数据库系统概论新技术篇

MongoDB文档数据库

窦志成

中国人民大学信息学院

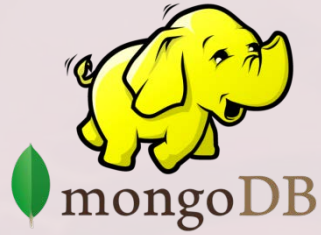
2017年4月

NOSQL回顾

- NoSQL： 打破关系模式限制，提供更为灵活的非关系型的数据存储和管理

列存储数据库 Hbase, Cassandra	键值存储数据库 Redis
图数据库 Neo4J, GraphDB	文档数据库 MongoDB, CouchDB

MongoDB简介



- 是一个开源的高性能无模式文档型数据库
  - 面向文档(document-oriented)：数据库中的每一条记录是一个文档对象，采用类JSON的文档格式，非常接近真实对象模型
  - 无模式(Schema-less)：每个文档格式都可以不同，没有严格的模式定义，灵活方便
  - 高性能：拥有卓越的读写性能，并具有高可用副本集和可扩展分片集群技术，从先天上支持数据库的高扩展性和高伸缩性

MongoDB简介 – 适用场景

适用场景

- 网站数据：MongoDB非常适合实时的插入，更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性
- 缓存：由于性能很高，MongoDB适合作为信息基础设施的持久化缓存层。
- 高伸缩性的集群场景：适合由数台服务器组成的大规模数据存储
- 用于对象及JSON数据的存储：MongoDB的BSON数据格式非常适合文档格式化的存储及查询

不适用场景

- 要求高度事务性的系统：例如银行结算系统，采用传统关系型数据库更适合
- 传统的商业智能应用：BI应用一般要求高度优化的特定查询方式，如上下钻取、切片等，采用数据仓库更适合

MongoDB的使用

提前准备：

- 下载：https://www.mongodb.com/download-center
- 安装：
  - 基本安装https://docs.mongodb.com/manual/installation/
  - 副本配置：https://docs.mongodb.com/manual/replication/
  - 分片配置：https://docs.mongodb.com/manual/sharding/
- 第三方客户端工具
  - Robomongo https://robomongo.org

MongoDB的使用

- MongoDB官方提供了各种不同语言版本的驱动程序和应用开发接口来使用MongoDB，包括C, C++, C#, Java, Node.js, Perl, PHP, Python, Motor, Ruby, 和Scala
- 在开源社区中还有开发人员提供了其他语言版本
- MongoDB官方还提供了原生的MongoDB Shell客户端，可通过Shell进行MongoDB的各种操作

使用MongoDB Shell

- 运行shell客户端: mongo.exe
  - Windows: 通常在你的安装目录下的bin目录中
  - 假设已经通过mongod.exe打开MongoDB服务器

```
D:\ThisMoment\MongoDB\bin>mongo.exe –port 38018
MongoDB shell version: 3.2.4
connecting to: 127.0.0.1:38018/test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
```

使用MongoDB Shell

```
> show dbs
Stock           2358.802GB
StockReport       61.924GB
TestWebPages       0.078GB
UCL                3.952GB
agri              31.938GB
local              0.078GB
test               0.078GB
> use test
switched to db test
> show collections
restaurants
system.indexes
```

查询当前服务器上的所有数据库

切换shell操作的当前数据库
use <database>

查看当前数据库中的表

使用MongoDB Shell

```
> db.restaurants.find({})
{ "_id" : ObjectId("590d63bd59e819a6ae569a64"), "address" :
{ "building" : "1269", "coord" : [ -73.871194, 40.6730975 ],
"street" : "Sutter Avenue", "zipcode" : "11208" }, "borough" :
"Brooklyn", "cuisine" : "Chinese", "grades" : [ { "date" :
ISODate("2014-09-16T00:00:00Z"), "grade" : "B", "score" : 21 },
{ "date" : ISODate("2013-08-28T00:00:00Z"), "grade" : "A",
"score" : 7 }, { "date" : ISODate("2013-04-02T00:00:00Z"),
"grade" : "C", "score" : 56 }, { "date" : ISODate("2012-08-
15T00:00:00Z"), "grade" : "B", "score" : 27 }, { "date" :
ISODate("2012-03-28T00:00:00Z"), "grade" : "B", "score" :
27 } ], "name" : "May May Kitchen", "restaurant_id" :
"40358429" }
{ "_id" : ObjectId("590d63bd59e819a6ae569a65"), "address" :
{ "building" : "203", "coord" : [ -73.97822040000001,
40.6435254 ], "street" : "Church Avenue", "zip code" :
"11218" }, "borough" : "Brooklyn", "cuisine" : "Ice Cream,
Gelato, Yogurt, Ices", "grades" : [ { "date" : ISODate("2014-
02-10T00:00:00Z"), "grade" : "A", "score" : 2 }, { "date" :
ISODate("2013-01-02T00:00:00Z"), "grade" : "A", "score" : 13 },
{ "date" : ISODate("2012-01-09T00:00:00Z"), "grade" : "A",
"score" :
```

查询restaurants集合中的所有文档，使用db.collection.find()

例如使用db.restaurants.find()

或者db.getCollection("restaurants").find()

这两种写法是等价的

```
db.getCollection('restaurants').find({})
```

restaurants 0.006 sec. 0 50

	_id	address	borough	cuisine	grades	name	restaurant_id
1	ObjectId(...	{ 4 fields }	Brooklyn	Chinese	[5 eleme...	May May ...	40358429
2	ObjectId(...	{ 4 fields }	Brooklyn	Ice Crea...	[5 eleme...	Carvel Ice...	40360076
3	ObjectId(...	{ 4 fields }	Manhattan	Irish	[4 eleme...	Dj Reynol...	30191841
4	ObjectId(...	{ 4 fields }	Brooklyn	Delicates...	[6 eleme...	Wilken'S ...	40356483
5	ObjectId(...	{ 4 fields }	Brooklyn	American	[4 eleme...	C & C Cat...	40357437
6	ObjectId(...	{ 4 fields }	Bronx	Bakery	[5 eleme...	Morris Pa...	30075445
7	ObjectId(...	{ 4 fields }	Queens	Ice Crea...	[3 eleme...	Carvel Ice...	40361322
8	ObjectId(...	{ 4 fields }	Manhattan	Chicken	[6 eleme...	Harriet'S ...	40362098
9	ObjectId(...	{ 4 fields }	Queens	Delicates...	[4 eleme...	Sal'S Deli	40361618
10	ObjectId(...	{ 4 fields }	Queens	Delicates...	[4 eleme...	Steve Ch...	40361998
11	ObjectId(...	{ 4 fields }	Manhattan	American	[4 eleme...	Angelika ...	40362274
12	ObjectId(...	{ 4 fields }	Brooklyn	Hamburg...	[4 eleme...	White Ca...	40362344
13	ObjectId(...	{ 4 fields }	Queens	Other	[0 eleme...	Laquana ...	50003441

```
db.getCollection('restaurants').find({})
```

restaurants

	_id	nt_id
1	ObjectId	8429
2	ObjectId	0076
3	ObjectId	1841
4	ObjectId	6483
5	ObjectId	7437
6	ObjectId	5445
7	ObjectId	1322
8	ObjectId	2098
9	ObjectId	1618
10	ObjectId	1998
11	ObjectId	2274
12	ObjectId	2344
13	ObjectId	3441
14	ObjectId	0045
15	ObjectId	2264
16	ObjectId	2715

View Document

localhost:38018 test restaurants


```
{
    "_id" : ObjectId("590d63bd59e819a6ae569a64"),
    "address" : {
        "building" : "1269",
        "coord" : [
            -73.871194,
            40.6730975
        ],
        "street" : "Sutter Avenue",
        "zipcode" : "11208"
    },
    "borough" : "Brooklyn",
    "cuisine" : "Chinese",
    "grades" : [
        {
            "date" : ISODate("2014-09-16T00:00:00.000Z"),
            "grade" : "B",
            "score" : 21
        },
        {
            "date" : ISODate("2013-08-28T00:00:00.000Z"),
            "grade" : "A",
            "score" : 7
        },
```


Cancel

MongoDB简介 – 文档

- 文档由字段/值对构成，展示形式和JSON格式非常相似，底层存储为二进制的BSON格式
- 字段的值可以是基本数据类型（字符串、整数、浮点数、日期等）、数组、也可以是其他的文档或文档数组
- 优点
  - 文档对象格式和某些编程语言（如Javascipt）中的对象格式基本一致，不需要转换，非常方便
  - 嵌套的文档格式避免了跨表Join

```
{
    "_id" : ObjectId("590d63bd59e819a6ae569a64"),
    "name" : "May May Kitchen"
    "address" : {
        "building" : "1269",
        "coord" : [-73.871194, 40.6730975],
        "street" : "Sutter Avenue",
        "zipcode" : "11208"
    },
    "borough" : "Brooklyn",
    "cuisine" : ["Chinese",”Japanese”]
    "grades" : [
        {
            "date" : ISODate("2014-09-16T00:00:00.000Z"),
            "grade" : "B",
            "score" : 21
        },
        {
            "date" : ISODate("2013-08-28T00:00:00.000Z"),
            "grade" : "A",
            "score" : 7
        },
    ]
}
```

使用MongoShell

- 创建集合
  - db.createCollection
- 创建数据库
  - MongoDB没有提供在shell中显示创建数据库的命令，可以简单的使用use <new_database> 然后通过db.createCollection像数据库中添加集合
  - 只有把集合添加到新的数据库后，通过use方式 创建的数据库才会真正保存

```
> use NewRucDB
switched to db NewRucDB
> db.createCollection(“teachers")
{ "ok" : 1 }
> show dbs
NewRucDB          0.078GB
Stock          2358.802GB
StockReport      61.924GB
TestWebPages      0.078GB
UCL               3.952GB
agri             31.938GB
local             0.078GB
test              0.078GB
```

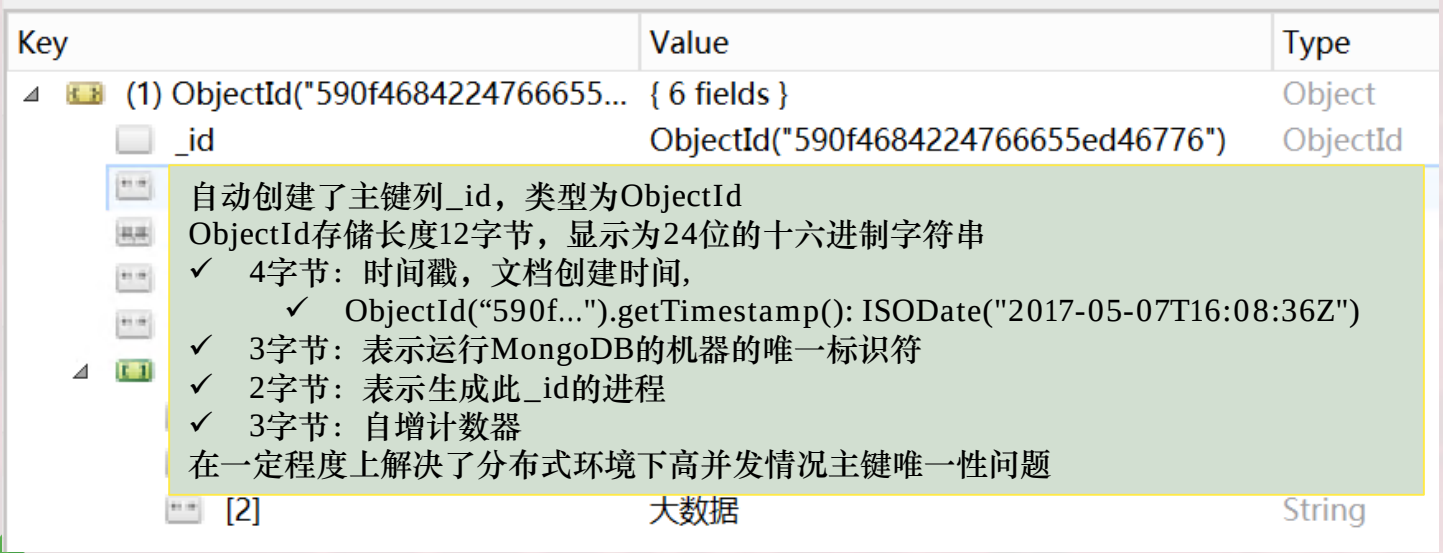
CRUD: Create, Read, Update, Delete

MONGODB 上的基本数据操作

MongoDB CRUD 基本操作

- 插入(create)
  - 统一调用db.collection.insert(<document or array of documents>)将一个或者多个文档插入到集合中
  - 在插入文档前，不需要为Collection指定模式
- 等价格式
  - 插入一个文档：db.collection.insertOne(doc)
  - 同时插入多个文档：db.collection.insertMany([doc1, doc2, ...])

```
db.teachers.insert({    //等价于insertOne
    "name" : "窦志成",
    "age" : 30,
    "gender" : "男",
    "school": "School of Information",
    "area" : ["信息检索", "自然语言处理", "大数据"]
})
```



```
db.teachers.insert([//等价于insertMany
    {
        "_id" : "20130001",
        "name" : "文继荣",
        "age" : 35,
        "gender" : "男",
        "school": "School of Information",
        "area" : [ "信息检索","大数据"],
        "title" : "院长"
    },
    {
        "_id" : "20100001",
        "name" : "杜小勇",
        "age" : 38,
        "gender" : "男",
        "area" : [ "数据库", "数据挖掘"],
    }
])
```

(1)指定了_id
(2)文继荣的文档中多了一个title字段
(3)杜小勇的文档中少了school字段

	_id	name	age	gender	school	area	title
1	ObjectId(...	窦志成	30.0	男	School of...	[3 eleme...	
2	20130001	文继荣	35.0	男	School of...	[2 eleme...	院长
3	20100001	杜小勇	38.0	男		[2 eleme...	

MongoDB CRUD 基本操作

- 查询(Read)
  - db.collection.find()

查询操作	命令	等价的SQL查询
查询全部	db.teachers.find()	SELECT * FROM teachers
等式筛选	db.teachers.find({name:"窦志成"})	SELECT * FROM teachers WHERE name='窦志成'

查询操作符$in	db.teachers.find({
	name: {
	$in: ["窦志成", "文继荣"]
	}
	})
		SELECT * FROM teachers WHERE name in ('窦志成', '文继荣')

```
db.getCollection('teachers').find({
   area:{
      $in:["自然语言处理","数据库"]
   }
 })
```

```
/* 1 */
{
    "_id" : ObjectId("590f4684224766655ed46776"),
    "name" : "窦志成",
    "age" : 30.0,
    "gender" : "男",
    "school" : "School of Information",
    "area" : [
        "信息检索",
        "自然语言处理",
        "大数据"
    ]
}

/* 2 */
{
    "_id" : "20100001",
    "name" : "杜小勇",
    "age" : 38.0,
    "gender" : "男",
    "area" : [
        "数据库",
        "数据挖掘"
    ]
}
```

值得一提的是，如果$in操作符对应的字段是数组，则数据中的任何一个值满足$in条件即可

AND条件和OR条件

```
db.teachers.find( {
   area: { $in: [ "大数据", "信息检索" ] },
   age: { $gt: 30 }
} );
```

```
/* 1 */
{
    "_id" : "20130001",
    "name" : "文继荣",
    "age" : 35.0,
    "gender" : "男",
    "school" : "School of Information",
    "area" : [
        "信息检索",
        "大数据"
    ],
    "title" : "院长"
}
```

```
db.teachers.find( {
   $or:[ { title:"院长" }, { age: { $gt: 35 }} ]
} );
```

```
/* 1 */
{
    "_id" : "20130001",
    "name" : "文继荣",
    "age" : 35.0,
    "gender" : "男",
    "school" : "School of Information",
    "area" : [
        "信息检索",
        "大数据"
    ],
    "title" : "院长"
}

/* 2 */
{
    "_id" : "20100001",
    "name" : "杜小勇",
    "age" : 38.0,
    "gender" : "男",
    "area" : [
        "数据库",
        "数据挖掘"
    ]
}
```

MongoDB CRUD 基本操作

- 修改(Update)
  - db.collection.update()
- 等价操作
  - db.collection.updateOne(): 修改满足条件的第一个文档
  - db.collection.updateMany()：修改满足条件的所有文档
  - db.collection.replaceOne()：替换一个文档

```
db.teachers.update(
    {  "name" : "杜小勇" }, // 筛选条件
    {  $set:{                    //更新操作
          title: "理工处长", school: "School of Information"}
    },
    {                           //更新选项
        "multi" : false,  // 仅更新匹配的第一个文档
    }
);
```

```
{
    "_id" : "20100001",
    "name" : "杜小勇",
    "age" : 38.0,
    "gender" : "男",
    "area" : [
        "数据库",
        "数据挖掘"
    ],
    "title" : "理工处长",
    "school" : "School of Information"
}
```

```
db.teachers.update(
    {  },
    {  $set:{ school:"信息学院" } },
    {  "multi" : true}
);
```

```
db.getCollection('teachers').find({})
```

teachers 0.001 sec. 0 50

	_id	name	age	gender	school	area	title
1	ObjectId(...	窦志成	30.0	男	信息学院	[3 eleme...	
2	20130001	文继荣	35.0	男	信息学院	[2 eleme...	院长
3	20100001	杜小勇	38.0	男	信息学院	[2 eleme...	理工处长

MongoDB CRUD 基本操作

- 删除(delete)
  - db.collection.remove()
- 等价形式
  - db.collection.deleteMany() 删除所有匹配文档
  - db.collection.deleteOne() 删除第一个匹配文档

```
db.teachers.remove(
   { "name": "窦志成"},
   { justOne: true}
);
```

MONGODB的聚合框架

MongoDB提供了一个通用的聚合(aggregation)框架来负责通用的数据处理流程，包括分组汇总统计、排序等。

聚合框架可以用来实现类似于SQL的“GROUP BY”的功能，但它的功能不限于此

MongoDB的聚合框架

- 聚合框架中支持的基本操作
  - $project：修改输入文档的结构。可以用来重命名、增加或删除字段
  - $match：用于过滤数据，只输出符合查询条件的文档
  - $limit：用来限制聚合返回的文档数
  - $skip：在聚合过程中跳过指定数量的文档，并返回余下的文档
  - $unwind：将文档中的某一个数组类型字段拆分成多条，每条包含数组中的一个值
  - $group：将集合中的文档分组，可用于统计结果
  - $sort：将输入文档排序后输出
  - $geoNear：输出接近某一地理位置的有序文档

MongoDB的聚合操作和SQL的关系

SQL	MongoDB
WHERE	$match
GROUP BY	$group
HAVING	$match
SELECT	$project
ORDER BY	$sort
LIMIT	$limit
SUM()	$sum
COUNT()	$sum: 1

```
db.teachers.aggregate( [
   { $match: { age : {$gt:30 }} },
   { $group: {
      _id: "$title",
      numPerson:{$sum:1},
      avgAge : { $avg: "$age" } } },
   { $sort: { numPerson: -1 } }
 ]
)
```

	_id	numPerson	avgAge
1	院长	1.0	35.0
2	理工处长	1.0	38.0

```
db.teachers.aggregate([
   { $unwind: "$area" },
   { $group: {
      _id : "$area",
      numPerson:{ $sum:1 },
      avgAge : { $avg: "$age" } }
   },
   { $sort: { numPerson: -1 } }
])
```

	_id	numPerson	avgAge
1	信息检索	2.0	32.5
2	大数据	2.0	32.5
3	自然语言...	1.0	30.0
4	数据挖掘	1.0	38.0
5	数据库	1.0	38.0

总结

- MongoDB是一个开源的高性能无模式文档型数据库
- 在MongoDB上可以通过非常简单的方式进行数据库、集合以及文档上的各种操作
- 因为时间关系无法赘述，更多详细的介绍请参考MongoDB官方手册：
  https://docs.mongodb.com/manual/