MongDB——Docker 安装与使用

作者：ljxlwyq
原文链接：https://ld246.com/article/1663850778615
来源网站：链滴

MongDB——Docker安装与使用

1、Docker安装MongoDB

```
#拉取镜像
docker pull mongo:4.0.3

#创建容器
docker create --name mongodb -p 27017:27017 -v /data/mongodb:/data/db mongo:4.0.3

#启动容器
docker start mongodb

#进入容器
docker exec -it mongodb /bin/bash

#使用MongoDB客户端进行操作
mongo > show dbs

#查询所有的数据库
admin 0.000GB
config 0.000GB
local 0.000GB
```

2、MongoDB基本概念

● MongoDB是一个基于分布式文件存储的数据库。由C++语言编写。旨在为WEB应用提供可扩展高性能数据存储解决方案。

● MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，

像关系数据库的，它支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的据类型。

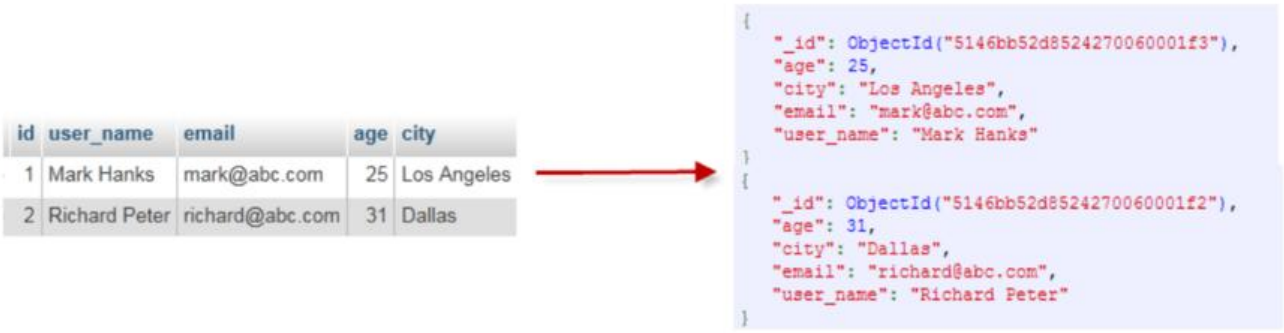
● MongoDB最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

● 官网：https://www.mongodb.com

为了更好的理解，下面与SQL中的概念进行对比：

SQL术语概念	MongoDB术语概念	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	fifield	数据字段/域
index	index	索引
table joins		表连接,MongoDB支持
primary key	primary key	主键MongoDB自动将_id字段设置为主键

mysql和MongoDB的转换



id	user_name	email	age	city
1	Mark Hanks	mark@abc.com	25	Los Angeles
2	Richard Peter	richard@abc.com	31	Dallas

3、数据库以及表的操作

```
#查看所有的数据库
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB

#通过use关键字切换数据库
> use admin
switched to db admin

#创建数据库
#说明：在MongoDB中，数据库是自动创建的，通过use切换到新数据库中，进行插入数据即可自动
```

```
建数据库
> use testdb
switched to db testdb

> show dbs #并没有创建数据库
admin 0.000GB
config 0.000GB
local 0.000GB

> db.user.insert({id:1,name:'zhangsan'}) #插入数据
WriteResult({ "nInserted" : 1 })

> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
testdb 0.000GB

#数据库自动创建
#查看表
> show tables
user
> show collections
user

#删除集合（表）
> db.user.drop()
true
#如果成功删除选定集合，则 drop() 方法返回 true，否则返回 false。

#删除数据库
> use testdb #先切换到要删除的数据中
switched to db testdb

> db.dropDatabase() #删除数据库
{ "dropped" : "testdb", "ok" : 1 }

> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
```

4、新增数据

在MongoDB中，存储的文档结构是一种类似于json的结构，称之为bson（全称为：Binary JSON
。

```
#插入数据
#语法：db.COLLECTION_NAME.insert(document)
> db.user.insert({id:1,username:'zhangsan',age:20})
WriteResult({ "nInserted" : 1 })

> db.user.save({id:2,username:'lisi',age:25})
WriteResult({ "nInserted" : 1 })
```

```
> db.user.find() #查询数据
{ "_id" : ObjectId("5c08c0024b318926e0c1f6dc"), "id" : 1, "username" : "zhangsan", "age" : 20 }

{ "_id" : ObjectId("5c08c0134b318926e0c1f6dd"), "id" : 2, "username" : "lisi", "age" : 25 }
```

5、更新数据

** **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se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，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,就把按条查出来多条记录全部更新。**
- **writeConcern**** :可选，抛出异常的级别。**

```
> db.user.find()
{ "_id" : ObjectId("5c08c0024b318926e0c1f6dc"), "id" : 1, "username" : "zhangsan", "age" : 20 }

{ "_id" : ObjectId("5c08c0134b318926e0c1f6dd"), "id" : 2, "username" : "lisi", "age" : 25 }

> db.user.update({id:1},{$set:{age:22}}) #更新数据
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.user.find()
{ "_id" : ObjectId("5c08c0024b318926e0c1f6dc"), "id" : 1, "username" : "zhangsan", "age" : 22 }

{ "_id" : ObjectId("5c08c0134b318926e0c1f6dd"), "id" : 2, "username" : "lisi", "age" : 25 }

#注意：如果这样写，会删除掉其他的字段
> db.user.update({id:1},{age:25})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.user.find() { "_id" : ObjectId("5c08c0024b318926e0c1f6dc"), "age" : 25 }
{ "_id" : ObjectId("5c08c0134b318926e0c1f6dd"), "id" : 2, "username" : "lisi", "age" : 25 }

#更新不存在的字段，会新增字段
> db.user.update({id:2},{$set:{sex:1}}) #更新数据
```

```
> db.user.find()
{ "_id" : ObjectId("5c08c0024b318926e0c1f6dc"), "age" : 25 }
{ "_id" : ObjectId("5c08c0134b318926e0c1f6dd"), "id" : 2, "username" : "lisi", "age" : 25, "sex" :
 }

#更新不存在的数据，默认不会新增数据
> db.user.update({id:3},{$set:{sex:1}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })

> db.user.find()
{ "_id" : ObjectId("5c08c0024b318926e0c1f6dc"), "age" : 25 }
{ "_id" : ObjectId("5c08c0134b318926e0c1f6dd"), "id" : 2, "username" : "lisi", "age" : 25, "sex" :
 }

#如果设置第一个参数为true，就是新增数据
> db.user.update({id:3},{$set:{sex:1}},true)
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : ObjectId("5c08cb281418
073246bc642") })

> db.user.find()
{ "_id" : ObjectId("5c08c0024b318926e0c1f6dc"), "age" : 25 }
{ "_id" : ObjectId("5c08c0134b318926e0c1f6dd"), "id" : 2, "username" : "lisi", "age" : 25, "sex" :
 }
{ "_id" : ObjectId("5c08cb281418d073246bc642"), "id" : 3, "sex" : 1 }
```

6、删除数据

通过remove()方法进行删除数据，语法如下：

```
db.collection.remove(
   <query>,
   {
     justOne: <boolean>,
     writeConcern: <document>
   }
)
```

参数说明：

- **query**** ：（可选）删除的文档的条件。**
- **justOne**** ：（可选）如果设为 true 或 1，则只删除一个文档，如果不设置该参数，或使用默认值 false，则删除所有匹配条件的文档。**
- **writeConcern**** ：（可选）抛出异常的级别。**

7、查询数据

MongoDB 查询数据的语法格式如下：

```
db.user.find([query],[fields])
```

- **query**** ： 可选，使用查询操作符指定查询条件**
- **fifields**** ： 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

可（默认省略）。**

如果你需要以易读的方式来读取数据，可以使用 pretty() 方法，语法格式如下：

```
>db.col.find().pretty()
```

pretty() 方法以格式化的方式来显示所有文档。

条件查询：

操作	格式	范例	RDBMS的类似语句
等于	{<key>:<value> }	db.col.find({"by":"黑程序员"}).pretty()	where by = '黑马程序员'
小于	{<key>:{$lt:<value>}}	db.col.find({"likes": {$lt:50}}).pretty()	where likes < 50
小于或等于	{<key>:{$lte: <value>}}	db.col.find({"likes": {$lte:50}}).pretty()	where likes <= 50
大于	{<key>:{$gt:<value>}}	db.col.find({"likes": {$gt:50}}).pretty()	where likes > 50
大于或等于	{<key>:{$gte: <value>}}	db.col.find({"likes": {$gte:50}}).pretty()	where likes >= 50
不等于	{<key>:{$ne: <value>}}	db.col.find({"likes": {$ne:50}}).pretty()	where likes != 50

实例：

```
#插入测试数据
db.user.insert({id:1,username:'zhangsan',age:20})
db.user.insert({id:2,username:'lisi',age:21})
db.user.insert({id:3,username:'wangwu',age:22})
db.user.insert({id:4,username:'zhaoliu',age:22})

db.user.find() #查询全部数据
db.user.find({},{id:1,username:1}) #只查询id与username字段
db.user.find().count() #查询数据条数
db.user.find({id:1}) #查询id为1的数据
db.user.find({age:{$lte:21}}) #查询小于等于21的数据
db.user.find({age:{$lte:21}, id:{$gte:2}}) #and查询，age小于等于21并且id大于等于2
db.user.find({$or:[{id:1},{id:2}]}) #查询id=1 or id=2

#分页查询：Skip()跳过几条，limit()查询条数
db.user.find().limit(2).skip(1) #跳过1条数据，查询2条数据

db.user.find().sort({id:-1}) #按照age倒序排序，-1为倒序，1为正序
```

8、索引

● 索引通常能够极大的提高查询的效率，如果没有索引，MongoDB在读取数据时必须扫描集合中的个文件并选取那些符合查询条件的记录。

● 这种扫描全集合的查询效率是非常低的，特别在处理大量的数据时，查询可以要花费几十秒甚至几钟，这对网站的性能是非常致命的。

● 索引是特殊的数据结构，索引存储在一个易于遍历读取的数据集合中，索引是对数据库表中一列或列的值进行排序的一种结构

```
#查看索引 > db.user.getIndexes()
[
    {
        "v" : 2,
        "key" : { "_id" : 1 },
        "name" : "_id_",
        "ns" : "testdb.user"
    }
]
#说明：1表示升序创建索引，-1表示降序创建索引。

#创建索引
> db.user.createIndex({'age':1})
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}

#删除索引
db.user.dropIndex("age_1")
#或者，删除除了_id之外的索引
db.user.dropIndexes()

#创建联合索引
db.user.createIndex({'age':1, 'id':-1})

#查看索引大小，单位：字节
db.user.totalIndexSize()
```

9、执行计划

MongoDB 查询分析可以确保我们建议的索引是否有效，是查询语句性能分析的重要工具。

```
#插入1000条数据
for(var i=1;i<1000;i++)db.user.insert({id:100+i,username:'name_'+i,age:10+i})

#查看执行计划 > db.user.find({age:{$gt:100},id:{$lt:200}}).explain()

#最佳执行计划 "stage" : "FETCH", #查询方式，常见的有COLLSCAN/全表扫描、IXSCAN/索引扫描
 FETCH/根据索引去检索文档、SHARD_MERGE/合并分片结果、IDHACK/针对_id进行查询

#测试没有使用索引 > db.user.find({username:'zhangsan'}).explain()
```