开发版　　可能存在各种影响运行的bug

偶数为稳定版(release)　　基本没有运行bug

位数选择

32位　　存在存储上限

64位　　基本无限制

2.MongoDB的安装配置

2.2 安装配置

2.2.1Windows平台

1.去 https://www.mongodb.com/download-center#community 下载

2.双击运行下载后的数据库安装包

3.将安装路径添加环境变量，我的电脑/属性/高级/环境变量/系统配置/path/添加安装的路径；

4.创建数据存储文件的路径

5.管理员权限运行cmd，在其中运行mongod，开启数据库服务端

6.在另一个cmd中运行mongo查看是否连接成功

exception c:/data/db

2.MongoDB的安装配置

2.2 安装配置

2.2.2 linux平台

1.去 https://www.mongodb.com/download-center#community 下载

2. tar -zxvf 对应版本压缩包名

3. sudo mv -r 解压创建目录名/ /usr/local/mongodb

4. export PATH=/usr/local/mongodb/bin:$PATH

注:

默认的配置文件是 /etc/mongo.conf
默认的日志文件是 /var/log/mongodb/mongod.log

2.MongoDB的安装配置

2.2 安装配置

2.2.2 linux平台

服务的形式操作数据库

```
sudo service mongod start          启动数据库
sudo service mongod stop           关闭数据库
sudo service mongod restart        重启数据库
```

普通形式操作数据库

```
mkdir –p /data/db
sudo mongod
```

运行客户端

```
mongo              启动客户端
exit/ctrl+c        退出客户端
```

2.MongoDB的安装配置

2.2 安装配置

2.2.3 mac平台

brew update
brew install mognodb

普通形式操作数据库

mkdir –p /data/db
sudo mongod

2.MongoDB的安装配置

2.3 常用配置参数

1.sudo mongod 数据默认存储在/data/db

2.sudo mongod --dbpath=/User/data/db 使用指定的目录存放数据

3.sudo mongod --logpath=path --dbpath=path –logappend --fork

--fork 后台开启一个进程运行mongodb服务器
--logpath 指定日志输出文件
--logappend 设置日志的写入模式为追加(不会覆盖原有日志)

4.sudo mongod –f logfile
-f 指定加载配置文件

5.终端远程关闭服务器
use admin
db.shutdownServer()

2.MongoDB的安装配置

2.3 GUI---robomongo

2.3.0下载地址:

https://robomongo.org/download

2.3.1 robomongo的安装

tar zxvf robomongo-0.9.0-linux-x86_64-0786489.tar.gz

直接在bin目录下运行

2.3.2 简单使用

1.链接数据库

2.测试

.MongoDB的权限管理

.1 为什么要设置权限管理

刚安装完毕的mongodb默认不适用权限认证方式启动，然而公网运行系统需要设置权限以保证数据安全

.2 MongoDB的权限管理方案

①mongodb没有默认管理员账号,需要添加管理员账号，开启验证
②用户只能在用户所在的数据库登录
③管理员账号可以管理所有数据库

.3 开启权限认证的方式

权限认证默认是关闭的

①在启动数据库的时候添加--auth参数

```
sudo mongod --auth
```

②在配置文件中添加auth = true，然后加载配置文件启动

.MongoDB的权限管理

.3mongodb创建用户

1.所有的用户都必须使用管理员账户创建

use admin　　创建管理员用户使用的数据库

2.选择数据库后使用相应方法创建用户及权限

```
db.createUser({
        user:”用户名”,
        pwd:”密码”,
        roles:[{role:”权限”,db:”数据库”}]
})
```

roles为权限设置的文档，文档中db为指定的数据库， role为权限(最常用的为root,read,readWrite)

.MongoDB的权限管理

.3mongodb创建用户

3.查看已经创建的用户权限

```
show users
```

```
use admin
db.system.users.find()
```

4.数据库权限的认证

```
use dbname
db.auth(user,pwd)
```

5.删除某一用户及权限

```
db.dropUser(用户名)
```

3.MongoDB数据库的基本操作

0.MongoDB

专有名词

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

3.MongoDB数据库的基本操作

1 数据库的相关操作

1. db　　查看当前使用数据库
2. show dbs　　查看磁盘上存在的数据库
3. use dbname　　切换到指定的数据库
   1. 数据库不存在也可以使用
   2. 使用use之后并没有创建数据库
   3. 数据库是在手动创建集合或者使用集合的时候创建的
4. db.dropDatabase()　删除当前的数据库
   1. 使用该命令时必须已经选择了一个数据库

3.MongoDB数据库的基本操作

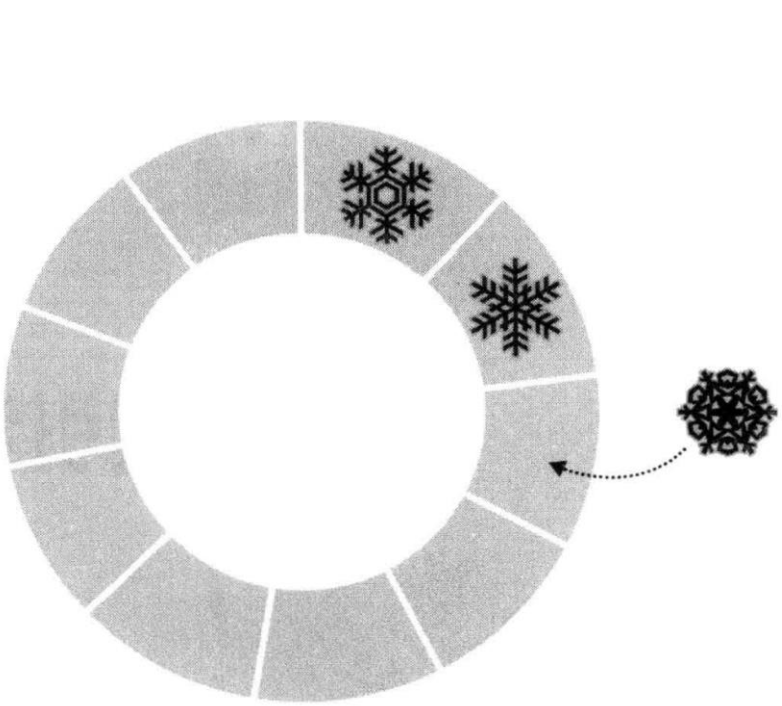
2 集合的相关操作

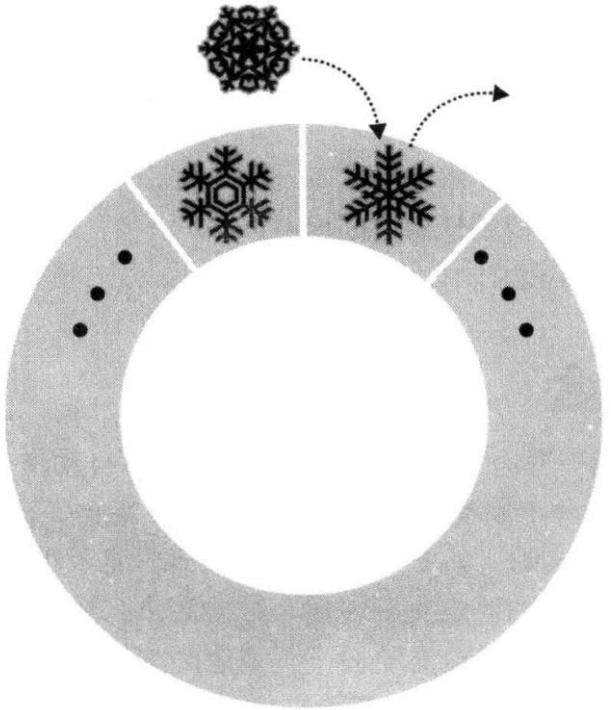
1	show collections	查看当前数据库中的集合列表(选择数据库之后使用)
2	db.createCollection(name)	创建集合
3	db.collection.drop()	删除集合

3.MongoDB数据库的基本操作

4. 固定集合

固定集合类似于循环队列，当没有存储空间的时候，最老的数据将会被新数据覆盖





3.MongoDB数据库的基本操作

2 集合的相关操作

4. 固定集合

1.db.createCollection('colname',{capped:true,size:n})

size的单位是字节

2. db.colname.isCapped()

判断一个集合是否为固定大小集合

3. db.runCommand({"convertToCapped":'colname',size:n})

将一个普通集合转换为固定大小集合，无法将固定集合转换为普通集合

固定大小集合特点:

插入速度快　　按照顺序查询速度快

新数据会替换旧数据　　不能使用remove删除数据

应用场景

存储日志信息　　缓存少量文档

4.MongoDB的数据类型

4.1MongoDB具有丰富的数据类型

1.ObjectID

4字节时间戳 3字节机器id 2字节进程id 3字节增量值

2.string类型

3.Boolean类型 true/false

4.integer类型

5.Double类型

6.Arrays类型

Python中的列表，js中的数组

7.Object类型

嵌入式的文档

8.null类型

9.Tiamestamp类型

10.Date类型

当前时间或者unix时间

5.MongoDB的基本操作

5.1查询操作

db.colname.find(query)

根据查询条件进行查询，并返回查询结果

5.2插入操作

1.单条数据插入

db.colname.insert(data)

data为要插入的数据，格式类似json，或者Python字典,可以直接输入一个字典格式的数据，也可以存一个变量再插入这个变量。

2.多条数据插入

db.colname.insert([data_list])

data_list为列表，列表中的每一项都是文档格式(python 字典)

5.MongoDB的基本操作

5.3 更新操作

语法

db.colname.update({query},{update})

query为查询条件，update为更新数据

1.全文档覆盖更新

db.colname.update({query},{key,vlaue})

将通过query条件查询出来的文档替换为第二个参数指定的数据

2.指定键值更新

db.colname.update({query},{$set:{key:value}})

将通过query条件查询出来的文档的指定属性设置为指定的值，而不会覆盖原有的其他数据，如果更新值存在则更新，不存在则添加

5.MongoDB的基本操作

5.3 更新操作

3.批量键值更新

db.colname.update(query,{$set:{key:value}},{multi:true})

multi决定是否是批量更新还是只更新一条数据，并且只有在对数据字段值操作(使用$)的时候才能使用

multi如果不指定默认为false

4.不指定查询条件进行修改

db.colname.update({ },{update},{multi:true})

5.MongoDB的基本操作

5.4删除数据

指定删除

db.colname.remove(query,{justOne:boolean})

justOne对应的值决定删除单条数据还是多条数据

删除全部数据

db.colname.remove({})

5.5保存操作

db.colname.save(data)

save = insert + update

以_id为判断依据，存在该id对应的数据则更新数据，不存在则插入数据

5.MongoDB的查询操作

5.1 基本操作

1.多数据查询

db.colname.find(query)

1.带query则是条件查询

2.不带query是显示集合中的所有数据

2.单条数据查询

db.colname.findOne(query)

查询单条数据的时候，返回结果自动美化

3.查询数据美化

db.colname.find().pretty()

```
db.stu.drop()
db.stu.insert({nar
',age:20,gender:t
db.stu.insert({nar
岛',age:18,gende
db.stu.insert({nar
',age:18,gender:fa
db.stu.insert({nar
花岛',age:40,gen
 db.stu.insert({na
理',age:16,gende
 db.stu.insert({na
大理',age:45,gen
db.stu.insert({nar
山',age:18,gende
```

6.MongoDB的运算符

6.1 比较运算符

$lt	小于	less than
$lte	小于等于	less than equal
$gt	大于	great than
$gte	大于等于	great than equal
$ne	不等于	not equal

比较运算符在查询中格式

```
db.colname.find({key:{$lt:n}})
```

6.MongoDB的运算符

6.2 逻辑运算符

多条件与查询

只返回所有条件都满足的文档

1.同一个字典中键值对的形式

```
db.colname.find({key:value,key1:value1})
```

2.使用and运算符查询的形式

```
db.colname.find({$and:[{key:value},{key1:value1}]})
```

多条件或查询

返回满足任意一个条件的文档

使用$or作为键，使用查询条件列表作为值，列表中的所有查询条件为或的关系

```
db.colname.find({$or:[{key:value},{key1:value1}]})
```

与或混合使用

```
db.colname.find({$or:[{key:value,key1:value1},key2:value2]})
```

6.MongoDB的运算符

6.3 范围运算符

$in运算

键为字段，值为$in为键，列表为值的文档

$in算符在查询中格式

```
db.colname.find({key:{$in:[n1,n2,n3]}})
```

查询key为列表中值的文档，列表不是范围

$nin运算

与$in相反，筛选出某一值不是列表中值的数据，格式同$in

$nin在查询语句中的使用

```
db.colname.find({key:{$nin:[n1,n2,n3]}})
```

6.4 正则表达式

{key:/正则表达式/}

{key:{$regex:'正则表达式'}}

6.MongoDB的运算符

6.4 正则表达式

1.正则表达式的使用范围很广，在mongodb中的使用如下

db.colname.find({key:/正则表达式/})

db.colname.find({key:{$regex:'正则表达式'}})

注:正则表达式只能应用于字符串类型数据

6.MongoDB的运算符

6.5 自定义查询

1.为什么有自定义查询？

有时候查询条件很复杂，前面的查询方法实现起来都不方便，于是我们可以使用自定义查询搞定复杂操作。

2.什么是自定义查询？

自定义查询是使用js语句实现的查询

```
db.colname.find({
        $where:function(){
        return this.key > 3
        }
})
```

1.以这种方式查询,实际上是对find()的结果应用一遍自定义的js的匿名函数,this的作用通Python的self，this指的是每一条文档

2.自定义查询可以定义复杂的判断条件

5.MongoDB的查询操作

5.1 查询结果的后续操作

1.跳过指定条数的查询结果

db.colname.find(query).skip(num)

num默认值为0

2.返回指定条数的查询结果

db.colname.find(query).limit(num)

注:可以和skip进行混合使用，并且没有先后顺序，常用于分页的实现

3.投影操作

db.colname.find(query,投影设定)

指定查询结果返回的字段,设定的格式为字典，键为字段名，值为1或0，为1表示显示，0表示不显示，_id默认为显示状态

5.MongoDB的查询操作

5.1 基本操作

4.排序操作

db.colname.find().sort({key:-1})

设定数据格式为字典格式，键为排序依据的字段，值决定升序还是降序

1表示升序，-1表示降序

5.统计数据操作

1 db.colname.find(query).count()

2 db.colname.count(query)

6.消除重复操作{

db.colname.distinct('字段名',{条件})

7.MongoDB的聚合运算

7.1 为什么要进行聚合运算

传统的查询运算只是将符合条件的数据放到结果集中，而聚合运算则将数据聚合到一起，然后通过管道做后续操作

7.2 如何进行聚合运算

实现聚合运算的方法

db.colname.aggregate([{管道：{表达式}}])

聚合运算是与管道配合使用的，管道是可以有多种的，聚合之后的数据会依次经过管道

7.MongoDB的聚合运算

7.3 常见管道

7.3.1 $group

①$group常被用来进行分组操作

②使用_id来标定分组用的字段，这里的_id与标识每条记录了的唯一的_id并不同

③用来进行分组的字段前面需要加上$,并且用引号引起来

④分组之后的数据还可以进行相关的运算操作

简单的分组

```
db.stu.aggregate([{$group:{_id:”$字段名”}}])
```

分组之后进行运算

```
db.stu.aggregate([{$group:{_id:”$字段名”,result:{$sum:1}}}])
```

①result可以自行定义

②$sum可以替换成其他运算，对应的值为运算结果的倍数

7.MongoDB的聚合运算

7.3 常见管道

7.3.2 $match

①$match实现在聚合中进行查找操作，实现的功能同find()

②与find()操作的区别在与查询结果可以进行后续操作，而find不可以

```
db.colname.aggregate([{$match:{key:{$gt:n}}}])
```

查找key的值大于n的数据，前面的运算符在这里都可以使用

7.3.3 $project

功能类似于查询之后的投影操作

```
db.colname.aggregate(
        {$match:{key:{$gt:n}}},
        {$project:{_id:0}}
)
```

使用match进行比对过滤之后的结果经过project进行字段筛选之后再输出出来

7.MongoDB的聚合运算

7.3 常见管道

7.3.4 $limit

①功能同查询之后的limit操作，返回规定的数据量

```
db.colname.aggregate(
        {$match:{key:{$gt:n}}},
        {$limit:n}
)
```

7.3.5 $skip

跳过指定数量的文档

```
db.colname.aggregate(
        {$match:{key:{$gt:n}}},
        {$skip:n}
)
```

在这里需要注意的是skip与limit的配合使用与find中是不同的,管道是依次执行的，需要根据业务需要进行配置

7.MongoDB的聚合运算

7.3 常见管道

7.3.6 $sort

功能同find的排序操作，对数据进行排序

```
db.colname.aggregate([
        {$match:{key:{$gt:n}}},
        {$sort:{key1:1,key2:-1}}
])
```

7.3.7 $unwind

拆分字段值为列表的字段

```
db.colname.aggregate([
        {$match:{key:{$gt:n}}},
        {$unwind:”$key”}
])
```

不包含该字段的数据

8.MongoDB的索引

8.1 为什么要使用索引

索引可以提升查询速度，提升数据库查询的性能

8.2 索引的优点与缺点:

优点:

提高数据的查询速度

缺点:

牺牲了数据库的插入和更新速度

8.2 如何查看语句执行情况

在执行语句之后追加explain(),并且提供参数executionStats

db.colname.find(query).explain('executionStats')

8.MongoDB的索引

8.3 索引的相关操作

1.查看已存在的索引

db.colname.getIndexes()

2.创建索引

索引创建的通用规则:

①创建索引使用的文档中键为设置索引的字段，值为1表示索引按照升序存储，值为-1则表示索引按照降序存储

②当有大量数据时，创建索引会非常缓慢，因此可以后台创建索引，在创建索引的时候添加{background:true}

单一索引

db.colname.ensureIndex({key:1},{background:true})

联合索引

db.colname.ensureIndex({key1:1,key2:1}

创建联合索引之后，可以使用key1或者key1，key2的查询进行查找

8.MongoDB的索引

8.3 索引的相关操作

2.创建索引

唯一索引

```
db.colname.ensureIndex({‘key’:1},{“unique”:true})
```

注意事项:

①当创建一个key为唯一索引时，新插入的数据如果key的值与已存在的数据相同，则会报错。

②当对已存在数据的集合创建唯一索引时，可能会因为重复，导致创建不成功，使用dropDups可以删除重复文档，但是我们一般不建议使用

```
db.colname.ensureIndex({‘key’:1},{“unique”:true,”dropDups”:true})
```

3.删除索引

```
db.colname.dropIndex({‘key’:1})
```

8.MongoDB的索引

8.4 查询优化器

单索引

当查询的key正好为设置的索引的时候，优化器直接使用索引

复合索引

①当有若干个索引能适合查询用到的key时，优化器会同时并行使用索引进行查询，选择最快索引

②优化器会定期或定查询次数重新进行最有索引的筛选

8.5 索引与全表扫描的对比选择

影响索引效率的属性

索引通常适用的情况	全表扫描通常适用的情况
集合较大	集合较小
文档较大	文档较小
选择性查询	非选择性查询

9.MongoDB的备份与恢复

9.1 数据的备份

9.1.1为什么进行数据备份?

数据备份指的是将数据备份到指定的目录，并在需要的时候进行恢复，一般用于灾难处理

9.1.2 如何进行数据备份?

mongodump -h host -d dbname -o directionary

将指定的数据库备份到指定的目录中

9.1.3 如何进行数据恢复?

mongorestore -h host -d dbname --dir directionary

9.MongoDB的备份与恢复

9.2 数据的导出与导入

9.1.1 为什么进行数据的导出遇导入?

数据库中的数据在与其他平台和应用进行交互时需要按照指定格式导出交给后续步骤进行处理

9.1.2 如何进行数据备份?

导出成json文件：

```
mongoexport -d dbname -c colname -o stu.json
```

导出成csv文件：

```
mongoexport  -d dbname -c colname --type csv -f filed1,filed2,… -o filename.csv
```

.Python与MongoDB的交互

.1 pymongo的简介

.1.1 安装与导入

安装

```
pip install pymongo
```

导入

```
from pymongo import MongoClient
```

.1.2 pymongo中的交互对象

客户端对象	MongoClient链接数据库之后生成的对象
数据库对象	使用客户端对象选择数据库之后生成的对象
集合对象	有数据库对象选择集合之后生成的对象
游标对象	查找之后生成的对象

.Python与MongoDB的交互

.2 pymongo的简单实用

.1.1 在创建链接对象

client= MongoClient(host,port)

.1.2 简单操作

获取数据库列表	client.database_names()
获取数据库链接地址	client.address
关闭数据库	client.close()

.Python与MongoDB的交互

.2 pymongo的简单实用

.1.1 创建数据库对象

db = client.dbname

db = client[dbname]

.1.2 简单操作

获取集合列表	db.collection_names()
创建结合	db.create_collection(colname)
删除集合	db.drop_collection(colname)
查看当前数据库名	db.name
权限认证	db.authenticate(user,pwd)

.Python与MongoDB的交互

.2 pymongo的简单实用

.1.1 创建集合对象

col = db.colname

col = db[colname]

注:对数据的操作主要集中在集合对象

.1.2 简单操作

查看当前集合名

col.name

col.full_name

创建游标对象

col.find()

与shell不同，find之后返回的是游标对象，游标对象可以进行遍历输出

.Python与MongoDB的交互

.3 pymongo的增删改查操作

3.1插入数据

①col.insert_one(data)　　插入单条数据，python字典格式

②col.insert_many(data_list)　　插入多条数据，数据为字典的列表
列表中的每一项都是字典

③ col.insert(data)　　插入单条字典数据或字典列表

3.2查询数据

①col.find(query)　　根据query条件进行查询操作
返回一个游标对象，可迭代输出
query为空，返回所有数据

②col.find_one(query)　　根据条件查询，返回一条数据
数据不存在则返回None

.Python与MongoDB的交互

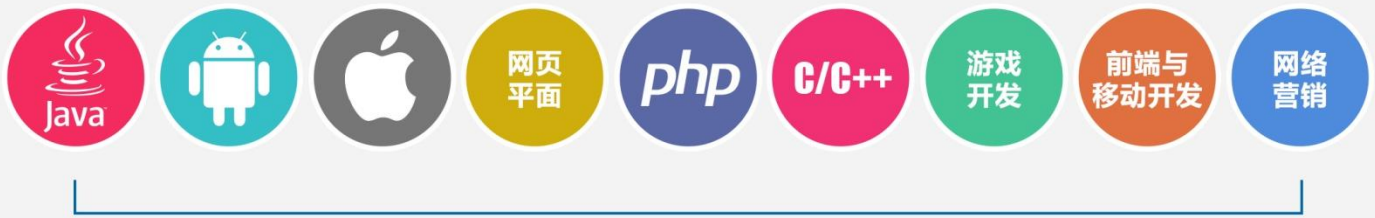
.3 pymongo的增删改查操作

3.3删除数据

col.delete_one(query)	根据条件删除一条数据
col.delete_many(query)	根据条件删除所有符合数据
col.remove(query,multi)	删除数据，根据multi决定数据量

3.4修改数据

col.update_one(query,update,upsert)	修改一条数据 upsert决定是插入数据还是更新数据
col.update_many(query,update,upsert)	修改多条符合条件的数据
col.update(query,update,upsert,multi)	修改更新数据，multi决定修改量



Thank You!

改变中国 IT 教育，我们正在行动

www.itcast.cn