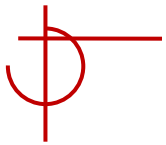
第三章 文档数据库

张元鸣

计算机学院



本章内容

3.1　文档数据库的基本概念

3.2　文档数据库的分区

3.3　文档数据库设计策略

3.4　MongoDB文档数据库

3.5　文档数据库应用开发

3.1 文档数据库基本概念

文档数据库是将数据以文档的形式存储，若干个文档形成一个集合，若干个集合构成了一个独立的文档数据库。

文档数据库	SQL数据库
Database（数据库）	Database
Collection（集合）	Table
Document（文档）	Row
Field（键、字段）	Column
Index（索引）	Index
Primary key（主键）	Primary key

3.1 文档数据库基本概念

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

```
{
    "_id": ObjectId("5146bb52d8524270060001f3"),
    "age": 25,
    "city": "Los Angeles",
    "email": "mark@abc.com",
    "user_name": "Mark Hanks"
}
{
    "_id": ObjectId("5146bb52d8524270060001f2"),
    "age": 31,
    "city": "Dallas",
    "email": "richard@abc.com",
    "user_name": "Richard Peter"
}
```

3.1 文档数据库基本概念

1、文档

文档是由一些键值对构成的有序集合。

– 文档中的成员都只能出现一次，这些成员是键值对，键值对的一般形式是JSON格式。

Json对象的语法规则

1、以{开头，以}结尾，
2、数据要以键值对的形式出现；
3、键值对之间要以逗号分隔；
4、键值对的名称都是字符串，表示某个属性；
5、键值对的值可以是数字、字符串、逻辑值、数组、对象或Null。
6、数组里的各元素值放在[]中；
7、值也可以键值对的形式表示，放在{}中。

3.1 文档数据库基本概念

结构相同的两份文档

```
{
 name:'sue',                    Key→Value
 age:26,                        Key→Value
 stagus:'A',                    Key→Value
 groups:['Sing', 'Sports'],     Key→Value
 Address:{street:'Hangzhou Liuhe road',     Key→Value
          ZipCode:'003210'}
 }
{
 name:'Kitty',
 age:28,
 stagus:'B',
 groups:['Dance', 'Sports'],
 Address:{street:'Hangzhou Zhaohui road',
          ZipCode:'320014'}
 }
```

3.1 文档数据库基本概念

1、文档

文档是由键值对构成的有序集合。

– 文档中的成员是有顺序的。

```
{
‘foo’:’a’,
‘bar’:’b’,
‘baz’:’c’
}
```

两个不同的文档

```
{
‘baz’:’c’,
‘foo’:’a’,
‘bar’:’b’
}
```

3.1 文档数据库基本概念

2、集合

集合包含一组相似的文档。

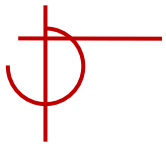
- 集合没有类似SQL数据库中严格的数据库结构（称为模式或纲要），是无纲要的数据库。
  - 纲要，也称模式，是指数据库的结构，如SQL数据库中的表、列、约束等。
  - SQL数据库中，每一行都有同名的键，有对应相同类型的键值，
- 文档数据库开发者无需提前定义模式，可直接创建集合并将文档添加到其中。

3.1 文档数据库基本概念

- 集合内的各个文档可以是各式各样的，非常灵活，根据需要可以随时添加键值对。
  - 由程序员负责检查使用过程中的规则。

```
{
    employeeName:  'Janice Collines',
    department:  'software engineering',
    startDate:  '2016/12/12',
    Projects:  [189,187,176,154]
}
```

```
{
    employeeName:  'Robetr Lucas',
    department:  'Finance',
    startDate:  '2014/7/10",
    certification:  'CAP'
 }
```



3.1 文档数据库基本概念

- 一般地，集合内的文档通常都与同一个主题相关，如产品、学生、课程、事件等。
  - 不同的主题可以存储在同一个集合内，但是显得比较混乱。
- 相同主题的各个文档不需拥有完全相同的结构。
  - 如果有10%的文档需要记录属性A和属性B，那么就不应该强迫另外90%的文档也记录这些内容。

3.1 文档数据库基本概念

- 文档数据库与关系型数据库不同，可以在一份大的文档中嵌入一些小的文档。例如：
  - 在关系数据库中，要用两张表存储学生及其选修课信息。
  - 在文档数据库中，可以在学生文档中嵌入其选修的课程。

3.1 文档数据库基本概念

Students基本表

Sno	Sname	Ssex	Sage	Dno
S01	王建平	男	21	D01
S02	刘华	女	19	D01
S03	范林军	女	18	D02
S04	李伟	男	19	D03
S05	黄河	男	18	D03
S06	长江	男	20	D03

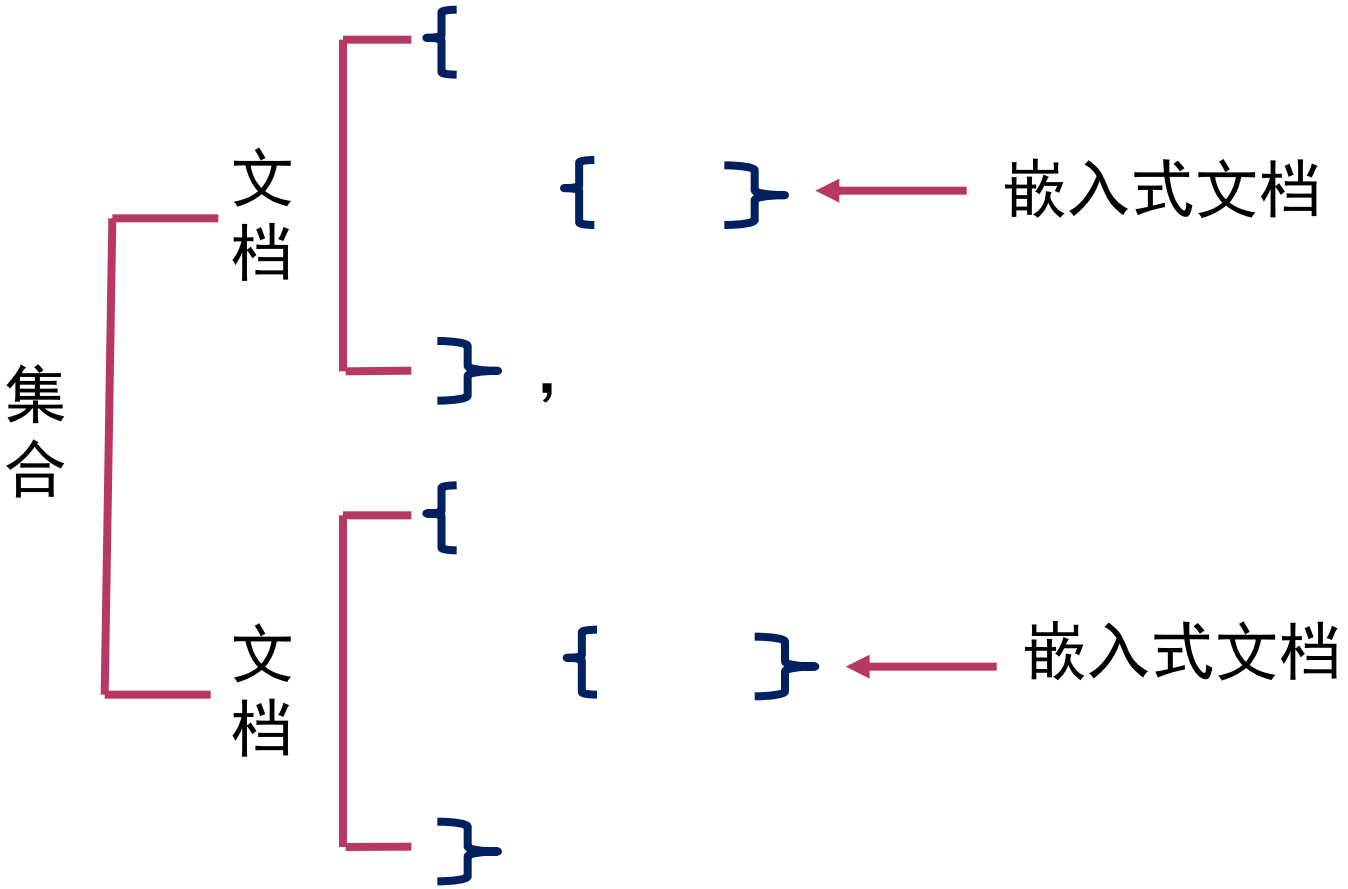
Reports基本表

Sno	Cno	Grade
S01	C01	92
S01	C03	84
S02	C01	90
S02	C02	94
S02	C03	82
S03	C01	72
S03	C02	90
S04	C03	75

```
{
    Sno: “S01” ,
    Sname:”王建平”,
    Sex:”男”,
    Sage:19,
    Dno:“D01”,
    Study:{C01: 92,  C03:84}
},
```

3.1 文档数据库基本概念

文档数据库的基本结构



3.1 文档数据库基本概念

```
{
"UserId": "1001",
"UserName": "张三",
"PassWord": "123456"
}
```

第一个文档

```
{ "UserId": "1002",
"UserName": "李四",
"PassWord": "123456",
 "Detail": { "Address": "湖北", "Age": 20, "Email": "lisi@163.com" }
}
```

第二个文档

```
{ "UserId": "1003",
 "UserName": "赵六",
"PassWord": "123456",
"Detail": { "Address": "湖北" },
}
```

第三个文档

3.1 文档数据库基本概念

3、文档数据库

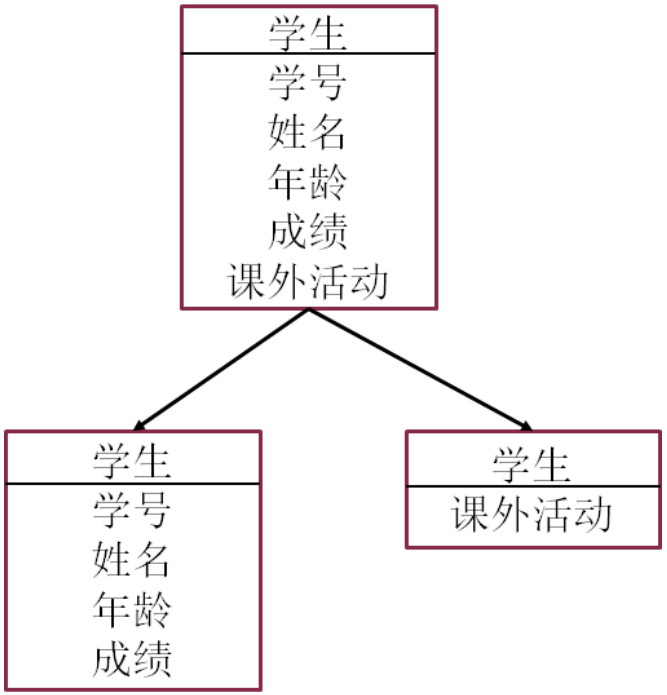
文档数据库是存放集合的容器。

- 一个文档数据库可以存放多个集合。
- 一个集合中可以存放多个文档。
- 文档数据库没有一套类似SQL的标准语言，不同的文档数据库提供了不同的实现。

3.2 文档数据库的分区

指将文档数据库中的数据划分成不同的部分，并把它们分别存储到不同的服务器上。

1）垂直分区：对文档的键值对进行分区，不同的键值对存储到不同的服务器上。



3.2 文档数据库的分区

2）水平分区

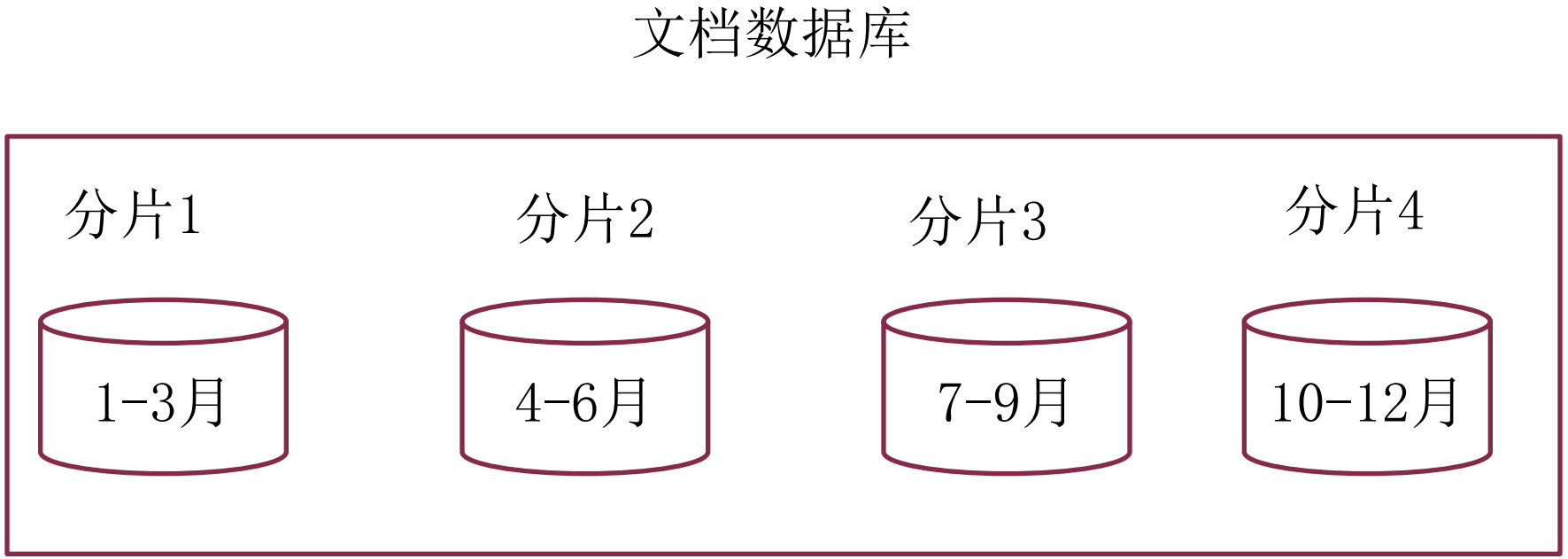
将不同的文档划分在不同的分区上，称为“分片”，这些分片会保存在不同的服务器上。

–优势：提高文档数据库的访问性能，并发度；提高文档数据库的可扩展性。

–用分片键来划分：用某个键去划分文档，如名称、类型、日期、地域等。

–用分区算法来实现数据划分，如按照范围、标准、哈希分区等。

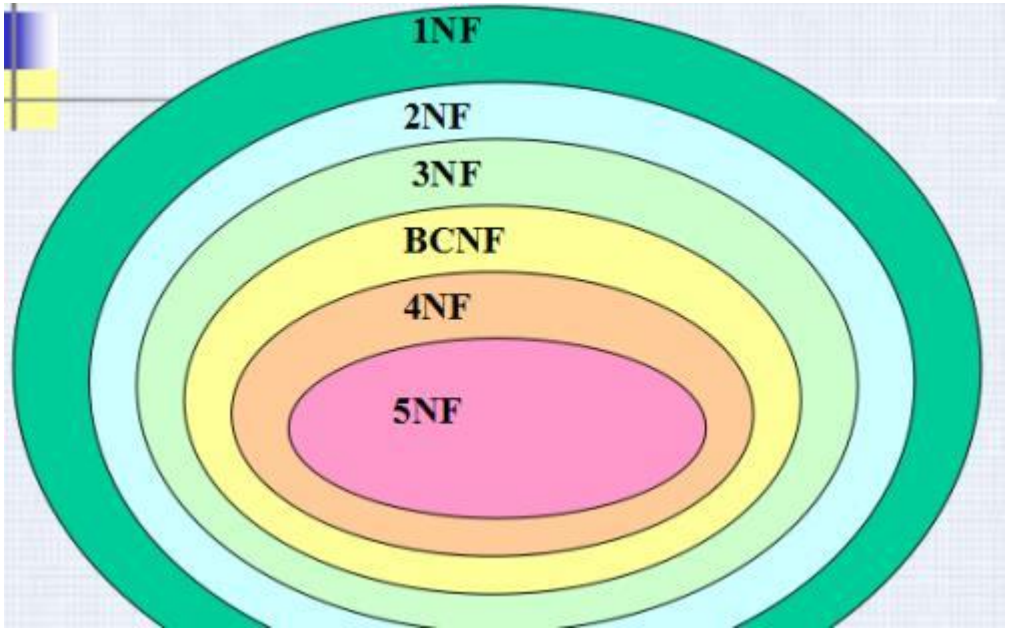
3.2 文档数据库的分区



3.3 文档数据库设计策略

1、文档数据库要适当去规范化

在关系数据库中，规范化的目标是减少数据冗余和更新异常。为此，需要将多个关系模式通过连接操作得到一个完整的信息表格，导致较低的性能问题。



3.3 文档数据库设计策略

1、文档数据库要适当去规范化

在文档数据库中，为了提高性能，尽量少地进行连接操作，把相关的数据都放在同一份文档之内，这个过程称为去规范化。

- 优点：避免连接操作引起的开销，改善查询效率，提高性能。
- 缺点：数据冗余，异常问题。

3.3 文档数据库设计策略

```
{
    order_item_id:83,
    quantity:3,
    cost:9.5,
    product_id:36,
}
```

两个文档放在同一个集合

```
{
    product_id:36,
    product_description:”printer paper”,
    product_name:”printer paper”,
    category:”office supplies”,
    list_price: 9.0
}
```

```
{
  order_item_id:83,
    quantity:3,
    cost:9.5,
    product: {
    product_id:36,
    product_description:”printer paper”,
    product_name:”printer paper”,
    category:”office supplies”,
    list_price: 9.0
  }
}
```

3.3 文档数据库设计策略

2、常见关系建模

多个文档之间在逻辑上存在相互联系，可以是：

- 1:N（1对多）
- N:M（多对多）

此时，文档间可以通过嵌入和引用来建立联系。

3.3 文档数据库设计策略

2、常见关系建模

- 1：n关系
  - 采用嵌套的两个文档表示一对多关系中的两个实体。

```
{
    person_id:319,
    name:”Gang zhang”,
    occupation:”teacher”,
    address: [
          {street:”Zhaohui road”, zip: 310014},
          {street:”Liuhe road”, zip:310032}
         ]
}
```

3.3 文档数据库设计策略

```
{
   "_id":ObjectId("52ffc33cd85242f436000001"),
   "contact": "987654321",
   "dob": "01-01-1991",
   "name": "Tom Benzamin",
   "address": [
      {
         "building": "22 A, Indiana Apt",
         "pincode": 123456,
         "city": "Los Angeles",
         "state": "California"
      },
      {
         "building": "170 A, Acropolis Apt",
         "pincode": 456789,
         "city": "Chicago",
         "state": "Illinois"
      }]
}
```

使用嵌入式方法或引用把用户地址嵌入到文档中

```
{
   "_id":ObjectId("52ffc33cd85242f436000001"),
   "contact": "987654321",
   "dob": "01-01-1991",
   "name": "Tom Benzamin",
   "address_ids": [
      ObjectId("52ffc4a5d85242602e000000"),
      ObjectId("52ffc4a5d85242602e000001")
   ]
}
```

>db.users.findOne({"name":"Tom Benzamin"},{"address":1})

3.3 文档数据库设计策略

2、常见关系建模

– 在文档中给出父节点/子节点的引用

```
{
    {productCategoryID:’PC233’, name:’铅笔’
     parentID:’PC72’},
    {productCategoryID:’PC72’, name:’书写工具’
     parentID:’PC37’},
     {productCategoryID:’PC37’, name:’办公用品’
     parentID:’PC01’},
   {productCategoryID:’PC01’, name:’产品类别’},
}
```

3.3 文档数据库设计策略

2、常见关系建模

- N:M关系
  - 采用两个集合来建模，每个集合表示一种实体，每个集合内的文档维护一份标识符列表。

3.3 文档数据库设计策略

```
{
    course_id: c01,
    name:”maths”,
    credits: 4
    enrolledStudents: [ ‘S01’,’S02’,’S03’]
}
```

```
 {
     StudentID: ‘S01’,
     name:”Zhang Hua”,
     courses: [ ‘C01’,’C02’,’C03’]
 }
```

3.4 MongoDB文档数据库

MongoDB 是一个基于分布式文件存储的数据库，旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。

MongoDB的数据结构非常松散，可以存储比较复杂的数据类型。其查询语言非常强大，几乎可以实现类似关系数据库单表查询的绝大部分功能。

3.4 MongoDB文档数据库

1、管理数据库

1）show dbs：查看所有的数据库。

2）db：显示当前数据库，Mongdb默认的数据库为test。

3）use DATABASE_NAME：如果数据库不存在，则创建数据库，否则切换到指定数据库。

4）db.dropDatabase()：删除当前数据库。

3.4 MongoDB文档数据库

2、管理集合

1）db.createCollection(name, options)：创建集合，类似数据库中的表。

- name: 要创建的集合名称；
- options: 可选参数, 指定有关内存大小及索引的选项；

2）show collections：显示当前数据库中的集合。

3）db.collection.drop()：删除集合。

3.4 MongoDB文档数据库

- use Teacher//切换数据库，注意大小写敏感
- db.createCollection("mycollect")
- db.createCollection(“orders”,{size:1024,capped:true,max:100}) //指定size 1024KB，大小固定，满了就会删除旧文档，最多存放100个文档。
- show collections //显示当前数据库的集合
- db.mycollect.drop() //删除firstcollect集合。

3.4 MongoDB文档数据库

3、管理文档

1）db.COLLECTION_NAME.insert(document):向集合中添加文档，大小写敏感。

```
db.firstcollect.insert(
   {  title: 'MongoDB 教程',
      description: ‘一个 Nosql 数据库’,
      url: 'http://www.runoob.com',
      tags: ['mongodb', 'database', 'NoSQL'],
      likes: 100 }
)
```

3.4 MongoDB文档数据库

3、管理文档

2）db.COLLECTION_NAME.insertOne():向指定集合中插入一条文档数据

- db.firstcollect.insertOne({"a": 3})

3）db.COLLECTION_NAME.insertMany():向指定集合中插入多条文档数据

- db.firstcollect.insertMany([{"b": 3}, {'c': 4}])

3.4 MongoDB文档数据库

3、管理文档

4）db.COLLECTION_NAME.update(<query>,<update>,

{upsert: <boolean>,multi: <boolean>, writeConcern: <document>}

)

- query:更新条件，必填。
- update:更新的对象和一些更新的操作符，必填。
- 更新参数：
  - upsert:如果不存在update的记录，是否插入新文档，true为插入，默认是false，不插入。选填
  - multi: 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。选填
  - writeConcern :可选，抛出异常的级别。选填

3.4 MongoDB文档数据库

- 将cust_id值是A123的文档的amount值修改为510，只修改第一个文档：
  - db.orders.update({cust_id:"A123"},{$set:{amount:510}})
- 将cust_id值是A123的文档的amount值修改为510，更新全部文档：
  - db.orders.update({cust_id:“A123”},{$set:{amount:510}},false,true)
- 修改时，如果没有满足条件的，则插入一个新文档：
  - db.orders.update({cust_id:"A1233"},{$set:{amount:500}},true,false)

3.4 MongoDB文档数据库

常见的更新操作符

① {$set:{field:value}}：把文档中某个字段field的值设为value。
② {$inc:{field:value}}：对一个数字字段的某个field增加value。
③ {$unset:{field:1}}：删除某个字段field。
④ {$push:{field:value}}：把value追加到field数组里，如果field不存在，会自动插入一个数组类型。
⑤ {$pushAll:{field:value_array}}：可以一次追加多个值到一个数组字段内。
⑥ {$addToSet:{field:value}}：加一个值到数组内，而且只有当这个值在数组中不存在时才增加。
⑦ {$pull:{field:_value}}：从数组field内删除一个等于_value的值。
⑧ {$rename:{old_field_name:new_field_name}}：对字段进行重命名

3.4 MongoDB文档数据库

3、管理文档

5）db.COLLECTION_NAME.save(<document}):插入或更新已存在的文档。

- 如果指定_id 字段，则会更新该_id的数据，根据其键值对对原有键值对重新修改。
- 不指定_id 字段，则是插入文档，类似insert（）。

```
>db.col.save({
    “_id” : ObjectId(“56064f89ade2f21f36b03136”),//更新该Key对应的文档
    "title" : "MongoDB",
    "description" : "MongoDB 是一个 Nosql 数据库",
    "by" : "Runoob",
    "url" : "http://www.runoob.com",
    "tags" : [ "mongodb", "NoSQL" ],
    "likes" : 110
})
```

3.4 MongoDB文档数据库

3、管理文档

6）db.COLLECTION_NAME.remove(<query>, <justOne>}) :

移除集合中的文档。

- query: 删除文档的条件，必填。
- justOne：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。

- 删除多个文档

  db.orders.remove({cust_id:"A1233"})
- 删除一个文档

  db.orders.remove({cust_id:“A1233”},1)

3.4 MongoDB文档数据库

3、管理文档

7）db.COLLECTION_NAME.deleteOne(<query>}):删除集合中满足条件的一份文档。

8）db.COLLECTION_NAME.deleteMany(<query>}):删除集合中满足条件的全部多份文档。

- 删除一个文档

  db.orders.deleteOne({amout:500})
- 删除全部文档

  db.orders.deleteMany({amout:500})

3.4 MongoDB文档数据库

4、查询文档

- db.COLLECTION_NAME.find({query}, {projection}): 查询指定集合中的文档，命令、集合名大小写敏感。

  –query：使用查询操作符指定查询条件，必填，但可以是空。类似于SQL中的where子句。

  –projection：可选参数，使用投影操作符指定返回的键。如果返回所有键值，省略该参数。类似于SQL中的Select子句。

3.4 MongoDB文档数据库

```
> db.orders.find({},{_id:0})
{"cust_id" : "A123",   "amount" : 500,   "status" : "A" }
{"cust_id" : "A123",   "amount" : 250,   "status" : "A" }
{"cust_id" : "B212",   "amount" : 200,   "status" : "A" }
{"cust_id" : "A123",   "amount" : 300,   "status" : "D" }
```

3.4 MongoDB文档数据库

- 查询 cust_id 是“A123”的文档

  db.orders.find({cust_id:"A123"})

- 查询cust_id是”A123”或”B123”的档

  db.orders.find({$or:[{cust_id:"A123"},{cust_id:"B123"}]})

- 查询amount大于260的文档

  db.orders.find({amount:{$gt:260}})

3.4 MongoDB文档数据库

MongoDB比较运算符

操作	格式	范例
等于	{<key>:<value>}	db.col.find({"by":"菜鸟教程"}).pretty()
小于	{<key>:{$lt:<value>}}	db.col.find({"likes":{$lt:50}}).pretty()
小于或等于	{<key>:{$lte:<value>}}	db.col.find({"likes":{$lte:50}}).pretty()
大于	{<key>:{$gt:<value>}}	db.col.find({"likes":{$gt:50}}).pretty()
大于或等于	{<key>:{$gte:<value>}}	db.col.find({"likes":{$gte:50}}).pretty()
不等于	{<key>:{$ne:<value>}}	db.col.find({"likes":{$ne:50}}).pretty()

3.4 MongoDB文档数据库

- 查询amount小于260的文档，并只返回cust_id和amout的键值。

```
db.orders.find({amount:{$gt:260}},{_id:0,cust_id:1,amount:1})
```

```
{ "cust_id" : "A123", "amount" : 500 }
{ "cust_id" : "A123", "amount" : 250 }
{ "cust_id" : "B212", "amount" : 200 }
{ "cust_id" : "A123", "amount" : 300 }
```

3.4 MongoDB文档数据库

- 使用projection参数，查询cust_id和amout的键值。

```
db.orders.find({},{_id:0,cust_id:1,amount:1})
{ "cust_id" : "A123", "amount" : 500 }
{ "cust_id" : "A124", "amount" : 250 }
{ "cust_id" : "B212", "amount" : 200 }
{ "cust_id" : “B124", "amount" : 300 }
```

3.4 MongoDB文档数据库

4、查询文档

- db.COLLECTION_NAME.find().pretty().sort().skip().limit():

  –pretty()方法：可选，以格式化的方式来显示所有文档。

  –sort({key: 1/-1}) 方法：对文档按照key排序， 1 为升序排列，-1降序排列。

  –skip()方法 ：可选，跳过指定数量的记录，优先级2。

  –limit()方法：可选，指定读取的记录条数，优先级3。

3.4 MongoDB文档数据库

- 使用limit方法限制文档数量

```
db.orders.find({},{_id:0,cust_id:1,amount:1}).limit(2)
{ "cust_id" : "A123", "amount" : 500 }
{ "cust_id" : "A123", "amount" : 250 }
```

- 使用skip方法跳过文档，跳过第1条，显示二条文档

```
db.orders.find({},{_id:0,cust_id:1,amount:1}).limit(2).skip(1)
{ "cust_id" : "A123", "amount" : 250 }
{ "cust_id" : "B212", "amount" : 200 }
```

3.4 MongoDB文档数据库

- 使用sort方法排序文档， 1 为升序排列，而 -1 降序。

```
db.orders.find({},{_id:0,cust_id:1,amount:1}).limit(2).skip(1).sort({amount:-1})
{ "cust_id" : "A123", "amount" : 300 }
{ "cust_id" : "A123", "amount" : 250 }
```

说明：先跳过1个文档，再限定输出2个文档，再排序。

3.4 MongoDB文档数据库

5、聚合

主要对文档进行过滤、分组、投影、拆分、或者排序等，功能强大。

- db.COLLECTION.aggregate([{pipeline1},{pipeline2},{pipeline3}…]):
  - Pipeline操作: 管道操作的目的在于将上一个管道命令的输出作为下一个管道命令的输入。

3.4 MongoDB文档数据库

聚合框架中常用Pipeline操作

①$match：用于过滤数据，只输出符合条件的文档。

②$group：将集合中的文档分组，可用于统计结果。

③$project：可以用来重命名、增加或删除域，也可以用于创建计算结果以及嵌套文档。

④$limit：用来限制MongoDB聚合返回的文档数。

⑤$skip：在聚合中跳过指定数量的文档，并返回余下的文档。

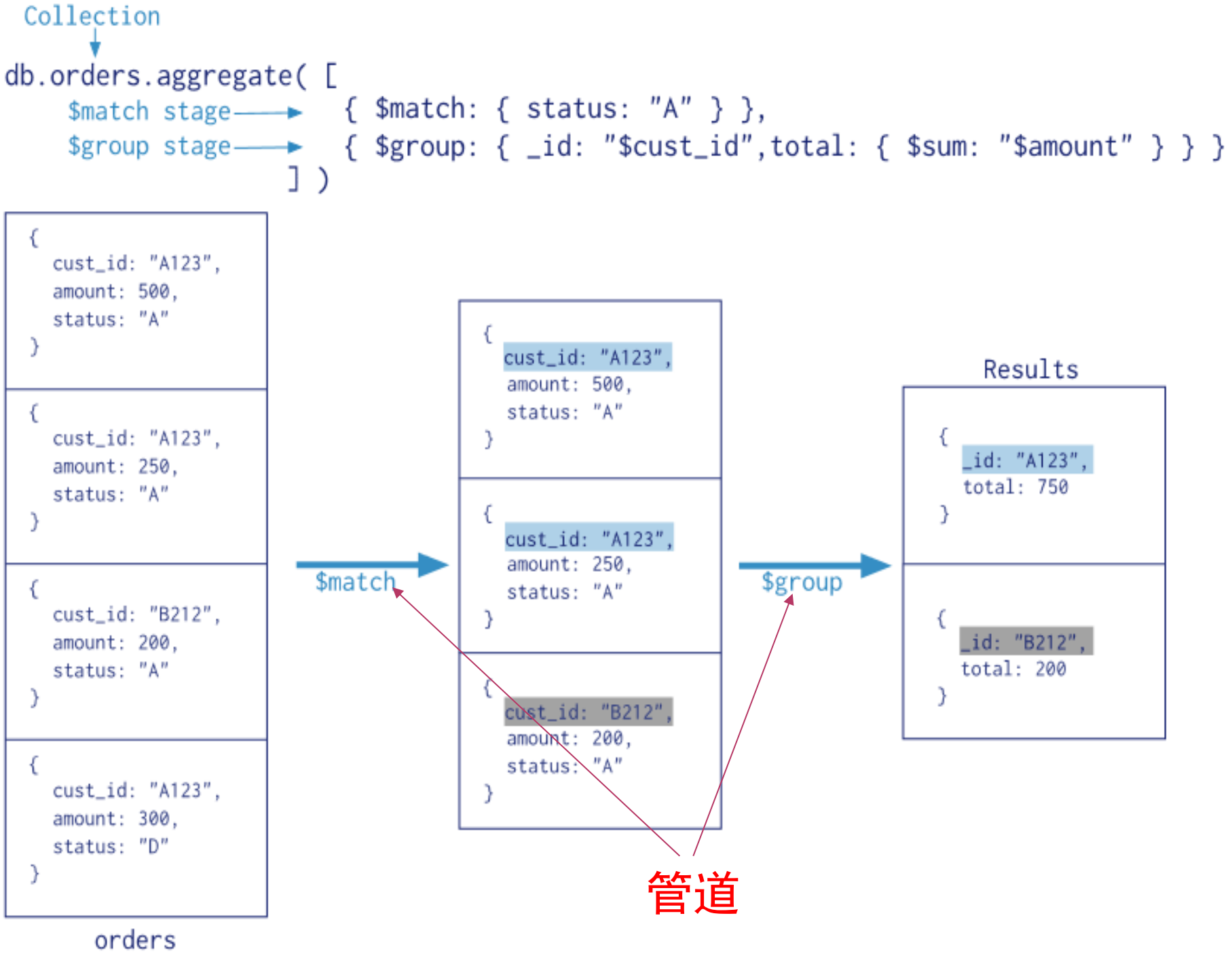
⑥$unwind：将文档中的某一个数组类型字段拆分成多条，每条包含数组中的一个值。

⑦$sort：将输入文档排序后输出。

3.4 MongoDB文档数据库

四个文档实例

```
{"cust_id" : "A123",   "amount" : 500,   "status" : "A" }
{"cust_id" : "A123",   "amount" : 250,   "status" : "A" }
{"cust_id" : "B212",   "amount" : 200,   "status" : "A" }
{"cust_id" : "A123",   "amount" : 300,   "status" : "D" }
```


Collection
db.orders.aggregate([
$match stage
{ $match: { status: "A" } },
$group stage
{ $group: { _id: "$cust_id",total: { $sum: "$amount" } } }
])
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }
orders
$match
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
$group
Results
{ _id: "A123", total: 750 }
{ _id: "B212", total: 200 }
管道

3.4 MongoDB文档数据库

- 管道命令$match：该管道命令用于过滤数据，只输出符合条件的文档。

  db.orders.aggregate([{$match:{status:"A"}}])

  –查询status键值为A的文档。

```
{ "_id" : ObjectId("5dce36dfff577bd66cad6229"), "cust_id" :
    "A123", "amount" : 500, "status" : "A" }
{ "_id" : ObjectId("5dce3773ff577bd66cad622a"), "cust_id" :
    "A123", "amount" : 250, "status" : "A" }
{ "_id" : ObjectId("5dce37b9ff577bd66cad622b"), "cust_id" :
    "B212", "amount" : 200, "status" : "A" }
```

3.4 MongoDB文档数据库

- 管道命令$group：该管道命令将集合中的文档按指定的键进行分组，对每组进行统计。

  db.orders.aggregate([{$group:{_id:"$cust_id",totals:{$sum:"$amount"}}}])

  –对相同主键（_id）下的amount键值进行求和。

  –_id：根据指定的主键进行分组，即按照cust_id进行分组，求每组指定键（amount）值的总和(sum)。

  –total：新设置的键名，代表分组后每组amount的总和。

  –$sum：聚合操作符，求和。

输出结果：
{ "_id" : "A123", "totals" : 1050 }
{ "_id" : "B212", "totals" : 200 }

3.4 MongoDB文档数据库

常用聚合表达式

表达式	描述
$sum	计算总和。
$avg	计算平均值
$min	获取集合中所有文档对应值得最小值。
$max	获取集合中所有文档对应值得最大值。
$push	在结果文档中插入值到一个数组中，值可以重复。
$addToSet	在结果文档中插入值到一个数组中，值不可以重复。
$first	根据资源文档的排序获取第一个文档数据。
$last	根据资源文档的排序获取最后一个文档数据

3.4 MongoDB文档数据库

- 管道命令$group：该管道命令将集合中的文档按指定的键进行分组，对每组进行统计。

```
db.orders.aggregate([{$group:{_id:"null",totals:{$sum:
"$amount"}}}])
```

–_id：id的值必须有，可以是一个null，表示将所有的文档分为一组。

输出结果：
{ "_id" : "null", "totals" : 1250 }

3.4 MongoDB文档数据库

- 联合使用管道命令$match 和$group：查询符合条件的文档，并分组统计。

```
db.orders.aggregate([{$match:{status:"A"}},{$group:{
    _id:"$cust_id",totals:{$sum:"$amount"}}}])
```

–先使用$match命令查询符合条件的文档，再使用$group命令分组统计。

输出结果：

```
{ "_id" : "A123", "totals" : 750 }
{ "_id" : "B212", "totals" : 200 }
```

3.4 MongoDB文档数据库

- 表达式$avg：求平均值。

```
db.orders.aggregate([{$match:{status:"A"}},{$group:{
    _id:"$cust_id",totals:{$avg:"$amount"}}}])
```

输出结果：

```
{ "_id" : "A123", "totals" : 375 }
{ "_id" : "B212", "totals" : 200 }
```

3.4 MongoDB文档数据库

- 表达式$push:将统计值插入数组，去掉重复的值。

```
db.orders.aggregate([{$match:{status:"A"}},{$group:{
_id:"$cust_id",totals:{$push:"$amount"}}}])
```

输出结果：

```
{ "_id" : "A123", "totals" : [ 500, 250 ] }
{ "_id" : "B212", "totals" : [ 200 ] }
```

3.4 MongoDB文档数据库

- 表达式$first:根据资源文档的排序获取第一个文档数据。

```
db.orders.aggregate([{$match:{status:"A"}},{$group:{
    _id:"$cust_id",totals:{$first:"$amount"}}}])
```

输出结果：

```
{ "_id" : "A123", "totals" : 500 }
{ "_id" : "B212", "totals" : 200 }
```

3.4 MongoDB文档数据库

- 管道命令$project :可以用来重命名、增加或删除字段，也可以用于创建计算结果以及嵌套文档。

```
db.orders.aggregate({$project:{_id:0, cust_id:1, amount:1}})
```

输出结果：

```
{ "cust_id" : "A123", "amount" : 500 }
{ "cust_id" : "A123", "amount" : 250 }
{ "cust_id" : "B212", "amount" : 200 }
{ "cust_id" : "A123", "amount" : 300 }
```

3.4 MongoDB文档数据库

- 管道命令$project :与其他管道命令一起用。

```
db.orders.aggregate(
[
{$group: {_id: "$cust_id", totals: {$sum:"$amount"}}},
{$project:{_id:0, newtotals: {$add:["$totals",10]}}}
]
)
```

说明：在之前totals值的基础上增加10 。

输出结果：

{ “newtotals" : 1060 }

{ “newtotals" : 210 }

3.4 MongoDB文档数据库

- 管道命令$limit :用来限制聚合返回的文档数。
  - db.orders.aggregate([{$group:{_id:"$cust_id",totals:{$sum:"$amount"}}}])

    { "_id" : "A123", "totals" : 1050 }

    { "_id" : "B212", "totals" : 200 }
  - db.orders.aggregate([{$group:{_id:"$cust_id",totals:{$sum:"$amount"}}},{$limit:1}])

    { "_id" : "A123", "totals" : 1050 }

3.4 MongoDB文档数据库

- 管道命令$skip :在聚合中跳过指定数量的文档，并返回余下的文档。
  - db.orders.aggregate([{$group:{_id:"$cust_id",totals:{$sum:"$amount"}}}])

    { "_id" : "A123", "totals" : 1050 }

    { "_id" : "B212", "totals" : 200 }
  - db.orders.aggregate([{$group:{_id:"$cust_id",totals:{$sum:"$amount"}}},{$skip:1}])

    { "_id" : "B212", "totals" : 200 }

3.4 MongoDB文档数据库

- 管道命令$sort :将输入文档排序后输出。
  - db.orders.aggregate([{$group:{_id:"$cust_id",totals:{$sum:"$amount"}}},{$sort:{totals:-1}}])

    { "_id" : "A123", "totals" : 1050 }

    { "_id" : "B212", "totals" : 200 }
  - db.orders.aggregate([{$group:{_id:"$cust_id",totals:{$sum:"$amount"}}},{$sort:{totals:1}}])

    { "_id" : "B212", "totals" : 200 }

    { "_id" : "A123", "totals" : 1050 }

3.4 MongoDB文档数据库

- 管道命令$unwind :将文档中的某一个数组类型字段拆分成多条文档，每条包含数组中的一个值。
  - db.wind.aggregate([{$project:{_id:0}}])

```
{ "name" : "chenyao", "address" : "hangzhou", "courses" : [ "C++", "JAVA", "NoSQL" ] }
```

  - db.wind.aggregate([{$project:{_id:0}},{$unwind:"$courses"}])

```
{ "name" : "chenyao", "address" : "hangzhou", "courses" : "C++" }
{ "name" : "chenyao", "address" : "hangzhou", "courses" : "JAVA" }
{ "name" : "chenyao", "address" : "hangzhou", "courses" : "NoSQL" }
```

3.4 MongoDB文档数据库

6、索引

1）db.collection.createIndex({keys：value}, options})

- Key:要创建的索引字段，value值是1按升序创建索引，-1则按降序创建索引。
- Options：根据该参数创建不同类型的索引。

- 在title上按升序创建一个索引

  db.col.createIndex({"title":1})
- 在title和description多个Key上创建复合索引

  db.col.createIndex({"title":1,"description":-1})

3.4 MongoDB文档数据库

创建索引的参数

Parameter	Typ	Description
background	Boolean	建索引过程会阻塞其它数据库操作，background可指定以后台方式创建索引，即增加 "background" 可选参数。 "background" 默认值为**false**。
unique	Boolean	建立的索引是否唯一。指定为true创建唯一索引。默认值为**false**.
name	string	索引的名称。如果未指定，MongoDB的通过连接索引的字段名和排序顺序生成一个索引名称。
sparse	Boolean	对文档中不存在的字段数据不启用索引；这个参数需要特别注意，如果设置为true的话，在索引字段中不会查询出不包含对应字段的文档.。默认值为 **false**.
expireAfterSeconds	integer	指定一个以秒为单位的数值，完成 TTL设定，设定集合的生存时间。
weights	document	索引权重值，数值在 1 到 99,999 之间，表示该索引相对于其他索引字段的得分权重。
default_language	string	对于文本索引，该参数决定了停用词及词干和词器的规则的列表。 默认为英语

3.4 MongoDB文档数据库

- 在后台执行创建索引

  db.orders.createIndex({cust_id:1},{name:"cust_index",background:true}))

- 创建唯一索引

  db.orders.createIndex({cust_id:1, amount: -1}, {unique: true});

- 设置在创建记录后，180 秒后删除。

  db.orders.createIndex({cust_id:1, amount: -1}, {unique: true, expireAfterSeconds: 180})

3.4 MongoDB文档数据库

2）查看集合索引

– db.orders.getIndexes()

3）查看集合索引大小

– db.orders.totalIndexSize()

4）删除集合所有索引

– db.orders.dropIndexes()

5）删除集合指定索引

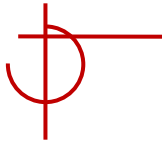
– db.orders.dropIndex(“cust_index”)

3.5 文档数据库应用开发

可以将文档数据库MongoDB嵌入在Java、PHP等程序中，用于开发数据库应用程序，此时需要安装MongoDB JDBC 驱动。

下载地址：

https://mongodb.github.io/mongo-java-driver/



3.5 文档数据库应用开发

```
import com.mongodb.MongoClient;
import com.mongodb.client.MongoDatabase;

public class MongoDBJDBC{
   public static void main( String args[] ){
      try{
       // 连接到 mongodb 服务
         MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

         // 连接到数据库
         MongoDatabase mongoDatabase = mongoClient.getDatabase("mycol");
       System.out.println("Connect to database successfully");

      }catch(Exception e){
        System.err.println( e.getClass().getName() + ": " + e.getMessage() );
     }
   }
}
```

3.6 应用案例

将关系型数据库转换为MongoDB文档数据库

Students基本表

Sno	Sname	Ssex	Sage	Dname
S01	王建平	男	21	自动化
S02	刘华	女	19	自动化
S03	范林军	女	18	计算机
S04	李伟	男	19	数学
S05	黄河	男	18	数学
S06	长江	男	20	数学

3.6 应用案例

Reports基本表

Sno	Cno	Grade
S01	C01	92
S01	C03	84
S02	C01	90
S02	C02	94
S02	C03	82
S03	C01	72
S03	C02	90
S04	C03	75

Courses基本表

Cno	Cname	Pre_Cno	Credits
C01	英语		4
C02	数据结构	C05	2
C03	数据库	C02	2
C04	DB_设计	C03	3
C05	C++		3
C06	网络原理	C07	3
C07	操作系统	C05	3

3.6 应用案例

1、创建Students数据集，并插入所有的文档

- 例1：db.Students.insert ({Sname : "王建平", Sex : "男", Sage : 21, Dname : "自动化", Courses : [{Cno: "C01", Grade : 92 }, {Cno: "C03",Grade:84 }] })
- 例2：db.Students.insert ({Sname: "刘华", Sex: "女", Sage: 19, Dname: "自动化", Courses: [{Cno: "C01", Grade: 90 }, {Cno: "C02", Grade: 94 },{Cno: "C03", Grade: 82 }] })

2、创建Courses数据集，并插入所有的文档

- 例3：db.Courses.insert({Cno:"C01", Cname: "英语", Credits:2, Students:["S01", "S02", "S03"]})

3.6 应用案例

3、完成以下操作：

1) 查询姓名是王建平的文档

2) 查询年龄大于20岁的学生文档

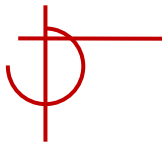
3) 查询选修了C01且及格的学生文档。

例5 db.Students.find({Courses.Cno:"C01",Courses.Grade:{$gte:60}})

4) 在姓名上创建一个升序的唯一索引。

5）查询各门课程的平均分数

例6 db.Students.aggregate([{$unwind:"$Courses"},{$group:{_id:"$Courses.Cno",total:{$sum:"$Courses.Grade"}}}]).pretty()



Chapter over