

前言

MongoDB 是一款开源的文档数据库，并且是业内领先的 NoSQL 数据库，用 C++ 编写而成。

本教程将介绍 MongoDB 的相关概念，从而能够较为深入地了解如何用它来创建并部署具有高度可扩展性，以性 能为导向的数据库。

适用人群

本教程面向的受众是那些想通过简单易学的步骤来学习 MongoDB 数据库的软件专业人员。本教程能帮助你更好 地理解 MongoDB 的相关概念。学完本教程后，你将达到中级水平，继而能够自学更高阶段的内容。

学习前提

在开始学习本教程之前，你需要对数据库、文本编辑器以及程序执行等概念有基本的了解。由于我们将要学习如 何开发高性能的数据库，所以最好能够基本了解数据库（以及 RDBMS）的相关概念。

原文出处：http://www.tutorialspoint.com//mongodb/index.htm

|  |  |
| --- | --- |
| 更新日期 | 更新内容 |
| 2015-06-15 | Mongodb 教程 |

目录

前言. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

第 1 章 教程. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

概述. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5 优势. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7 安装环境. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9 数据模型. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13 创建数据库. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15 删除数据库. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16 创建集合. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17 删除集合. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19 数据类型. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20 插入文档. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21 查询文档. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23 更新文档. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27 删除文档. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29 映射. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31 限制记录. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32 记录排序. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34 索引. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35 聚合. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37 第 2 章 高级教程. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

关系. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52 数据库引用. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54 覆盖索引查询. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56

查询分析. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58 原子操作. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60 高级索引. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62 索引限制. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64 ObjectId. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66 Map Reduce . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68 全文检索. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71 正则表达式. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73 Rockmongo 管理工具. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75 GridFS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77 固定集合. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79 自动增长. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81



|  |  |
| --- | --- |
| |  | | --- | | 1 | |

教程

第 1 章 教程 | 5

概述

MongoDB 是一款跨平台、面向文档的数据库。用它创建的数据库可以实现高性能、高可用性，并且能够轻松扩

展。MongoDB 的运行方式主要基于两个概念：集合（collection）与文档（document）。

数据库

数据库是集合的实际容器。每一数据库都在文件系统中有自己的一组文件。一个 MongoDB 服务器通常有多个数

据库。

集合

集合就是一组 MongoDB 文档。它相当于关系型数据库（RDBMS）中的表这种概念。集合位于单独的一个数据

库中。集合不能执行模式（schema）。一个集合内的多个文档可以有多个不同的字段。一般来说，集合中的文

档都有着相同或相关的目的。

文档

文档就是一组键-值对。文档有着动态的模式，这意味着同一集合内的文档不需要具有同样的字段或结构。

下表展示了关系型数据库与 MongoDB 在术语上的对比：

|  |  |
| --- | --- |
| 关系型数据库 | MongoDB |
| 数据库 | 数据库 |
| 表 | 集合 |
| 行 | 文档 |
| 列 | 字段 |
| 表 Join | 内嵌文档 |
| 主键 | 主键（由 MongoDB 提供的默认 key\_id） |

|  |  |
| --- | --- |
| 数据库服务器 | 客户端 |
| MySQL/Oracle | MongoDB |
| mysql/sqlplus | mongo |

第 1 章 教程 | 6

范例文档

下面这个范例展示了一个简单的博客站点的文档结构，它是由逗号分隔的键值对构成的。

{   
 \_id: ObjectId(7df78ad8902c)   
 title: 'MongoDB Overview',   
 description: 'MongoDB is no sql database',   
 by: 'tutorials point',   
 url: 'http://www.tutorialspoint.com',   
 tags: ['mongodb', 'database', 'NoSQL'],   
 likes: 100,   
 comments: [   
 {   
 user:'user1',   
 message: 'My first comment',   
 dateCreated: new Date(2011,1,20,2,15),   
 like: 0   
 },   
 {   
 user:'user2',   
 message: 'My second comments',   
 dateCreated: new Date(2011,1,25,7,45),   
 like: 5   
 }   
 ]   
}

\_id 是一个 12 字节长的十六进制数，它保证了每一个文档的唯一性。在插入文档时，需要提供\_id 。如果你不

提供，那么 MongoDB 就会为每一文档提供一个唯一的 id。\_id 的头 4 个字节代表的是当前的时间戳，接着的

后 3 个字节表示的是机器 id 号，接着的 2 个字节表示 MongoDB 服务器进程 id，最后的 3 个字节代表递增值。

第 1 章 教程 | 7

优势

任何关系型数据库都采用一种典型的设计模式，展示表的数目以及表之间的关系。然而 MongoDB 却没有关系这 个概念。

MongoDB 相比 RDBMS 的优势

• 模式较少：MongoDB 是一种文档数据库，一个集合可以包含各种不同的文档。每个文档的字段数、内容以 及文档大小都可以各不相同。

|  |  |
| --- | --- |
| • • • | 采用单个对象的模式，清晰简洁。  没有复杂的连接功能。  深度查询功能。MongoDB 支持对文档执行动态查询，使用的是一种不逊色于 SQL 语言的基于文档的查询 |

语言。

|  |  |
| --- | --- |
| • • • • | 具有调优功能。  易于扩展。MongoDB 非常易于扩展。  不需要从应用对象到数据库对象的转换/映射。  使用内部存储存储（窗口化）工作集，能够更快地访问数据。 |

为何选择使用 MongoDB

|  |  |
| --- | --- |
| • • • • • • • | 面向文档的存储：以 JSON 格式的文档保存数据。任何属性都可以建立索引。  复制以及高可扩展性。  自动分片。  丰富的查询功能。  快速的即时更新。  来自 MongoDB 的专业支持。 |

第 1 章 教程 | 8

MongoDB 适用的领域

|  |  |
| --- | --- |
| • • • • • | 大数据  内容管理及交付  移动及社会化基础设施  用户数据管理  数据中心 |

第 1 章 教程 | 9

安装环境

在 Windows 上安装 MongoDB

在 Windows 上安装 MongoDB，先要从http://www.mongodb.org/downloads 上下载 MongoDB 的最新版

本。根据你的 Windows 版本选择正确的 MongoDB 版本。要想知道你的 Windows 版本，在命令行中输入下列

指令：

|  |
| --- |
| C:\>wmic os get osarchitecture |

|  |
| --- |
| OSArchitecture |

|  |
| --- |
| 64-bit |

|  |
| --- |
| C:\> |

32 位版本的 MongoDB 只支持 2G 以下的数据库，只适用于测试及评估。

现在将下载的文件解压至c:\ 或其他位置。解压后的文件夹名称应该是mongodb-win32-i386-[version] 或

mongodb-win32-x86\_64-[version] 。这里的 [version] 代表下载的 MongoDB 版本号。

打开命令行，运行下列命令：

|  |
| --- |
| C:\>move mongodb-win64-\* mongodb |

|  |
| --- |
| 1 dir(s) moved. |

|  |
| --- |
| C:\>  假如将文件解压缩至其他位置，可以采用cd FOOLDER/DIR cd FOOLDER/DIR 找到指定路径，然后运行上面的代码。 |

MongoDB 需要一个 data 文件夹来保存文件。默认的 MongoDB data 目录位于c:\data\db 。所以需要用命令

行来创建这个文件夹。执行下列命令即可：

|  |
| --- |
| C:\>md data |

|  |
| --- |
| C:\md data\db  如果已经把 MongoDB 安装在其他位置，则需要在 mongod.exe 设置 dbpath 路径来指定\data\db \data\db 的替换路 |

径。如下面代码所示。

在命令行中，导航至 bin 目录，进入 MongoDB 安装文件夹。假设我的安装文件夹是：D:\set up\mongodb

|  |
| --- |
| C:\Users\XYZ>d: |

|  |
| --- |
| D:\>cd "set up" |

|  |
| --- |
| D:\set up>cd mongodb |

|  |
| --- |
| D:\set up\mongodb>cd bin |

|  |
| --- |
| D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data" |

第 1 章 教程 | 10

控制台输出会显示waiting for connections waiting for connections 消息，这表示 mongod.exe 进程已经成功运行。

要想运行 mongodb，需要输入下列命令：

|  |
| --- |
| D:\set up\mongodb\bin>mongo.exe |

|  |
| --- |
| MongoDB shell version: 2.4.6 |

|  |
| --- |
| connecting to: test |

|  |
| --- |
| >db.test.save( { a: 1 } ) |

|  |
| --- |
| >db.test.find() |

|  |
| --- |
| { "\_id" : ObjectId(5879b0f65a56a454), "a" : 1 } |

|  |
| --- |
| > |

运行显示 mongodb 已安装并成功运行。下次运行 mongodb 时，只需输入以下命令即可：

|  |
| --- |
| D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data" |

|  |
| --- |
| D:\set up\mongodb\bin>mongo.exe |

在 Ubuntu 上安装 MongoDB

运行下列命令，导入 MongoDB 公开 GPG 键：

|  |
| --- |
| sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10 |

使用下列命令，创建一个 /etc/apt/sources.list.d/mongodb.list 文件。

|  |
| --- |
| echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/source |

|  |
| --- |
| s.list.d/mongodb.list |

运行下列命令，更新存储库：

|  |
| --- |
| sudo apt-get update |

然后利用下列命令安装 MongoDB：

|  |
| --- |
| apt-get install mongodb-10gen=2.2.3 |

在上面的命令中，安装的 2.2.3 版本正是 MongoDB 的当前版本。记住一定要及时更新至最新的版本。至此，M

ongoDB 的安装就成功了。

启动 MongoDB：

|  |
| --- |
| sudo service mongodb start |

停止 MongoDB：

第 1 章 教程 | 11

|  |
| --- |
| sudo service mongodb stop |

重启 MongoDB：

|  |
| --- |
| sudo service mongodb restart |

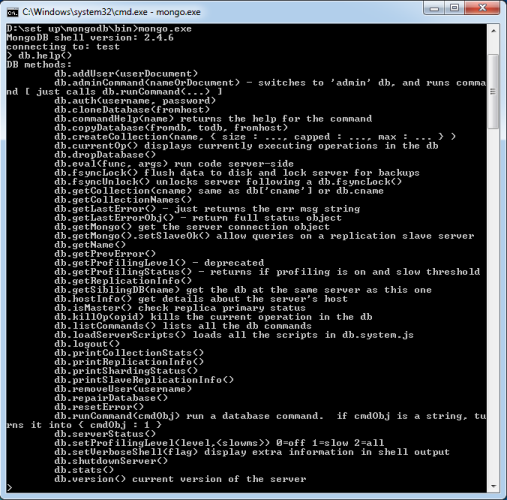
使用 mongodb 时，输入下列命令即可：

|  |
| --- |
| mongo |

这将连接到运行中的 mongod 实例中。

MongoDB 帮助

要想获取命令列表，在 mongodb 客户端中输入db.help() db.help()，将显示如下图所示的命令列表：



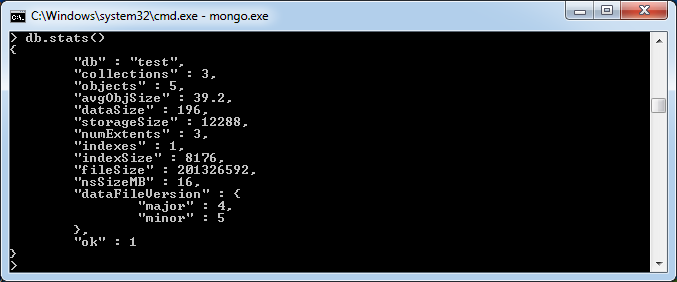
图片 1.1 db\_help

第 1 章 教程 | 12

MongoDB 统计信息

要想获取 MongoDB 服务器的统计信息，在 mongodb 客户端中输入db.stat() db.stat()，随即将显示数据库名称、集合

数目，以及数据库中的文档等信息。如图所示：



图片 1.2 db\_stats

第 1 章 教程 | 13

数据模型

MongoDB 中的数据模式非常灵活。同一集合中的文档不需要具有同一字段或结构，集合文档的公用字段可能包

含不同类型的数据。

设计 MongDB 模式时应注意的问题

|  |  |
| --- | --- |
| • • • • • • | 根据用户需求来设计模式。  如果想一起使用对象，请将这些对象合并到一个文档中，否则要将它们分开（但是要确保不需要连接）。经常复制数据（但要有一定限度），因为与计算时间相比，硬盘空间显得非常便宜。  在写入时进行连接，而不能在读取时连接。  针对经常发生的用例来设计模式。  在模式中实现复杂的聚合。 |

范例

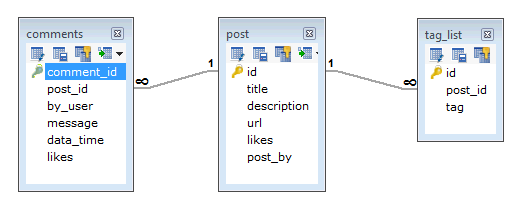
假使一个客户需要为他的博客站点设计一个数据库，让我们来看看 RDBMS 与 MongoDB 在模式设计上的差

异。网站需求如下所示：

|  |  |
| --- | --- |
| • • • • • | 每篇博客都具有唯一的标题、描述以及 URL。  每篇博客都具有一个或多个标签。  每篇博客都具有发表者的名称，以及喜欢  每篇博客都有用户的评论，用户名、消息、日期时间以及评论的喜欢度。每篇博客都可以有 0 个或多个评论。 |

在 RDBMS 中，设计一个能够满足上述需求的数据库模式至少需要 3 个表。如下图所示。

第 1 章 教程 | 14



图片 1.3 rdbms1

在 MongoDB 中，设计出来的模式却只有一个集合 post，其结构如下：

{   
 \_id: POST\_ID   
 title: TITLE\_OF\_POST,   
 description: POST\_DESCRIPTION,   
 by: POST\_BY,   
 url: URL\_OF\_POST,   
 tags: [TAG1, TAG2, TAG3],   
 likes: TOTAL\_LIKES,   
 comments: [   
 {   
 user:'COMMENT\_BY',   
 message: TEXT,   
 dateCreated: DATE\_TIME,   
 like: LIKES   
 },   
 {   
 user:'COMMENT\_BY',   
 message: TEXT,   
 dateCreated: DATE\_TIME,   
 like: LIKES   
 }   
 ]   
}

虽然只是展示数据，在 RDBMS 中需要连接三张表，而在 MongoDB 中则只需要一个集合。

第 1 章 教程 | 15

创建数据库

use 命令

MongoDB 用use + 数据库名称 数据库名称的方式来创建数据库。use 会创建一个新的数据库，如果该数据库存在，则

返回这个数据库。

语法格式

use 语句的基本格式如下：

|  |  |
| --- | --- |
| |  | | --- | | use DATABASE\_NAME | |

范例

创建一个名为的数据库，使用use 语句如下：

|  |
| --- |
| >use mydb |

|  |
| --- |
| switched to db mydb |

使用命令db 检查当前选定的数据库。

|  |
| --- |
| >db |

|  |
| --- |
| mydb |

使用命令show dbs 来检查数据库列表。

|  |
| --- |
| >show dbs |

|  |
| --- |
| local 0.78125GB |

|  |
| --- |
| test 0.23012GB |

刚创建的数据库（mydb）没有出现在列表中。为了让数据库显示出来，至少应该插入一个文档。

|  |
| --- |
| >db.movie.insert({"name":"tutorials point"}) |

|  |
| --- |
| >show dbs |

|  |
| --- |
| local 0.78125GB |

|  |
| --- |
| mydb 0.23012GB |

|  |
| --- |
| test 0.23012GB |

在 MongoDB 中，默认的数据库是 test，如果你没有创建任何数据库，那么集合就会保存在 test 数据库中。

第 1 章 教程 | 16

删除数据库

dropDatabase() 方法

MongoDB 的dropDatabase() 命令用于删除已有数据库。

语法格式

dropDatabase() 命令的语法格式如下：

|  |
| --- |
| db.dropDatabase() |

它将删除选定的数据库。如果没有选定要删除的数据库，则它会将默认的 test 数据库删除。

范例

首先使用show dbs 来列出已有的数据库。

|  |
| --- |
| >show dbs |

|  |
| --- |
| local 0.78125GB |

|  |
| --- |
| mydb 0.23012GB |

|  |
| --- |
| test 0.23012GB |

|  |
| --- |
| > |

如果想删除新数据库<mydb> ，如下面这样使用dropDatabase() 方法：

|  |
| --- |
| >use mydb |

|  |
| --- |
| switched to db mydb |

|  |
| --- |
| >db.dropDatabase() |

|  |
| --- |
| >{ "dropped" : "mydb", "ok" : 1 } |

|  |
| --- |
| > |

再来看一下数据库列表，确实删除了<mydb> 。

|  |
| --- |
| >show dbs |

|  |
| --- |
| local 0.78125GB |

|  |
| --- |
| test 0.23012GB |

|  |
| --- |
| > |

第 1 章 教程 | 17

创建集合

createCollection() 方法

|  |
| --- |
| 在 MongoDB 中，创建集合采用db.createCollection(name, options) db.createCollection(name, options) 方法。 |

语法格式

createCollection() 方法的基本格式如下：

|  |
| --- |
| db.createCollection(name, options) |

在该命令中，name 是所要创建的集合名称。options 是一个用来指定集合配置的文档。

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 描述 |
| name | 字符串 | 所要创建的集合名称 |
| options | 文档 | 可选。指定有关内存大小及索引的选项 |

参数 options 是可选的，所以你必须指定的只有集合名称。下表列出了所有可用选项：

|  |  |  |
| --- | --- | --- |
| 字段 | 类  型 | 描述 |
| capped | 布  尔 | （可选）如果为 true，则创建固定集合。固定集合是指有着固定大小的集合，当达到最大值 时，它会自动覆盖最早的文档。  当该值为 true 时，必须指定 size 参数。 |
| autoInde xID | 布  尔 | （可选）如为 true，自动在 \_id 字段创建索引。默认为 false。 |
| size | 数  值 | （可选）为固定集合指定一个最大值（以字节计）。如果 capped 为 true，也需要指定该字段。 |
| max | 数  值 | （可选）指定固定集合中包含文档的最大数量。 |

在插入文档时，MongoDB 首先检查固定集合的 size 字段，然后检查 max 字段。

范例

不带参数的createCollection() 方法的基本格式为：

|  |
| --- |
| >use test |

|  |
| --- |
| switched to db test |

第 1 章 教程 | 18

|  |
| --- |
| >db.createCollection("mycollection") |

|  |
| --- |
| { "ok" : 1 } |

|  |
| --- |
| > |

可以使用show collections 来查看创建了的集合。

|  |
| --- |
| >show collections |

|  |
| --- |
| mycollection |

|  |
| --- |
| system.indexes |

下面是带有几个关键参数的createCollection() 的用法：

|  |
| --- |
| >db.createCollection("mycol", { capped : true, autoIndexID : true, size : 6142800, max : 10000 } ) |

|  |
| --- |
| { "ok" : 1 } |

|  |
| --- |
| > |

在 MongoDB 中，你不需要创建集合。当你插入一些文档时，MongoDB 会自动创建集合。

|  |
| --- |
| >db.tutorialspoint.insert({"name" : "tutorialspoint"}) |

|  |
| --- |
| >show collections |

|  |
| --- |
| mycol |

|  |
| --- |
| mycollection |

|  |
| --- |
| system.indexes |

|  |
| --- |
| tutorialspoint |

|  |
| --- |
| > |

第 1 章 教程 | 19

删除集合

drop() 方法

MongoDB 利用db.collection.drop() 来删除数据库中的集合。

语法格式

drop() 命令的基本格式如下：

|  |  |
| --- | --- |
| |  | | --- | | db.COLLECTION\_NAME.drop() | |

范例

首先检查在数据库 mydb 中已有集合：

|  |
| --- |
| >use mydb |

|  |
| --- |
| switched to db mydb |

|  |
| --- |
| >show collections |

|  |
| --- |
| mycol |

|  |
| --- |
| mycollection |

|  |
| --- |
| system.indexes |

|  |
| --- |
| tutorialspoint |

|  |
| --- |
| > |

接着删除集合 mycollection。

|  |
| --- |
| >db.mycollection.drop() |

|  |
| --- |
| true |

|  |
| --- |
| > |

再次检查数据库中的现有集合：

|  |
| --- |
| >show collections |

|  |
| --- |
| mycol |

|  |
| --- |
| system.indexes |

|  |
| --- |
| tutorialspoint |

|  |
| --- |
| > |

如果成功删除选定集合，则drop() 方法返回 true，否则返回 false。

第 1 章 教程 | 20

数据类型

MongoDB 支持如下数据类型：

|  |  |
| --- | --- |
| • • • • • • • • • • • | String：字符串。存储数据常用的数据类型。在 MongoDB 中，UTF-8 编码的字符串才是合法的。Integer：整型数值。用于存储数值。根据你所采用的服务器，可分为 32 位或 64 位。  Boolean：布尔值。用于存储布尔值（真/假）。  Double：双精度浮点值。用于存储浮点值。  Min/Max keys：将一个值与 BSON（二进制的 JSON）元素的最低值和最高值相对比。  Arrays：用于将数组或列表或多个值存储为一个键。  Timestamp：时间戳。记录文档修改或添加的具体时间。  Object：用于内嵌文档。  Null：用于创建空值。  Symbol：符号。该数据类型基本上等同于字符串类型，但不同的是，它一般用于采用特殊符号类型的语言。  Date：日期时间。用 UNIX 时间格式来存储当前日期或时间。你可以指定自己的日期时间：创建 Date 对 |

象，传入年月日信息。

|  |  |
| --- | --- |
| • • • • | Object ID：对象 ID。用于创建文档的 ID。  Binary Data：二进制数据。用于存储二进制数据。  Code：代码类型。用于在文档中存储 JavaScript 代码。  Regular expression：正则表达式类型。用于存储正则表达式。 |

第 1 章 教程 | 21

插入文档

insert() 方法

要想将数据插入 MongoDB 集合中，需要使用insert() 或save() 方法。

语法格式

insert() 方法的基本格式为：

|  |  |
| --- | --- |
| |  | | --- | | >db.COLLECTION\_NAME.insert(document) | |

范例 1

|  |
| --- |
| >db.mycol.insert({ |

|  |
| --- |
| \_id: ObjectId(7df78ad8902c), |

|  |
| --- |
| title: 'MongoDB Overview', |

|  |
| --- |
| description: 'MongoDB is no sql database', |

|  |
| --- |
| by: 'tutorials point', |

|  |
| --- |
| url: 'http://www.tutorialspoint.com', |

|  |
| --- |
| tags: ['mongodb', 'database', 'NoSQL'], |

|  |
| --- |
| likes: 100 |

|  |
| --- |
| }) |

mycol 是上一节所创建的集合的名称。如果数据库中不存在该集合，那么 MongoDB 会创建该集合，并向其中插

入文档。

在插入的文档中，如果我们没有指定\_id 参数，那么 MongoDB 会自动为文档指定一个唯一的 ID。

\_id 是一个 12 字节长的 16 进制数，这 12 个字节的分配如下：

|  |  |
| --- | --- |
| |  | | --- | | \_id: ObjectId(4 bytes timestamp, 3 bytes machine id, 2 bytes process id, 3 bytes incrementer) | |

为了，你可以将用insert() 方法传入一个文档数组，范例如下：

范例 2

|  |
| --- |
| >db.post.insert([ |

|  |
| --- |
| { |

第 1 章 教程 | 22

title: 'MongoDB Overview',   
 description: 'MongoDB is no sql database',   
 by: 'tutorials point',   
 url: 'http://www.tutorialspoint.com',   
 tags: ['mongodb', 'database', 'NoSQL'],   
 likes: 100   
},   
{   
 title: 'NoSQL Database',   
 description: 'NoSQL database doesn't have tables',   
 by: 'tutorials point',   
 url: 'http://www.tutorialspoint.com',   
 tags: ['mongodb', 'database', 'NoSQL'],   
 likes: 20,   
 comments: [   
 {   
 user:'user1',   
 message: 'My first comment',   
 dateCreated: new Date(2013,11,10,2,35),   
 like: 0   
 }   
 ]   
}   
])

也可以用db.post.save(document) 插入文档。如果没有指定文档的\_id ，那么save() 就和insert() 完全

一样了。如果指定了文档的\_id ，那么它会覆盖掉含有save() 方法中指定的\_id 的文档的全部数据。

第 1 章 教程 | 23

查询文档

find() 方法

要想查询 MongoDB 集合中的数据，使用find() 方法。

语法格式

find() 方法的基本格式为：

|  |  |
| --- | --- |
| |  | | --- | | >db.COLLECTION\_NAME.find() | |

find() 方法会以非结构化的方式来显示所有文档。

pretty() 方法

用格式化方式显示结果，使用的是pretty() 方法。

语法格式

|  |
| --- |
| >db.mycol.find().pretty() |

范例

|  |
| --- |
| >db.mycol.find().pretty() |

|  |
| --- |
| { |

|  |
| --- |
| "\_id": ObjectId(7df78ad8902c), |

|  |
| --- |
| "title": "MongoDB Overview", |

|  |
| --- |
| "description": "MongoDB is no sql database", |

|  |
| --- |
| "by": "tutorials point", |

|  |
| --- |
| "url": "http://www.tutorialspoint.com", |

|  |
| --- |
| "tags": ["mongodb", "database", "NoSQL"], |

|  |
| --- |
| "likes": "100" |

|  |
| --- |
| } |

|  |
| --- |
| > |

除了find() 方法之外，还有一个findOne() 方法，它只返回一个文档。

第 1 章 教程 | 24

MongoDB 中类似于 WHERE 子句的语句

如果想要基于一些条件来查询文档，可以使用下列操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 操作 | 格式 | 范例 | RDBMS中的类似语句 |
| 等于 | {<key>:<value> } | |  | | --- | | db.mycol.find({"by":"tutorials poi |  |  | | --- | | nt"}).pretty() | | |  | | --- | | where by = 'tutorial |  |  | | --- | | s point' | |
| 小于 | |  | | --- | | {<key>:{$lt:<valu |  |  | | --- | | e>}} | | |  | | --- | | db.mycol.find({"likes":{$lt:50}}).p |  |  | | --- | | retty() | | |  | | --- | | where likes < 50 | |
| 小于或 等于 | |  | | --- | | {<key>:{$lte:<val |  |  | | --- | | ue>}} | | |  | | --- | | db.mycol.find({"likes":{$lte:5 |  |  | | --- | | 0}}).pretty() | | |  | | --- | | where likes <= 50 | |
| 大于 | |  | | --- | | {<key>:{$gt:<val |  |  | | --- | | ue>}} | | |  | | --- | | db.mycol.find({"likes":{$gt:5 |  |  | | --- | | 0}}).pretty() | | |  | | --- | | where likes > 50 | |
| 大于或 等于 | |  | | --- | | {<key>:{$gte:<v |  |  | | --- | | alue>}} | | |  | | --- | | db.mycol.find({"likes":{$gte:5 |  |  | | --- | | 0}}).pretty() | | |  | | --- | | where likes >= 50 | |
| 不等于 | |  | | --- | | {<key>:{$ne:<val |  |  | | --- | | ue>}} | | |  | | --- | | db.mycol.find({"likes":{$ne:5 |  |  | | --- | | 0}}).pretty() | | |  | | --- | | where likes != 50 | |

下例将展示所有由 “tutorials point” 发表的标题为 “MongoDB Overview” 的教程。

|  |
| --- |
| >db.mycol.find({"by":"tutorials point","title": "MongoDB Overview"}).pretty() |

|  |
| --- |
| MongoDB 中的 And 条件  语法格式 |
| 条件的基本语法格式为：  范例  "tags": ["mongodb", "database", "NoSQL"], |

|  |
| --- |
| { |

|  |
| --- |
| "\_id": ObjectId(7df78ad8902c), |

|  |
| --- |
| "title": "MongoDB Overview", |

|  |
| --- |
| "description": "MongoDB is no sql database", |

|  |
| --- |
| "by": "tutorials point", |

|  |
| --- |
| "url": "http://www.tutorialspoint.com", |

|  |
| --- |
| 在find() 方法中，如果传入多个键，并用逗号( , )分隔它们，那么 MongoDB 会把它看成是AND AND 条件。AND |

|  |
| --- |
| >db.mycol.find({key1:value1, key2:value2}).pretty() |

第 1 章 教程 | 25

|  |
| --- |
| "likes": "100" |

|  |
| --- |
| } |

|  |
| --- |
| > |

对于上例这种情况，RDBMS 采用的 WHERE 子句将会是：where by='tutorials point' AND title='MongoDB

Overview'。你可以在 find 子句中传入任意的键值对。

MongoDB 中的 OR 条件

语法格式

若基于OR OR 条件来查询文档，可以使用关键字$or $or。 OR 条件的基本语法格式为：

|  |
| --- |
| >db.mycol.find( |

|  |
| --- |
| { |

|  |
| --- |
| $or: [ |

|  |
| --- |
| {key1: value1}, {key2:value2} |

|  |
| --- |
| ] |

|  |
| --- |
| } |

|  |
| --- |
| ).pretty() |

范例

下例将展示所有由 “tutorials point” 发表的标题为 “MongoDB Overview” 的教程。

|  |
| --- |
| >db.mycol.find({$or:[{"by":"tutorials point"},{"title": "MongoDB Overview"}]}).pretty() |

|  |
| --- |
| { |

|  |
| --- |
| "\_id": ObjectId(7df78ad8902c), |

|  |
| --- |
| "title": "MongoDB Overview", |

|  |
| --- |
| "description": "MongoDB is no sql database", |

|  |
| --- |
| "by": "tutorials point", |

|  |
| --- |
| "url": "http://www.tutorialspoint.com", |

|  |
| --- |
| "tags": ["mongodb", "database", "NoSQL"], |

|  |
| --- |
| "likes": "100" |

|  |
| --- |
| } |

|  |
| --- |
| > |

第 1 章 教程 | 26

结合使用 AND 与 OR 条件

范例

下例所展示文档的条件为：喜欢数大于 100，标题是 “MongoDB Overview”，或者是由 “tutorials point”

所发表的。响应的 SQL WHERE 子句为：where likes>10 AND (by = 'tutorials point' OR title = 'MongoDB

Overview')

|  |
| --- |
| >db.mycol.find({"likes": {$gt:10}, $or: [{"by": "tutorials point"},{"title": "MongoDB Overview"}]}).pretty() |

|  |
| --- |
| { |

|  |
| --- |
| "\_id": ObjectId(7df78ad8902c), |

|  |
| --- |
| "title": "MongoDB Overview", |

|  |
| --- |
| "description": "MongoDB is no sql database", |

|  |
| --- |
| "by": "tutorials point", |

|  |
| --- |
| "url": "http://www.tutorialspoint.com", |

|  |
| --- |
| "tags": ["mongodb", "database", "NoSQL"], |

|  |
| --- |
| "likes": "100" |

|  |
| --- |
| } |

|  |
| --- |
| > |

第 1 章 教程 | 27

更新文档

MongoDB 中的update() 与save() 方法都能用于更新集合中的文档。update() 方法更新已有文档中的

值，而save() 方法则是用传入该方法的文档来替换已有文档。

update() 方法

update() 方法更新已有文档中的值。

语法格式

update() 方法基本格式如下：

|  |
| --- |
| >db.COLLECTION\_NAME.update(SELECTIOIN\_CRITERIA, UPDATED\_DATA) |

范例

假如 mycol 集合中有下列数据：

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} |

下面的例子将把文档原标题 'MongoDB Overview' 替换为新的标题 'New MongoDB Tutorial'。

|  |
| --- |
| >db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}}) |

|  |
| --- |
| >db.mycol.find() |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} |

|  |
| --- |
| > |

MongoDB 默认只更新单个文档，要想更新多个文档，需要把参数multi 设为true 。

|  |  |
| --- | --- |
| |  | | --- | | >db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}},{multi:true}) | |

第 1 章 教程 | 28

save() 方法

save() 方法利用传入该方法的文档来替换已有文档。

语法格式

save() 方法基本语法格式如下：

|  |  |
| --- | --- |
| |  | | --- | | >db.COLLECTION\_NAME.save({\_id:ObjectId(),NEW\_DATA}) | |

范例

下例用\_id 为 '5983548781331adf45ec7' 的文档代替原有文档。

|  |
| --- |
| >db.mycol.save( |

|  |
| --- |
| { |

|  |
| --- |
| "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point New Topic", "by":"Tutorials Point" |

|  |
| --- |
| } |

|  |
| --- |
| ) |

|  |
| --- |
| >db.mycol.find() |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec5), "title":"Tutorials Point New Topic", "by":"Tutorials Point"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} |

|  |
| --- |
| > |

第 1 章 教程 | 29

删除文档

remove() 方法

MongoDB 利用remove() 方法 清除集合中的文档。它有 2 个可选参数：

|  |  |
| --- | --- |
| • • | deletion criteria：（可选）删除文档的标准。  justOne：（可选）如果设为 true 或 1，则只删除一个文档。 |

语法格式

remove() 方法的基本语法格式如下所示：

|  |
| --- |
| >db.COLLECTION\_NAME.remove(DELLETION\_CRITTERIA) |

范例

假如 mycol 集合中包含下列数据：

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} |

下面我们将删除其中所有标题为 'MongoDB Overview' 的文档。

|  |
| --- |
| >db.mycol.remove({'title':'MongoDB Overview'}) |

|  |
| --- |
| >db.mycol.find() |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} |

|  |
| --- |
| > |

只删除一个文档

如果有多个记录，而你只想删除第一条记录，那么就设置remove() 方法中的justOne 参数：

|  |  |
| --- | --- |
| |  | | --- | | >db.COLLECTION\_NAME.remove(DELETION\_CRITERIA,1) | |

第 1 章 教程 | 30

删除所有文档

如果没有指定删除标准，则 MongoDB 会将集合中所有文档都予以删除。这等同于 SQL 中的 truncate 命令

|  |
| --- |
| >db.mycol.remove() |

|  |
| --- |
| >db.mycol.find() |

|  |
| --- |
| > |

第 1 章 教程 | 31

映射

在 MongoDB 中，映射（Projection）指的是只选择文档中的必要数据，而非全部数据。如果文档有 5 个字

段，而你只需要显示 3 个，则只需选择 3 个字段即可。

find() 方法

MongoDB 的查询文档曾介绍过find() 方法，它可以利用 AND 或 OR 条件来获取想要的字段列表。在 Mongo

DB 中执行find() 方法时，显示的是一个文档的所有字段。要想限制，可以利用 0 或 1 来设置字段列表。1 用于

显示字段，0 用于隐藏字段。

语法格式

带有映射的find() 方法的基本语法格式为：

|  |
| --- |
| >db.COLLECTION\_NAME.find({},{KEY:1}) |

范例

假如 mycol 集合拥有下列数据：

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} |

下例将在查询文档时显示文档标题。

|  |
| --- |
| >db.mycol.find({},{"title":1,\_id:0}) |

|  |
| --- |
| {"title":"MongoDB Overview"} |

|  |
| --- |
| {"title":"NoSQL Overview"} |

|  |
| --- |
| {"title":"Tutorials Point Overview"} |

|  |
| --- |
| > |

注意：在执行find() 方法时，\_id 字段是一直显示的。如果不想显示该字段，则可以将其设为 0。

第 1 章 教程 | 32

限制记录

limit() 方法

要想限制 MongoDB 中的记录，可以使用limit() 方法。limit() 方法接受一个数值类型的参数，其值为想要显

示的文档数。

语法格式

limit() 方法的基本语法格式为：

|  |  |
| --- | --- |
| |  | | --- | | >db.COLLECTION\_NAME.find().limit(NUMBER) | |

范例

假设 mycol 集合拥有下列数据：

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} |

下例将在查询文档时只显示 2 个文档。

|  |
| --- |
| >db.mycol.find({},{"title":1,\_id:0}).limit(2) |

|  |
| --- |
| {"title":"MongoDB Overview"} |

|  |
| --- |
| {"title":"NoSQL Overview"} |

|  |
| --- |
| > |

如果未指定limit() 方法中的数值参数，则将显示该集合内的所有文档。

skip() 方法

语法格式

skip() 方法基本语法格式为：

|  |
| --- |
| >db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER) |

第 1 章 教程 | 33

范例

下例将只显示第二个文档：

|  |
| --- |
| >db.mycol.find({},{"title":1,\_id:0}).limit(1).skip(1) |

|  |
| --- |
| {"title":"NoSQL Overview"} |

|  |
| --- |
| > |

注意：skip() 方法中的默认值为 0。

第 1 章 教程 | 34

记录排序

sort() 方法

MongoDB 中的文档排序是通过sort() 方法来实现的。sort() 方法可以通过一些参数来指定要进行排序的字

段，并使用 1 和 -1 来指定排序方式，其中 1 表示升序，而 -1 表示降序。

格式

sort() 方法基本格式为：

|  |  |
| --- | --- |
| |  | | --- | | >db.COLLECTION\_NAME.find().sort({KEY:1}) | |

范例

假设集合 myycol 包含下列数据：

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"} |

|  |
| --- |
| { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} |

下面的范例将显示按照降序排列标题的文档。

|  |
| --- |
| >db.mycol.find({},{"title":1,\_id:0}).sort({"title":-1}) |

|  |
| --- |
| {"title":"Tutorials Point Overview"} |

|  |
| --- |
| {"title":"NoSQL Overview"} |

|  |
| --- |
| {"title":"MongoDB Overview"} |

|  |
| --- |
| > |

注意，如果不指定排序规则，sort() 方法将按照升序排列显示文档。

第 1 章 教程 | 35

索引

索引能够实现高效地查询。没有索引，MongoDB 就必须扫描集合中的所有文档，才能找到匹配查询语句的文

档。这种扫描毫无效率可言，需要处理大量的数据。

索引是一种特殊的数据结构，将一小块数据集保存为容易遍历的形式。索引能够存储某种特殊字段或字段集的

值，并按照索引指定的方式将字段值进行排序。

ensureIndex() 方法

要想创建索引，需要使用 MongoDB 的ensureIndex() 方法。

语法格式

ensureIndex() 方法的基本语法格式为：

|  |
| --- |
| >db.COLLECTION\_NAME.ensureIndex({KEY:1}) |

这里的 key 是想创建索引的字段名称，1 代表按升序排列字段值。-1 代表按降序排列。

范例

|  |
| --- |
| >db.mycol.ensureIndex({"title":1}) |

|  |
| --- |
| > |

可以为ensureIndex() 方法传入多个字段，从而为多个字段创建索引。

|  |
| --- |
| >db.mycol.ensureIndex({"title":1,"description":-1}) |

|  |
| --- |
| > |

ensureIndex() 方法也可以接受一些可选参数，如下所示：

|  |  |  |
| --- | --- | --- |
| 参数 | 类  型 | 描述 |
| |  | | --- | | backgroun |  |  | | --- | | d | | 布尔 值 | 在后台构建索引，从而不干扰数据库的其他活动。取值为true 时，代表在后台构建 索引。默认值为false |
| |  | | --- | | unique | | 布尔 值 | 创建一个唯一的索引，从而当索引键匹配了索引中一个已存在值时，集合不接受文档的 插入。取值为true 代表创建唯一性索引。默认值为false 。 |

第 1 章 教程 | 36

|  |  |  |
| --- | --- | --- |
| 参数 | 类  型 | 描述 |
| |  | | --- | | name | | 字符 串 | 索引名称。如果未指定，MongoDB 会结合索引字段名称和排序序号，生成一个索引名 称。 |
| |  | | --- | | dropDups | | 布尔 值 | 在可能有重复的字段内创建唯一性索引。MongoDB 只在某个键第一次出现时进行索 引，去除该键后续出现时的所有文档。 |
| |  | | --- | | sparse | | 布尔 值 | 如果为 true，索引只引用带有指定字段的文档。这些索引占据的空间较小，但在一些情 况下的表现也不同（特别是排序）。默认值为false 。 |
| |  | | --- | | expireAfter |  |  | | --- | | Seconds | | 整型 值 | 指定一个秒数值，作为 TTL 来控制 MongoDB 保持集合中文档的时间。 |
| |  | | --- | | v | | 索引 版本 | 索引版本号。默认的索引版本跟创建索引时运行的 MongoDB 版本号有关。 |
| |  | | --- | | weights | | 文档 | 文档数值，范围从 1 到 99, 999。表示就字段相对于其他索引字段的重要性。 |
| |  | | --- | | default\_lan |  |  | | --- | | guage | | 字符 串 | 对文本索引而言，用于确定停止词列表，以及词干分析器（stemmer）与断词器（tok enizer）的规则。默认值为 english。 |
| |  | | --- | | languag |  |  | | --- | | e\_override | | 字符 串 | 对文本索引而言，指定了文档所包含的字段名，该语言将覆盖默认语言。默认值为 lang uage。 |

第 1 章 教程 | 37

聚合

聚合操作能够处理数据记录并返回计算结果。聚合操作能将多个文档中的值组合起来，对成组数据执行各种操

作，返回单一的结果。它相当于 SQL 中的 count(\*) 组合 group by。

aggregate() 方法

对于 MongoDB 中的聚合操作，应该使用aggregate() 方法。

语法格式

aggregate() 方法中的基本格式如下所示：

|  |
| --- |
| >db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION) |

范例

假如某个集合包含下列数据：

{   
 \_id: ObjectId(7df78ad8902c)   
 title: 'MongoDB Overview',   
 description: 'MongoDB is no sql database',   
 by\_user: 'tutorials point',   
 url: 'http://www.tutorialspoint.com',   
 tags: ['mongodb', 'database', 'NoSQL'],   
 likes: 100   
},   
{   
 \_id: ObjectId(7df78ad8902d)   
 title: 'NoSQL Overview',   
 description: 'No sql database is very fast',   
 by\_user: 'tutorials point',   
 url: 'http://www.tutorialspoint.com',   
 tags: ['mongodb', 'database', 'NoSQL'],   
 likes: 10   
},   
{   
 \_id: ObjectId(7df78ad8902e)

第 1 章 教程 | 38

|  |
| --- |
| title: 'Neo4j Overview', |

|  |
| --- |
| description: 'Neo4j is no sql database', |

|  |
| --- |
| by\_user: 'Neo4j', |

|  |
| --- |
| url: 'http://www.neo4j.com', |

|  |
| --- |
| tags: ['neo4j', 'database', 'NoSQL'], |

|  |
| --- |
| likes: 750 |

|  |
| --- |
| }, |

假如想从上述集合中，归纳出一个列表，以显示每个用户写的教程数量，需要像下面这样使用aggregate() 方

法：

> db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : 1}}}]) {   
 "result" : [   
 {   
 "\_id" : "tutorials point",   
 "num\_tutorial" : 2   
 },   
 {   
 "\_id" : "Neo4j",   
 "num\_tutorial" : 1   
 }   
 ],   
 "ok" : 1   
}   
>   
```

|  |
| --- |
| 假如用 SQL 来处理上述查询，则需要使用这样的命令：`select by\_user, count(\*) from mycol group by by\_user`。 |

|  |
| --- |
| 上例使用 \*\*by\_user\*\* 字段来组合文档，每遇到一次 `by\_user`，就递增之前的合计值。下面是聚合表达式列表。 |

|表达式|描述|范例|   
|---|---|---|   
|`$sum`|对集合中所有文档的定义值进行加和操作|`db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : "$lik |`$avg`|对集合中所有文档的定义值进行平均值|`db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$avg : "$likes |`$min`|计算集合中所有文档的对应值中的最小值|`db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$min : "$like |`$max`|计算集合中所有文档的对应值中的最大值|`db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$max : "$lik |`$push`|将值插入到一个结果文档的数组中|`db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$push: "$url"}}}])`| |`$addToSet`|将值插入到一个结果文档的数组中，但不进行复制|`db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$addTo |`$first`|根据成组方式，从源文档中获取第一个文档。但只有对之前应用过 `$sort` 管道操作符的结果才有意义。|`db.mycol.aggregat |`$last`|根据成组方式，从源文档中获取最后一个文档。但只有对之前进行过 `$sort` 管道操作符的结果才有意义。|`db.mycol.aggreg

### 管道的概念

第 1 章 教程 | 39

在 UNIX 命令 Shell 中，管道（pipeline）概念指的是能够在一些输入上执行一个操作，然后将输出结果用作下一个命令的输入。Mong

聚合架构中可能采取的管道操作符有：

- \*\*$project\*\* 用来选取集合中一些特定字段。

- \*\*$match\*\* 过滤操作。减少用作下一阶段输入的文档的数量。

- \*\*$group\*\* 如上所述，执行真正的聚合操作。

- \*\*$sort\*\* 对文档进行排序。

- \*\*$skip\*\* 在一组文档中，跳过指定数量的文档。

- \*\*$limit\*\* 将查看文档的数目限制为从当前位置处开始的指定数目。

- \*\*$unwind\*\* 解开使用数组的文档。当使用数组时，数据处于预连接状态，通过该操作，数据重新回归为各个单独的文档的状态。利

## 复制

复制是一种在多个服务器上同步数据的过程。通过在不同的数据库服务器上实现多个数据副本，复制能够实现数据冗余，提高数据的可用

### 为什么需要复制

- 保持数据安全  
- 保证数据的高可用性（24 小时 × 7 天，全年无休）  
- 灾难恢复  
- 无需停机维护（比如进行备份、索引重建，压缩等任务）  
- 读取的可扩展性（可读取其他副本）  
- 副本集对应用的公开性

### 复制在 MongoDB 中的运作方式

MongoDB 使用副本集（replica set）来实现复制操作。副本集是一组托管同一数据集的 \*\*mongod\*\* 对象。在副本集中，主节点负责

1. 副本集具有 2 个或多个节点（但一般最少需要 3 个节点）。

2. 副本集只有一个主节点，其他全是从节点。

3. 所有数据都是从主节点复制到从节点上的。

4. 当发生自动故障转移或维护时，会重新推举一个新的主节点。

5. 当失败节点恢复后，该节点重新又连接到副本集中，重新作为从节点。

下图展示了一个典型的 MongoDB 复制图。客户端应用总是跟主节点交互，主节点将数据复制到从节点上。

![replication](images/replication.png)

### 副本集特点

- 具有 N 个节点的集群  
- 任何节点都可能成为主节点  
- 所有写入操作必须由主节点来完成

第 1 章 教程 | 40

- 自动故障转移  
- 自动故障恢复  
- 重新推举主节点

|  |
| --- |
| ### 建立副本集 |

|  |
| --- |
| 在本教程中，我们将把单独的一个 mongod 实例转变为副本集，步骤如下： |

|  |
| --- |
| 1. 关闭正在运行的 MongoDB 服务器。 |

|  |
| --- |
| 2. 指定 \*\*--replSet\*\* 选项来开启 MongoDB 服务器。\*\*--replSet\*\* 的基本格式如下： |

|  |
| --- |
| `mongod --port "PORT" --dbpath "YOUR\_DB\_DATA\_PATH" --replSet "REPLICA\_SET\_INSTANCE\_NAME"` |

|  |
| --- |
| #### 范例 |

|  |
| --- |
| `mongod --port 27017 --dbpath "D:\set up\mongodb\data" --replSet rs0` |

该命令会在端口 27017 处启动一个名为 rs0 的 MongoDB 实例。在命令行提示符上输入命令连接到该 MongoDB 对象上。在 Mongo

|  |
| --- |
| ### 为副本集添加成员 |

|  |
| --- |
| 为了向副本集添加成员，在多台机器上开启多个 MongoDB 实例。开启一个 MongoDB 客户端，然后使用 `rs.add()` 命令。 |

|  |
| --- |
| #### 语法格式 |

|  |
| --- |
| 基本的 `rs.add()` 命令语法格式如下所示： |

|  |
| --- |
| `>rs.add(HOST\_NAME:PORT)` |

|  |
| --- |
| #### 范例 |

假设 MongoDB 实例名称为 mongodb1.net，且运行在 27017 端口处。为了将该实例添加到副本集中，请在 MongoDB 客户端中使用

rs.add("mongod1.net:27017")

只有当连接到主节点上时，才能向副本集中添加 MongoDB 实例。要想查看是否连接的是主节点，请在 MongoDB 客户端上使用 `db.

## 分片

第 1 章 教程 | 41

分片是一种在多台机器上存储数据记录的操作，它是 MongoDB 为应对数据增长需求而采取的办法。当数据量增长时，单台机器有可能

### 为何要分片

- 将所有的写入操作复制到主节点

- 对延迟敏感的查询将在主节点上完成

- 单个副本集的节点数限制为 12 个

- 当活跃数据集过大时，内存有可能不够

- 本地磁盘空间不足

- 纵向扩展太过昂贵

### MongoDB 中的分片

下图展示了 MongoDB 使用分片集群的情形：

![sharding](images/sharding.png)

在上图中，有三个组件值得说明：

- \*\*分片\*\* 分片用来存储数据。它们提供了高可用性与数据一致性。在生产环境中，每个分片都是一个独立的副本集。

- \*\*配置服务器\*\* 配置服务器（Config server）保存着集群的元数据。该数据含有集群数据集到分片的映射关系。查询路由使用该元数

- \*\*查询路由\*\* 查询路由（Query Router）基本上就是 mongos 实例，是客户端的接口，直接操作适当的分片。查询路由定位并处理

## 创建备份

### MongoDB 数据转储

为了在 MongoDB 中创建数据库备份，需要使用 \*\*mongodump\*\* 命令。该命令会将服务器上的所有数据都转储到 dump 目录中。你

#### 格式

\*\*mongodump\*\* 命令的基本语法格式为：

`>mongodump`

#### 范例

开启 mongod 服务器。假设 mongod 服务器运行在 localhost 上，端口为 27017。在命令行上输入命令，在 MongoDB 实例的 bin 目

假设 mycol 集合包含如下数据：

`>mongodump`

上述命令会连接在 127.0.0.1 运行的服务器（端口为 27017），将所有数据备份到 \*\*/bin/dump\*\* 上。命令输出结果如下图所示：

第 1 章 教程 | 42

![mongodump](images/mongodump.png)

mongodump 命令其实包含很多选项。

|语法格式|描述|范例|

|---|---|---|

|`mongodump --host HOST\_NAME --port PORT\_NUMBER`|该命令将指定 mongod 实例上的所有数据库都进行了备份|`mon

|`mongodump --dbpath DB\_PATH --out BACKUP\_DIRECTORY`|-|`mongodump --dbpath /data/db/ --out /data/backu

|`mongodump --collection COLLECTION --db DB\_NAME`|该命令只备份那些指定路径上的指定数据库|`mongodump --colle

### 重新恢复数据

恢复备份数据使用 \*\*mongorestore\*\* 命令，该命令将备份目录中的所有数据给予恢复。

#### 语法格式

\*\*mongorestore\*\* 命令的基本语法格式为：

`> mongorestore`

该命令输入结果如下图所示：

|  |
| --- |
| ![mongorestore](images/mongorestore.png) |

|  |
| --- |
| ## 部署 |

在准备一个 MongoDB 部署时，应该先了解一下应用是如何在生产环境中运行的。使用一个持久性可重复的策略来管理部署环境无疑是

最佳的策略应包括以下几点：规划设置原型，执行负载测试，监控关键参数，并使用该信息来扩展设置。关键在于积极监控整个系统，这

为了监控部署，MongoDB 提供了多种命令。

#### mongostat

该命令检查所有运行中的 mongod 实例的状态，返回数据库操作的统计结果。这些统计命令包括插入数、查询数、更新数、删除数以及

先运行 mongod 实例，然后在另一个命令行提示符中输入命令，在 mongodb 安装目录的 bin 目录下输入 \*\*mongostat\*\*。

`D:\set up\mongodb\bin>mongostat`

该命令输出结果如下：

![](images/mongostat.png)

第 1 章 教程 | 43

#### mongotop

该命令能够记录并报告 MongoDB 实例基于每个集合的读写活动。mongotop 默认每秒返回一次结果，但我们可以修改间隔时间。你应

先运行 mongod 实例，然后在另一个命令行提示符中输入命令，在 mongodb 安装目录的 bin 目录下输入 \*\*mongotop\*\*。

|  |
| --- |
| `D:\set up\mongodb\bin>mongotop` |

|  |
| --- |
| 该命令输出结果如下所示： |

![](images/mongotop.png)

要想使 mongotop 命令返回信息的间隔时间变长，可以在 mongotop 命令的后面指定一个数字。

`D:\set up\mongodb\bin>mongotop 30`

上述命令表示每 30 秒返回数值。

除了 mongodb 工具之外，10gen 还提供了免费的托管监控服务：MongoDB Management Service（MMS）。通过它所提供的仪表

## Java

### 安装

要想在 Java 程序中使用 MongoDB，需要先确定是否安装了 MongoDB JDBC 驱动，并且要在机器上安装了 Java。查看 Java 教程

- 从路径 Download mongo.jar 处下载 jar 文件，注意下载最新版本。

- 在类路径中包括 mongo.jar 文件。

### 连接数据库

为了连接数据库，需要指定数据库名称，如果数据库不存在，mongodb 就会自动创建它。

连接数据库的代码段如下所示：

import com.mongodb.MongoClient; import com.mongodb.MongoException; import com.mongodb.Wri

teConcern; import com.mongodb.DB; import com.mongodb.DBCollection; import com.mongodb.Basi

cDBObject; import com.mongodb.DBObject; import com.mongodb.DBCursor; import com.mongod

b.ServerAddress; import java.util.Arrays;

public class MongoDBJDBC{ public static void main( String args[] ){ try{

// To connect to mongodb server MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// Now connect to your databases DB db = mongoClient.getDB( "test" ); System.out.println("Connect t

第 1 章 教程 | 44

o database successfully"); boolean auth = db.authenticate(myUserName, myPassword); System.ou t.println("Authentication: "+auth); }catch(Exception e){ System.err.println( e.getClass().getName() + ": " + e.getMessage() ); } } }

下面编译并运行上面的程序，以便测试数据库。可以针对每个需求改变路径。假设当前 JDBC 驱动版本 mongo-2.10.1.jar 可用于当前

$javac MongoDBJDBC.java $java -classpath ".:mongo-2.10.1.jar" MongoDBJDBC Connect to datab ase successfully Authentication: true

|  |
| --- |
| 如果想使用 Windows 系统的机器，则编译并执行代码如下： |

$javac MongoDBJDBC.java $java -classpath ".;mongo-2.10.1.jar" MongoDBJDBC Connect to datab ase successfully Authentication: true

|  |
| --- |
| 如果选定数据库的用户名及密码有效，则 \*\*auth\*\* 值为 true。 |

|  |
| --- |
| ### 创建集合 |

|  |
| --- |
| 创建集合需要使用 `com.mongodb.DB` 类的 `createCollection()` 方法。 |

|  |
| --- |
| 创建集合的代码段如下所示： |

import com.mongodb.MongoClient; import com.mongodb.MongoException; import com.mongodb.Wri teConcern; import com.mongodb.DB; import com.mongodb.DBCollection; import com.mongodb.Basi cDBObject; import com.mongodb.DBObject; import com.mongodb.DBCursor; import com.mongod b.ServerAddress; import java.util.Arrays;

public class MongoDBJDBC{ public static void main( String args[] ){ try{   
// To connect to mongodb server MongoClient mongoClient = new MongoClient( "localhost" , 27017 ); // Now connect to your databases DB db = mongoClient.getDB( "test" ); System.out.println("Connect t o database successfully"); boolean auth = db.authenticate(myUserName, myPassword); System.ou t.println("Authentication: "+auth); DBCollection coll = db.createCollection("mycol"); System.out.printl n("Collection created successfully"); }catch(Exception e){ System.err.println( e.getClass().getName() + ": " + e.getMessage() ); } } }

|  |
| --- |
| 编译并执行该程序，结果如下所示： |

Connect to database successfully Authentication: true Collection created successfully

第 1 章 教程 | 45

|  |
| --- |
| ### 获取/选择一个集合 |

|  |
| --- |
| 从数据库中获取/选择一个集合，使用 `com.mongodb.DBCollection` 类的 `getCollection()` 方法。代码段如下： |

import com.mongodb.MongoClient; import com.mongodb.MongoException; import com.mongodb.Wri teConcern; import com.mongodb.DB; import com.mongodb.DBCollection; import com.mongodb.Basi cDBObject; import com.mongodb.DBObject; import com.mongodb.DBCursor; import com.mongod b.ServerAddress; import java.util.Arrays;

public class MongoDBJDBC{ public static void main( String args[] ){ try{   
// To connect to mongodb server MongoClient mongoClient = new MongoClient( "localhost" , 27017 ); // Now connect to your databases DB db = mongoClient.getDB( "test" ); System.out.println("Connect t o database successfully"); boolean auth = db.authenticate(myUserName, myPassword); System.ou t.println("Authentication: "+auth); DBCollection coll = db.createCollection("mycol"); System.out.printl n("Collection created successfully"); DBCollection coll = db.getCollection("mycol"); System.out.printl n("Collection mycol selected successfully"); }catch(Exception e){ System.err.println( e.getClass().get Name() + ": " + e.getMessage() ); } } }

|  |
| --- |
| 编译并执行该程序，结果如下所示： |

Connect to database successfully Authentication: true Collection created successfully Collection myc ol selected successfully

|  |
| --- |
| ### 插入文档 |

|  |
| --- |
| 插入文档使用 `com.mongodb.DBCollection` 类的 `insert()` 方法。代码段如下所示： |

import com.mongodb.MongoClient; import com.mongodb.MongoException; import com.mongodb.Wri teConcern; import com.mongodb.DB; import com.mongodb.DBCollection; import com.mongodb.Basi cDBObject; import com.mongodb.DBObject; import com.mongodb.DBCursor; import com.mongod b.ServerAddress; import java.util.Arrays;

public class MongoDBJDBC{ public static void main( String args[] ){ try{   
// To connect to mongodb server MongoClient mongoClient = new MongoClient( "localhost" , 27017 ); // Now connect to your databases DB db = mongoClient.getDB( "test" ); System.out.println("Connect t o database successfully"); boolean auth = db.authenticate(myUserName, myPassword); System.ou t.println("Authentication: "+auth);   
DBCollection coll = db.getCollection("mycol"); System.out.println("Collection mycol selected successf

第 1 章 教程 | 46

ully"); DBCursor cursor = coll.find(); int i=1; while (cursor.hasNext()) { System.out.println("Inserted Doc ument: "+i); System.out.println(cursor.next()); i++; } }catch(Exception e){ System.err.println( e.getClas s().getName() + ": " + e.getMessage() ); } } }

|  |
| --- |
| 编译并执行该程序，结果如下所示： |

Connect to database successfully Authentication: true Collection mycol selected successfully Docum ent inserted successfully

|  |
| --- |
| ### 检索所有文档 |

选择集合中的所有文档，使用 `com.mongodb.DBCollection` 类的 `find()` 方法。该方法返回一个游标，所以需要迭代该游标。

|  |
| --- |
| 选择所有文档的代码如下所示： |

import com.mongodb.MongoClient; import com.mongodb.MongoException; import com.mongodb.Wri teConcern; import com.mongodb.DB; import com.mongodb.DBCollection; import com.mongodb.Basi cDBObject; import com.mongodb.DBObject; import com.mongodb.DBCursor; import com.mongod b.ServerAddress; import java.util.Arrays;

public class MongoDBJDBC{ public static void main( String args[] ){ try{   
// To connect to mongodb server MongoClient mongoClient = new MongoClient( "localhost" , 27017 ); // Now connect to your databases DB db = mongoClient.getDB( "test" ); System.out.println("Connect t o database successfully"); boolean auth = db.authenticate(myUserName, myPassword); System.ou t.println("Authentication: "+auth);   
DBCollection coll = db.getCollection("mycol"); System.out.println("Collection mycol selected successf ully"); DBCursor cursor = coll.find(); while (cursor.hasNext()) { DBObject updateDocument = cursor.n ext(); updateDocument.put("likes","200") col1.update(updateDocument); } System.out.println("Docum ent updated successfully"); cursor = coll.find(); int i=1; while (cursor.hasNext()) { System.out.println("U pdated Document: "+i); System.out.println(cursor.next()); i++; } }catch(Exception e){ System.err.printl n( e.getClass().getName() + ": " + e.getMessage() ); } } }

|  |
| --- |
| 编译并执行程序的结果如下： |

Connect to database successfully Authentication: true Collection mycol selected successfully Docum ent updated successfully Updated Document: 1 { "\_id" : ObjectId(7df78ad8902c), "title": "MongoDB",

第 1 章 教程 | 47

"description": "database", "likes": 100, "url": "http://www.tutorialspoint.com/mongodb/", "by": "tutorials p oint" }

|  |
| --- |
| ### 更新文档 |

|  |
| --- |
| 更新集合中的文档，使用 `com.mongodb.DBCollection` 类的 `update()` 方法。代码如下所示： |

import com.mongodb.MongoClient; import com.mongodb.MongoException; import com.mongodb.Wri teConcern; import com.mongodb.DB; import com.mongodb.DBCollection; import com.mongodb.Basi cDBObject; import com.mongodb.DBObject; import com.mongodb.DBCursor; import com.mongod b.ServerAddress; import java.util.Arrays;

public class MongoDBJDBC{ public static void main( String args[] ){ try{   
// To connect to mongodb server MongoClient mongoClient = new MongoClient( "localhost" , 27017 ); // Now connect to your databases DB db = mongoClient.getDB( "test" ); System.out.println("Connect t o database successfully"); boolean auth = db.authenticate(myUserName, myPassword); System.ou t.println("Authentication: "+auth);   
DBCollection coll = db.getCollection("mycol"); System.out.println("Collection mycol selected successf ully"); DBCursor cursor = coll.find(); while (cursor.hasNext()) { DBObject updateDocument = cursor.n ext(); updateDocument.put("likes","200") col1.update(updateDocument); } System.out.println("Docum ent updated successfully"); cursor = coll.find(); int i=1; while (cursor.hasNext()) { System.out.println("U pdated Document: "+i); System.out.println(cursor.next()); i++; } }catch(Exception e){ System.err.printl n( e.getClass().getName() + ": " + e.getMessage() ); } } }

|  |
| --- |
| 程序编译及运行结果如下： |

Connect to database successfully Authentication: true Collection mycol selected successfully Docum ent updated successfully Updated Document: 1 { "\_id" : ObjectId(7df78ad8902c), "title": "MongoDB", "description": "database", "likes": 100, "url": "http://www.tutorialspoint.com/mongodb/", "by": "tutorials p oint" }

|  |
| --- |
| ### 删除第一个文档 |

删除集合中的第一个文档，需要使用 `findOne()` 方法选中第一个文档，然后使用 `com.mongodb.DBCollection` 类的 `remove` 方法

import com.mongodb.MongoClient; import com.mongodb.MongoException; import com.mongodb.Wri teConcern; import com.mongodb.DB; import com.mongodb.DBCollection; import com.mongodb.Basi

第 1 章 教程 | 48

cDBObject; import com.mongodb.DBObject; import com.mongodb.DBCursor; import com.mongod b.ServerAddress; import java.util.Arrays;

public class MongoDBJDBC{ public static void main( String args[] ){ try{   
// To connect to mongodb server MongoClient mongoClient = new MongoClient( "localhost" , 27017 ); // Now connect to your databases DB db = mongoClient.getDB( "test" ); System.out.println("Connect t o database successfully"); boolean auth = db.authenticate(myUserName, myPassword); System.ou t.println("Authentication: "+auth);   
DBCollection coll = db.getCollection("mycol"); System.out.println("Collection mycol selected successf ully"); DBObject myDoc = coll.findOne(); col1.remove(myDoc); DBCursor cursor = coll.find(); int i=1; w hile (cursor.hasNext()) { System.out.println("Inserted Document: "+i); System.out.println(cursor.nex t()); i++; } System.out.println("Document deleted successfully"); }catch(Exception e){ System.err.printl n( e.getClass().getName() + ": " + e.getMessage() ); } } }

|  |
| --- |
| 程序编译并执行结果如下所示： |

Connect to database successfully Authentication: true Collection mycol selected successfully Docum ent deleted successfully

|  |
| --- |
| 剩余的 mongodb 的 `save()`、`limit()`、`skip()`、`sort()` 等方法将在教程剩余部分予以介绍。 |

|  |
| --- |
| ## PHP |

要想在 MongoDB 上使用 PHP，需要使用 mongodb PHP 驱动。从[该链接处](https://s3.amazonaws.com/drivers.mongodb.org

|  |
| --- |
| `extension=php\_mongo.dll` |

|  |
| --- |
| ### 创建连接并选择数据库 |

|  |
| --- |
| 创建连接需要指定数据库名称，如果数据库不存在，则 mongodb 会自动创建它。 |

|  |
| --- |
| 连接数据库的代码如下所示： |

mydb; echo "Database mydb selected"; ?>

|  |
| --- |
| 该程序的执行结果如下所示： |

Connection to database successfully Database mydb selected

第 1 章 教程 | 49

|  |
| --- |
| ### 创建集合 |

|  |
| --- |
| 创建集合的代码如下所示： |

mydb; echo "Database mydb selected"; $collection = $db->createCollection("mycol"); echo "Collectio n created succsessfully"; ?>

|  |
| --- |
| 执行程序的结果如下所示： |

Connection to database successfully Database mydb selected Collection created succsessfully

|  |
| --- |
| ### 插入文档 |

|  |
| --- |
| 插入文档使用 `insert()` 方法。 |

mydb; echo "Database mydb selected"; $collection = $db->mycol; echo "Collection selected succses sfully"; $document = array( "title" => "MongoDB", "description" => "database", "likes" => 100, "url" => "h ttp://www.tutorialspoint.com/mongodb/", "by", "tutorials point" ); $collection->insert($document); echo "Document inserted successfully"; ?>

|  |
| --- |
| 执行程序的结果如下所示： |

Connection to database successfully Database mydb selected Collection selected succsessfully Doc ument inserted successfully

|  |
| --- |
| ### 寻找所有文档 |

|  |
| --- |
| 选择集合中的所有文档，使用 `find()` 方法。 |

mydb; echo "Database mydb selected"; $collection = $db->mycol; echo "Collection selected succses sfully"; $cursor = $collection->find(); // iterate cursor to display title of documents foreach ($cursor as $document) { echo $document["title"] . "\n"; } ?>

|  |
| --- |
| 程序执行的结果如下所示： |

Connection to database successfully Database mydb selected Collection selected succsessfully { "titl e": "MongoDB" }

|  |  |
| --- | --- |
| |  | | --- | | ### 更新文档 | |

第 1 章 教程 | 50

|  |
| --- |
| 更新文档使用 `update()` 方法。 |

|  |
| --- |
| 下例将把插入文档的标题改为 \*\*MongoDB Tutorial\*\*。代码段如下所示： |

mydb; echo "Database mydb selected"; $collection = $db->mycol; echo "Collection selected succses sfully"; // now update the document $collection->update(array("title"=>"MongoDB"), array('$set'=>arr ay("title"=>"MongoDB Tutorial"))); echo "Document updated successfully"; // now display the updated document $cursor = $collection->find(); // iterate cursor to display title of documents echo "Updated d ocument"; foreach ($cursor as $document) { echo $document["title"] . "\n"; } ?>

|  |
| --- |
| 程序执行的结果如下所示： |

Connection to database successfully Database mydb selected Collection selected succsessfully Doc ument updated successfully Updated document { "title": "MongoDB Tutorial" }

|  |
| --- |
| ### 删除文档 |

|  |
| --- |
| 删除文档使用 `remove()` 方法。 |

|  |
| --- |
| 下例中将把标题为 \*\*MongoDB Tutorial\*\* 的文档全部删除。代码如下所示： |

mydb; echo "Database mydb selected"; $collection = $db->mycol; echo "Collection selected succses sfully"; // now remove the document $collection->remove(array("title"=>"MongoDB Tutorial"),false); e cho "Documents deleted successfully"; // now display the available documents $cursor = $collection->find(); // iterate cursor to display title of documents echo "Updated document"; foreach ($cursor as $document) { echo $document["title"] . "\n"; } ?>

|  |
| --- |
| 程序执行的结果如下所示： |

Connection to database successfully Database mydb selected Collection selected succsessfully Doc uments deleted successfully ```

在上述范例中，第二个参数是布尔值类型，用于remove() 方法的justOne 字段。

剩余的 mongodb 的findOne() 、save() 、limit() 、skip() 、sort() 等方法将在教程剩余部分予以介绍。



|  |  |
| --- | --- |
| |  | | --- | | 2 | |

高级教程

第 2 章 高级教程 | 52

关系

|  |
| --- |
| MongoDB 中的关系表示文档之间的逻辑相关方式。关系可以通过内嵌 内嵌（Embedded）或引用 引用（Reference |

d）两种方式建模。这样的关系可能是 1：1、1：N、N：1，也有可能是 N：N。

先来考虑保存用户地址的例子。一个用户可能有多个地址，这是一个 1：N 的关系。

下面是一个结构非常简单的user user 文档。

|  |
| --- |
| { |

|  |
| --- |
| "\_id":ObjectId("52ffc33cd85242f436000001"), |

|  |
| --- |
| "name": "Tom Hanks", |

|  |
| --- |
| "contact": "987654321", |

|  |
| --- |
| "dob": "01-01-1991" |

|  |
| --- |
| }  下面是address address 文档的结构： |

|  |
| --- |
| { |

|  |
| --- |
| "\_id":ObjectId("52ffc4a5d85242602e000000"), |

|  |
| --- |
| "building": "22 A, Indiana Apt", |

|  |
| --- |
| "pincode": 123456, |

|  |
| --- |
| "city": "Los Angeles", |

|  |
| --- |
| "state": "California" |

|  |
| --- |
| } |

内嵌关系的建模

利用内嵌方法，我们将把地址文档内嵌到 user 文档中。

|  |
| --- |
| { |

|  |
| --- |
| "\_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" |

第 2 章 高级教程 | 53

|  |
| --- |
| }, |

|  |
| --- |
| { |

|  |
| --- |
| "building": "170 A, Acropolis Apt", |

|  |
| --- |
| "pincode": 456789, |

|  |
| --- |
| "city": "Chicago", |

|  |
| --- |
| "state": "Illinois" |

|  |
| --- |
| }] |

|  |
| --- |
| } |

该方法会将所有相关数据都保存在一个文档中，从而易于检索和维护。通过一个查询命令就能检索整个文档：

|  |  |
| --- | --- |
| |  | | --- | | >db.users.findOne({"name":"Tom Benzamin"},{"address":1}) | |
| 其中的db db 和users users 分别对应的是数据库和集合。 |

这种方法的缺点是，如果内嵌文档不断增长，会对读写性能造成影响。

引用关系的建模

这是一种设计归一化关系的方法。按照这种方法，所有的用户和地址文档都将分别存放，而用户文档会包含一个

字段，用来引用地址文档id id 字段。

|  |
| --- |
| { |

|  |
| --- |
| "\_id":ObjectId("52ffc33cd85242f436000001"), |

|  |
| --- |
| "contact": "987654321", |

|  |
| --- |
| "dob": "01-01-1991", |

|  |
| --- |
| "name": "Tom Benzamin", |

|  |
| --- |
| "address\_ids": [ |

|  |
| --- |
| ObjectId("52ffc4a5d85242602e000000"), |

|  |
| --- |
| ObjectId("52ffc4a5d85242602e000001") |

|  |
| --- |
| ] |

|  |
| --- |
| }  如上所示，user 文档包含的数组字段address\_ids address\_ids 含有相应地址的 ObjectId 对象。利用这些 ObjectId，能够  查询地址文档，从而获取地址细节信息。利用这种方法时，需要进行两种查询：首先从user user 文档处获取addres  s\_ids，其次从address address 集合中获取这些地址。 |

|  |
| --- |
| >var result = db.users.findOne({"name":"Tom Benzamin"},{"address\_ids":1}) |

|  |
| --- |
| >var addresses = db.address.find({"\_id":{"$in":result["address\_ids"]}}) |

第 2 章 高级教程 | 54

数据库引用

|  |
| --- |
| 在上一节中，我们使用引用关系 引用关系实现了归一化的数据库结构，这种引用关系也被称作手动引用 手动引用，即可以手动地将  引用文档的 id 保存在其他文档中。但在有些情况下，文档包含其他集合的引用时，我们可以使用数据库引用 数据库引用（M |

ongoDB DBRefs）。

数据库引用 vs 手动引用

我们将利用一个例子来展示如何用数据库引用代替手动引用。假设一个数据库中存储有多个类型的地址（家庭地

址、办公室地址、邮件地址，等等），这些地址保存在不同的集合中（address\_home、address\_office、add

ress\_mailing，等等）。当user user 集合的文档引用了一个地址时，它还需要按照地址类型来指定所需要查看的集

合。这种情况下，一个文档引用了许多结合的文档，所以就应该使用 DBRef。

使用数据库引用

在 DBRef 中有三个字段：

$ref 该字段指定所引用文档的集合。

$id 该字段指定引用文档的-id 字段$db $db 该字段是可选的，包含引用文档所在数据库的名称。

假如在一个简单的 user 文档中包含着 DBRef 字段address address，如下所示：

|  |
| --- |
| { |

|  |
| --- |
| "\_id":ObjectId("53402597d852426020000002"), |

|  |
| --- |
| "address": { |

|  |
| --- |
| "$ref": "address\_home", |

|  |
| --- |
| "$id": ObjectId("534009e4d852427820000002"), |

|  |
| --- |
| "$db": "tutorialspoint"}, |

|  |
| --- |
| "contact": "987654321", |

|  |
| --- |
| "dob": "01-01-1991", |

|  |
| --- |
| "name": "Tom Benzamin" |

|  |
| --- |
| }  数据库引用字段address address 指定出，引用地址文档位于tutorialspoint tutorialspoint 数据库的address\_home address\_home 集合中，并且它 |

的 id 为534009e4d852427820000002 。

下例展示了，在由$ref $ref 所指定的集合（本例中为address\_home address\_home）中，如何动态查找由$id $id 所确定的文档。

第 2 章 高级教程 | 55

|  |
| --- |
| >var user = db.users.findOne({"name":"Tom Benzamin"}) |

|  |
| --- |
| >var dbRef = user.address |

|  |
| --- |
| >db[dbRef.$ref].findOne({"\_id":(dbRef.$id)})  上述代码返回了address\_home address\_home 集合中的地址文档，如下所示： |

|  |
| --- |
| { |

|  |
| --- |
| "\_id" : ObjectId("534009e4d852427820000002"), |

|  |
| --- |
| "building" : "22 A, Indiana Apt", |

|  |
| --- |
| "pincode" : 123456, |

|  |
| --- |
| "city" : "Los Angeles", |

|  |
| --- |
| "state" : "California" |

|  |
| --- |
| } |

第 2 章 高级教程 | 56

覆盖索引查询

何为覆盖查询

在每一个 MongoDB 官方文档中，覆盖查询都具有以下两个特点：

|  |  |
| --- | --- |
| • • | 查询中的所有字段都属于一个索引； 查询所返回的所有字段也都属于同一索引内。 |

既然查询中的所有字段都属于一个索引，MongoDB 就会利用同一索引，匹配查询集合并返回结果，而不需要实

际地查看文档。因为索引存在于 RAM 中，从索引中获取数据要比通过扫描文档获取数据快得多。

使用覆盖查询

为了测试覆盖查询，假设在一个users users 集合中包含下列文档：

|  |
| --- |
| { |

|  |
| --- |
| "\_id": ObjectId("53402597d852426020000002"), |

|  |
| --- |
| "contact": "987654321", |

|  |
| --- |
| "dob": "01-01-1991", |

|  |
| --- |
| "gender": "M", |

|  |
| --- |
| "name": "Tom Benzamin", |

|  |
| --- |
| "user\_name": "tombenzamin" |

|  |
| --- |
| }  下面我们将为users users 集合中的gender gender 和user\_name user\_name 字段创建一个符合索引。使用下列查询： |

|  |
| --- |
| >db.users.ensureIndex({gender:1,user\_name:1}) |

这一索引将覆盖下列查询：

|  |
| --- |
| >db.users.find({gender:"M"},{user\_name:1,\_id:0}) |

也就是说，对于上面的查询，MongoDB 不会去查看文档，转而从索引数据中获取所需的数据，这显然要快得

多。

既然索引不包含\_id \_id 字段，那么当查询中默认返回 \_id 时，我们可以在 MongoDB 的查询结果集中将其排除

掉。下面的查询就不会被覆盖。

|  |
| --- |
| >db.users.find({gender:"M"},{user\_name:1}) |

第 2 章 高级教程 | 57

最后，需要记住的是，如果出现下列情况，索引不能覆盖查询：

|  |  |
| --- | --- |
| • • | 索引字段是数组  索引字段是子文档 |

第 2 章 高级教程 | 58

查询分析

|  |
| --- |
| 对于衡量数据库及索引设计的效率来说，分析查询是一个很重要的衡量方式。经常使用的查询有$explain $explain 和$hi |

nt。

使用 $explain

$explain 操作提供的消息包括：查询消息、查询所使用的索引以及其他的统计信息。在分析索引优化方案时，这

是一个非常有用的工具。

在上一节中，我们使用如下查询，针对 users 集合的字段 gender 和 user\_name 创建了索引：

|  |  |
| --- | --- |
| |  | | --- | | >db.users.ensureIndex({gender:1,user\_name:1}) | |
| 接下来，在下列查询中使用$explain $explain。 |

|  |
| --- |
| >db.users.find({gender:"M"},{user\_name:1,\_id:0}).explain() |

上述explain() 查询返回下列分析结果：

{   
 "cursor" : "BtreeCursor gender\_1\_user\_name\_1",   
 "isMultiKey" : false,   
 "n" : 1,   
 "nscannedObjects" : 0,   
 "nscanned" : 1,   
 "nscannedObjectsAllPlans" : 0,   
 "nscannedAllPlans" : 1,   
 "scanAndOrder" : false,   
 "indexOnly" : true,   
 "nYields" : 0,   
 "nChunkSkips" : 0,   
 "millis" : 0,   
 "indexBounds" : {   
 "gender" : [   
 [   
 "M",   
 "M"   
 ]   
 ],   
 "user\_name" : [

第 2 章 高级教程 | 59

|  |
| --- |
| [ |

|  |
| --- |
| { |

|  |
| --- |
| "$minElement" : 1 |

|  |
| --- |
| }, |

|  |
| --- |
| { |

|  |
| --- |
| "$maxElement" : 1 |

|  |
| --- |
| } |

|  |
| --- |
| ] |

|  |
| --- |
| ] |

|  |
| --- |
| } |

|  |
| --- |
| } |

我们将在结果集中看到这些字段：

• indexOnly 的真值代表该查询使用了索引。

• cursor 字段指定了游标所用的类型。BTreeCursor 类型代表了使用了索引并且提供了所用索引的名称。Ba

sicCursor 表示进行了完整扫描，没有使用任何索引。

|  |  |
| --- | --- |
| • • • | n 代表所返回的匹配文档的数量。  nscannedObjects 表示已扫描文档的总数。nscanned 所扫描的文档或索引项的总数。 |

使用 $hint

$hint 操作符强制索引优化器使用指定的索引运行查询。这尤其适用于测试带有多个索引的查询性能。比如，下列

查询指定了用于该查询的 gender 和 user\_name 字段的索引：

|  |
| --- |
| >db.users.find({gender:"M"},{user\_name:1,\_id:0}).hint({gender:1,user\_name:1}) |

使用 $hint 来优化上述查询：

|  |  |
| --- | --- |
| |  | | --- | | >db.users.find({gender:"M"},{user\_name:1,\_id:0}).hint({gender:1,user\_name:1}).explain() | |

第 2 章 高级教程 | 60

原子操作

|  |
| --- |
| MongoDB 并不支持多文档原子事务 多文档原子事务（multi-document atomic transactions）。但它提供了针对单个文档的 |

原子操作。假如一个文档包含数百个字段，则 update 语句将更新所有的字段，或者一个也不更新，从而维持了

文档级的原子性。

原子操作数据模型

维持原子性的建议方法是利用内嵌文档 内嵌文档（embedded document）将所有经常更新的相关信息都保存在一个文档

中。这能确保所有针对单一文档的更新具有原子性。

考虑下列关于产品的文档：

{   
 "\_id":1,   
 "product\_name": "Samsung S3",   
 "category": "mobiles",   
 "product\_total": 5,   
 "product\_available": 3,   
 "product\_bought\_by": [   
 {   
 "customer": "john",   
 "date": "7-Jan-2014"   
 },   
 {   
 "customer": "mark",   
 "date": "8-Jan-2014"   
 }   
 ]

|  |
| --- |
| }  在这个文档中，将购买产品的顾客的信息内嵌在product\_bought\_by product\_bought\_by 字段中。无论何时，只要新顾客购买产  品，我们就能使用product\_available product\_available 字段查看产品是否还有足够的数量。如果产品还有，就减少 product\_av  ailable 字段值，并且将新顾客的内嵌文档插入到 product\_bought\_by 字段中。使用findAndModify findAndModify 命令来实 |

现该功能，因为它能同时搜索并更新文档。

|  |
| --- |
| >db.products.findAndModify({ |

|  |
| --- |
| query:{\_id:2,product\_available:{$gt:0}}, |

|  |
| --- |
| update:{ |

|  |
| --- |
| $inc:{product\_available:-1}, |

第 2 章 高级教程 | 61

|  |
| --- |
| $push:{product\_bought\_by:{customer:"rob",date:"9-Jan-2014"}} |

|  |
| --- |
| } |

|  |
| --- |
| }) |

上述内嵌文档并使用 findAndModify 查询的方法确保了，只有当产品还有足够数量时，才更新产品购买信息。在 整个过程中，同一查询中的事务是原子性的。

与之相反的情况是，我们可能会想分别保持产品可用性与购买产品的顾客信息。在这种情况下，我们会首先使用 第一个查询来检查产品是否够用。然后在第二个查询中更新购买信息。但在这两个查询的执行过程之间，其他一 些顾客也可能会购买了产品，从而使产品变得不够用了。由于没有了解到这种情况，我们的第二个查询根据第一 个查询的结果进行了更新。这将造成数据库的不一致性。

第 2 章 高级教程 | 62

高级索引

|  |
| --- |
| 假如一个users users 集合中具有下列文档： |

{

|  |
| --- |
| "address": { |

|  |
| --- |
| "city": "Los Angeles", |

|  |
| --- |
| "state": "California", |

|  |
| --- |
| "pincode": "123" |

|  |
| --- |
| }, |

|  |
| --- |
| "tags": [ |

|  |
| --- |
| "music", |

|  |
| --- |
| "cricket", |

|  |
| --- |
| "blogs" |

|  |
| --- |
| ], |

|  |
| --- |
| "name": "Tom Benzamin" |

|  |
| --- |
| }  上述文档包含一个地址子文档 地址子文档（address sub-document）与一个标签数组 标签数组（tags array）。 |

索引数组字段

假设我们想要根据标签来搜索用户文档。首先在集合中创建一个标签数组的索引。

反过来说，在标签数组上创建一个索引，也就为每一个字段创建了单独的索引项。因此在该例中，当我们创建了

标签数组的索引时，也就为它的music（音乐）、cricket（板球）以及 blog（博客）值创建了独立的索引。

使用下列命令创建标签数据的索引：

|  |
| --- |
| >db.users.ensureIndex({"tags":1}) |

创建完该索引后，按照如下方式搜索集合中的标签字段：

|  |  |
| --- | --- |
| |  | | --- | | >db.users.find({tags:"cricket"}) | |
| 为了验证所使用索引的正确性，使用explain explain 命令，如下所示： |

|  |
| --- |
| >db.users.find({tags:"cricket"}).explain() |

上述 explain 命令的执行结果是 "cursor" : "BtreeCursor tags\_1"，表示使用了正确的索引。

第 2 章 高级教程 | 63

索引子文档字段

假设需要根据市（city）、州（state）、个人身份号码（pincode）字段来搜索文档。因为所有这些字段都属于

地址子文档字段的一部分，所以我们将在子文档的所有字段上创建索引。

使用如下命令在子文档的所有三个字段上创建索引：

|  |
| --- |
| >db.users.ensureIndex({"address.city":1,"address.state":1,"address.pincode":1}) |

一旦创建了索引，就可以使用索引来搜索任何子文档字段：

|  |
| --- |
| >db.users.find({"address.city":"Los Angeles"}) |

记住，查询表达式必须遵循指定索引的顺序。因此上面创建的索引将支持如下查询：

|  |
| --- |
| >db.users.find({"address.city":"Los Angeles","address.state":"California"}) |

另外也支持如下这样的查询：

|  |  |
| --- | --- |
| |  | | --- | | >db.users.find({"address.city":"LosAngeles","address.state":"California","address.pincode":"123"}) | |

第 2 章 高级教程 | 64

索引限制

额外开销

每个索引都会占据一些空间，从而也会在每次插入、更新与删除操作时产生一定的开销。所以如果集合很少使用

读取操作，就尽量不要使用索引。

内存使用

因为索引存储在内存中，所以应保证索引总体的大小不超过内存的容量。如果索引总体积超出了内存容量，就会

删除部分索引，从而降低性能。

查询限制

当查询使用以下元素时，不能使用索引：

|  |  |
| --- | --- |
| • • • | 正则表达式或否定运算符（$nin 、$not ，等等）算术运算符（比如$mod ） $where 子句 |

因此，经常检查查询使用的索引是一个明智的做法。

索引键限制

自 MongoDB 2.6 版本起，如果已有索引字段的值超出了索引键限制，则无法创建索引。

插入文档超过索引键限制

如果文档的索引字段值超出了索引键的限制，MongoDB 不会将任何文档插入已索引集合。类似于使用 mongor

estore 和 mongoimport 工具时的情况。

第 2 章 高级教程 | 65

最大范围

|  |  |
| --- | --- |
| • • • | 集合索引数不能超过 64 个。  索引名称长度不能大于 125 个字符。复合索引最多能有 31 个被索引的字段。 |

第 2 章 高级教程 | 66

ObjectId

前面的几章中都涉及到了 MongoDB 的对象 id。本章将介绍 ObjectId 的结构。

ObjectId 是一个 12 字节的 BSON 类型，其结构如下：

|  |  |
| --- | --- |
| • | 前 4 个字节代表 UNIX 的时间戳（以秒计）。 |
| • | 接下来的 3 个字节代表机器标识符 机器标识符。 |
| • | 接下来的 2 个字节代表进程 id 进程 id。 |
| • | 最后 3 个字节代表随机数 随机数。 |
| MongoDB 使用 ObjectId 作为每一文档的\_id \_id 字段的默认值（在创建文档时产生）。ObjectId 的复杂组合保证 | |

了 \_id 字段的唯一性。

创建新的 ObjectId

使用下列代码创建新的 ObjectId：

|  |
| --- |
| >newObjectId = ObjectId() |

上述代码返回一个唯一的 id：

|  |
| --- |
| ObjectId("5349b4ddd2781d08c09890f3") |

如果不用 MongoDB 来生成，可以自己提供一个 12 字节的 id：

|  |
| --- |
| >myObjectId = ObjectId("5349b4ddd2781d08c09890f4") |

创建文档时间戳

因为 \_id ObjectId 默认保存 4 字节的时间戳，所以在大多数情况下不需要保存文档的创建时间。使用getTimes

tamp 方法可获取文档的创建时间。

|  |
| --- |
| >ObjectId("5349b4ddd2781d08c09890f4").getTimestamp() |

返回标准时期格式表示的文档创建时间：

|  |
| --- |
| ISODate("2014-04-12T21:49:17Z") |

第 2 章 高级教程 | 67

将 ObjectId 转换为字符串

在某些情况下，你需要用字符串格式表示 ObjectId 值，使用如下命令可实现这一点：

|  |
| --- |
| >newObjectId.str |

上述代码以 GUID 格式返回字符串。

|  |  |
| --- | --- |
| |  | | --- | | 5349b4ddd2781d08c09890f3 | |

第 2 章 高级教程 | 68

Map Reduce

在 MongoDB 文档中，Map-Reduce（映射归约）是一种将大量数据压缩成有用的聚合结果的数据处理范

式。MongoDB 使用mapReduce mapReduce 命令来实现映射归约操作。映射归约通常用来处理大型数据。

映射归约命令

mapReduce 命令的基本格式为：

|  |
| --- |
| >db.collection.mapReduce( |

|  |
| --- |
| function() {emit(key,value);}, //map function |

|  |
| --- |
| function(key,values) {return reduceFunction}, //reduce function |

|  |
| --- |
| { |

|  |
| --- |
| out: collection, |

|  |
| --- |
| query: document, |

|  |
| --- |
| sort: document, |

|  |
| --- |
| limit: number |

|  |
| --- |
| } |

|  |
| --- |
| ) |

mapReduce 函数首先查询集合，然后将结果文档利用 emit 函数映射为键值对，然后再根据有多个值的键来简

化。

上述语法格式中：

|  |  |
| --- | --- |
| • • • • • • | map 一个 JavaScript 函数，将一个值与键对应起来，并生成键值对。reduce 一个 JavaScript 函数，用来减少或组合所有拥有同一键的文档。out 指定映射归约查询结果的位置。  query 指定选择文档所用的选择标准（可选的）。  sort 指定可选的排序标准。  limit 指定返回的文档的最大数量值（可选的）。 |

使用 mapReduce

以下面这个存储用户发帖的文档结构。该文档存储用户的用户名（user\_name）和发帖状态（status）。

第 2 章 高级教程 | 69

|  |
| --- |
| { |

|  |
| --- |
| "post\_text": "tutorialspoint is an awesome website for tutorials", |

|  |
| --- |
| "user\_name": "mark", |

|  |
| --- |
| "status":"active" |

|  |
| --- |
| }  在posts posts 集合上使用 mapReduce 函数选择所有的活跃帖子，将它们基于用户名组合起来，然后计算每个用户 |

的发帖量。代码如下：

|  |
| --- |
| >db.posts.mapReduce( |

|  |
| --- |
| function() { emit(this.user\_id,1); }, |

|  |
| --- |
| function(key, values) {return Array.sum(values)}, |

|  |
| --- |
| { |

|  |
| --- |
| query:{status:"active"}, |

|  |
| --- |
| out:"post\_total" |

|  |
| --- |
| } |

|  |
| --- |
| ) |

上面的 mapReduce 查询输出结果如下：

|  |
| --- |
| { |

|  |
| --- |
| "result" : "post\_total", |

|  |
| --- |
| "timeMillis" : 9, |

|  |
| --- |
| "counts" : { |

|  |
| --- |
| "input" : 4, |

|  |
| --- |
| "emit" : 4, |

|  |
| --- |
| "reduce" : 2, |

|  |
| --- |
| "output" : 2 |

|  |
| --- |
| }, |

|  |
| --- |
| "ok" : 1, |

|  |
| --- |
| } |

结果显示，只有 4 个文档符合查询条件（status:"active" ），于是map 函数就生成了 4 个带有键值对的文

档，而最终reduce 函数将具有相同键值的映射文档变为了 2 个。

要想查看mapReduce 查询的结果，使用find 操作符。

|  |
| --- |
| >db.posts.mapReduce( |

|  |
| --- |
| function() { emit(this.user\_id,1); }, |

|  |
| --- |
| function(key, values) {return Array.sum(values)}, |

|  |
| --- |
| { |

|  |
| --- |
| query:{status:"active"}, |

|  |
| --- |
| out:"post\_total" |

|  |
| --- |
| } |

|  |
| --- |
| ).find() |

第 2 章 高级教程 | 70

上述查询的结果如下，显示出用户 tom 和 mark 都发了 2 个活跃的帖子。

|  |
| --- |
| { "\_id" : "tom", "value" : 2 } |

|  |
| --- |
| { "\_id" : "mark", "value" : 2 } |

MapReduce 查询同样也可以用来构建大型复杂的聚合查询，自定义 JavaScript 函数使得 MapReduce 更为灵

活与强大。

第 2 章 高级教程 | 71

全文检索

|  |
| --- |
| MongoDB 从 2.4 版本起就开始支持全文索引，以便搜索字符串内容。文本搜索 文本搜索使用字干搜索技术查找字符串字 |

段中的指定词语，丢弃字干停止词（比如 a、an、the等）。迄今为止，MongoDB 支持大约 15 种语言。

启用文本搜索

最初的文本搜索只是一种试验性功能，但从 2.6 版本起就成为默认功能了。但如果使用的是之前的 MongoD

B，则需要使用下列代码启用文本搜索：

|  |
| --- |
| >db.adminCommand({setParameter:true,textSearchEnabled:true}) |

创建文本索引

假设在posts posts 集合中的下列文档中含有帖子文本及其标签。

|  |
| --- |
| { |

|  |
| --- |
| "post\_text": "enjoy the mongodb articles on tutorialspoint", |

|  |
| --- |
| "tags": [ |

|  |
| --- |
| "mongodb", |

|  |
| --- |
| "tutorialspoint" |

|  |
| --- |
| ] |

|  |
| --- |
| } |

我们将在post\_text 字段上创建文本索引，以便搜索帖子文本之内的内容。

|  |
| --- |
| >db.posts.ensureIndex({post\_text:"text"}) |

使用文本索引

|  |
| --- |
| 现在我们已经在post\_text 字段上创建了文本索引，接下来搜索包含tutorialspoint tutorialspoint 文本内容的帖子。 |

|  |  |
| --- | --- |
| |  | | --- | | >db.posts.find({$text:{$search:"tutorialspoint"}}) | |
| 在返回的结果文档中，果然包含了具有tutorialspoint tutorialspoint 文本的帖子文本。 |

|  |
| --- |
| { |

|  |
| --- |
| "\_id" : ObjectId("53493d14d852429c10000002"), |

第 2 章 高级教程 | 72

|  |
| --- |
| "post\_text" : "enjoy the mongodb articles on tutorialspoint", |

|  |
| --- |
| "tags" : [ "mongodb", "tutorialspoint" ] |

|  |
| --- |
| } |

|  |
| --- |
| { |

|  |
| --- |
| "\_id" : ObjectId("53493d1fd852429c10000003"), |

|  |
| --- |
| "post\_text" : "writing tutorials on mongodb", |

|  |
| --- |
| "tags" : [ "mongodb", "tutorial" ] |

|  |
| --- |
| } |

如果用的是旧版本的 MongoDB，则需使用下列代码：

|  |
| --- |
| >db.posts.runCommand("text",{search:" tutorialspoint "}) |

总之，与普通搜索相比，使用文本搜索可极大地改善搜索效率。

删除文本索引

要想删除已有的文本索引，首先要找到索引名称：

|  |  |
| --- | --- |
| |  | | --- | | >db.posts.getIndexes() | |
| 从上述查询中获取了索引名称后，运行下列命令，post\_text\_text post\_text\_text 是我们需要删掉的索引名称。 |

|  |  |
| --- | --- |
| |  | | --- | | >db.posts.dropIndex("post\_text\_text") | |

第 2 章 高级教程 | 73

正则表达式

正则表达式在所有语言当中都是经常会用到的一个功能，可以用来搜索模式或字符串中的单词。MongoDB 也提

供了这一功能，使用$regex $regex 运算符来匹配字符串模式。MongoDB 使用 PCRE（可兼容 Perl 的正则表达

式）作为正则表达式语言。

与文本搜索不同，使用正则表达式不需要使用任何配置或命令。

假如posts posts 集合有下面这个文档，它包含着帖子文本及其标签。

|  |
| --- |
| { |

|  |
| --- |
| "post\_text": "enjoy the mongodb articles on tutorialspoint", |

|  |
| --- |
| "tags": [ |

|  |
| --- |
| "mongodb", |

|  |
| --- |
| "tutorialspoint" |

|  |
| --- |
| ] |

|  |
| --- |
| } |

使用正则表达式

|  |
| --- |
| 使用下列正则表达式来搜索包含tutorialspoint tutorialspoint 的所有帖子。 |

|  |
| --- |
| >db.posts.find({post\_text:{$regex:"tutorialspoint"}}) |

同样的查询也可以写作下列形式：

|  |
| --- |
| >db.posts.find({post\_text:/tutorialspoint/}) |

使用不区分大小写的正则表达式

要想使搜索不区分大小写，使用$options 参数和值$i 。下列命令将搜索包含 “tutorialspoint” 的字符

串，不区分大小写。

|  |
| --- |
| >db.posts.find({post\_text:{$regex:"tutorialspoint",$options:"$i"}}) |

该查询返回的一个结果文档中含有不同大小写的 “tutorialspoint” 文本。

|  |
| --- |
| { |

|  |
| --- |
| "\_id" : ObjectId("53493d37d852429c10000004"), |

|  |
| --- |
| "post\_text" : "hey! this is my post on TutorialsPoint", |

第 2 章 高级教程 | 74

|  |
| --- |
| "tags" : [ "tutorialspoint" ] |

|  |
| --- |
| } |

使用正则表达式来处理数组元素

我们还可以在数组字段上使用正则表达式。在实现标签的功能时，这尤为重要。假如想搜索标签以 “tutorial”开始（tutorial、tutorials、tutorialpoint 或 tutorialphp）的帖子，可以使用下列代码：

|  |
| --- |
| >db.posts.find({tags:{$regex:"tutorial"}}) |

优化正则表达式查询

• 如果文档字段已经设置了索引，查询将使用索引值来匹配正则表达式，从而使查询效率相对于扫描整个集合 的正则表达式而言大大提高。

|  |  |
| --- | --- |
| • | 如果正则表达式为前缀表达式，所有的匹配结果都要在前面带有特殊的前缀字符串。比如，如果正则表达式  为^tut ^tut，那么查询将搜索所有以tut tut 开始的字符串。 |

第 2 章 高级教程 | 75

Rockmongo 管理工具

Rockmongo 是一个 MongoDB 的管理工具，可以用来管理服务器、数据库、集合、文档、索引以及很多其他内 容。它的操作简单便利，易于读写创建文档。它的作用有点像是 PHP 和 MySQL 所使用的 PHPMyAdmin 工 具。

下载 Rockmongo

从这里(http://rockmongo.com/downloads) 下载最新版的 Rockmongo。

安装 Rockmongo

|  |
| --- |
| 下载完毕后，可以将包解压缩至服务器的根目录处，将解压文件夹重新命名为rockmongo rockmongo。打开任何一个浏览  器，从rockmongo 文件夹处访问index.php index.php 页面。当询问用户名和密码时，分别输入 admin/admin。 |

使用 Rockmongo

下面介绍一些 Rockmongo 的基本操作。

|  |  |
| --- | --- |
| • | 创建新数据库 |
| 要想创建新数据库，请点击Databases Databases 标签，然后点击Create New Database Create New Database。在下一个界面中，为新数据  库命名，然后点击Create Create 即可。你就会看到一个新数据库添加到了左边的面板中。 | |
| • | 创建新集合 |
| 要想在数据库中创建新的集合，从左边的面板中选择数据库，点击最上方的New Collection New Collection。然后提供新集合的  名称，不用理会其他字段（Is Capped，Size and Max），点击Create Create 即可。新的集合就创建好了，就显示在 | |

左边的面板中。

• 创建新文档

要想创建新文档，点击想要在其中添加文档的集合。点击一个集合，就能看到其中的所有文档。点击最上方的Ins ert 创建新文档。输入的文档数据既可以是 JSON 格式，也可以是数组格式。输入文档数据之后，点击Save Save 即 可创建好一个新的文档。

第 2 章 高级教程 | 76

|  |  |
| --- | --- |
| • | 数据的导出与导入 |
| 要想导入/导出任何集合中的数据，点击该集合，然后点击上方面板的Export/Import Export/Import。根据接下来的指令，将数 | |

据导出为 zip 格式的压缩数据，或者将同样格式的 zip 文件导入。

第 2 章 高级教程 | 77

GridFS

GridFS 简介

GridFS 是 MongoDB 的一个用来存储/获取大型数据（图像、音频、视频等类型的文件）的规范。它相当于一个

存储文件的文件系统，但它的数据存储在 MongoDB 的集合中。GridFS 能存储超过文档尺寸限制（16 MB）的

文件。

GridFS 将文件分解成块，将每块数据保存在不同的文档中，每块大小最高为 255 KB。

GridFS 默认使用fs.files fs.files 和fs.chunks fs.chunks 集合来存储文件的元数据和块。每个块都由唯一的\_id ObjectId 字段

来标识。fs.files 用作父级文档。fs.chunks fs.chunks 文档中的files\_id files\_id 字段将块连接到父级文档上。

fs.files 集合的一个范例文档如下所示：

|  |
| --- |
| { |

|  |
| --- |
| "filename": "test.txt", |

|  |
| --- |
| "chunkSize": NumberInt(261120), |

|  |
| --- |
| "uploadDate": ISODate("2014-04-13T11:32:33.557Z"), |

|  |
| --- |
| "md5": "7b762939321e146569b07f72c62cca4f", |

|  |
| --- |
| "length": NumberInt(646) |

|  |
| --- |
| } |

该文档指定了文件名、块大小、上传日期以及长度。

下面是 fs.chunks 文档的一个范例文档：

|  |
| --- |
| { |

|  |
| --- |
| "files\_id": ObjectId("534a75d19f54bfec8a2fe44b"), |

|  |
| --- |
| "n": NumberInt(0), |

|  |
| --- |
| "data": "Mongo Binary Data" |

|  |
| --- |
| } |

为 GridFS 添加文件

|  |
| --- |
| 下面我们将使用put put 命令，利用 GridFS 存储一个 mp3 文件。该例中，我们将使用 MongoDB 的 bin 文件夹下  的mongofiles.exe mongofiles.exe 工具。 |

打开命令行提示符，浏览至 mongofiles.exe，输入下列代码：

第 2 章 高级教程 | 78

|  |  |
| --- | --- |
| |  | | --- | | >mongofiles.exe -d gridfs put song.mp3 | |
| 上述代码中，gridfs gridfs 是保存文件所在的数据库名称。如果数据库不存在，MongoDB 将自动创建一个新数据 |

库，Song.mp3 是上传文件名。要想查看数据库中文件的文档，使用find() 查询：

|  |
| --- |
| >db.fs.files.find() |

返回结果文档如下所示：

|  |
| --- |
| { |

|  |
| --- |
| \_id: ObjectId('534a811bf8b4aa4d33fdf94d'), |

|  |
| --- |
| filename: "song.mp3", |

|  |
| --- |
| chunkSize: 261120, |

|  |
| --- |
| uploadDate: new Date(1397391643474), md5: "e4f53379c909f7bed2e9d631e15c1c41", |

|  |
| --- |
| length: 10401959 |

|  |
| --- |
| }  我们还可以查看fs.chunks fs.chunks 集合中所有与存储文件相关的块。代码如下，其中使用了上次查询所返回的文档 id。 |

|  |
| --- |
| >db.fs.chunks.find({files\_id:ObjectId('534a811bf8b4aa4d33fdf94d')}) |

该例中，查询返回了 40 个文档，这就是说整个 Mp3 文件被分解成了 40 个数据块。

第 2 章 高级教程 | 79

固定集合

固定集合（Capped Collection）是一种尺寸固定的“循环”集合，可提供高效的创建、读取、删除等操作。这 里所指的“循环”的意思是，当分配给集合的文件尺寸耗尽时，就会自动开始删除最初的文档，不需要提供任何 显式的指令。

如果文档更新后增加了文档的尺寸，那么固定集合会限制对文档的更新。因为固定集合按照磁盘存储的顺序来保 存文档，所以能确保文档尺寸不会增加磁盘分配的尺寸。固定集合最适合保存日志信息，缓存数据以及任何其他 大容量数据。

创建固定集合

要想创建固定集合，需要使用createCollection createCollection 命令，并将capped capped 选项设为true true，同时还需要指定集合的最 大尺寸（以字节计）。

|  |  |
| --- | --- |
| |  | | --- | | >db.createCollection("cappedLogCollection",{capped:true,size:10000}) | |
| 除了集合尺寸外，还可以使用max max 参数限制集合中的文档最大数量。 |

|  |  |
| --- | --- |
| |  | | --- | | >db.createCollection("cappedLogCollection",{capped:true,size:10000,max:1000}) | |
| 如果想要检查集合是否固定，使用isCapped isCapped 命令即可。 |

|  |
| --- |
| >db.cappedLogCollection.isCapped() |

如想将现有集合转化为固定集合，使用下列代码：

|  |  |
| --- | --- |
| |  | | --- | | >db.runCommand({"convertToCapped":"posts",size:10000}) | |
| 上述代码会将现有的posts posts 集合转化为固定集合。 |

查询固定集合

默认情况下，利用 find 查询固定集合，结果会按照插入顺序进行显示。但如果想按相反顺序获取文档，可以使用 sort 命令，如下所示：

|  |
| --- |
| >db.cappedLogCollection.find().sort({$natural:-1}) |

关于固定集合，有以下几个非常值得注意的要点：

第 2 章 高级教程 | 80

|  |  |
| --- | --- |
| • • • | 无法从固定集合中删除文档。  固定集合没有默认索引，甚至在 \_id 字段中也没有。  在插入新的文档时，MongoDB 并不需要寻找磁盘空间来容纳新文档。它只是盲目地将新文档插入到集合末 |

尾。这使得固定集合中的插入操作是非常快速的。

|  |  |
| --- | --- |
| • | 同样的，在读取文档时，MongoDB 会按照插入磁盘的顺序来读取文档，从而使读取操作也非常快。 |

第 2 章 高级教程 | 81

自动增长

MongoDB 没有提供类似 SQL 数据库所具有的自动增长功能（auto-increment）。默认情况下，MongoDB

将\_id \_id 字段（使用 12 字节的 ObjectId）来作为文档的唯一标识。但在有些情况下，我们希望 \_id 字段值能够自

动增长，而不是固守在 ObjectId 值上。

由于这不是 MongoDB 的默认功能，所以我们按照 MongoDB 文档所建议的方式，使用counters counters 集合来程序

化地实现该功能。

|  |
| --- |
| 使用counters counters 集合  假设存在下列文档products products，我们希望 \_id 字段值是一个能够自动增长的整数序列 能够自动增长的整数序列（1、2、3、4 …… n）。 |

|  |
| --- |
| { |

|  |
| --- |
| "\_id":1, |

|  |
| --- |
| "product\_name": "Apple iPhone", |

|  |
| --- |
| "category": "mobiles" |

|  |
| --- |
| } |

创建一个counters 集合，其中包含了所有序列字段最后的序列值。

|  |  |
| --- | --- |
| |  | | --- | | >db.createCollection("counters") | |
| 现在，将下列文档（productid productid 是它的键）插入到counters 集合中： |

|  |
| --- |
| { |

|  |
| --- |
| "\_id":"productid", |

|  |
| --- |
| "sequence\_value": 0 |

|  |
| --- |
| } |

sequence\_value 字段保存了序列的最后值。

使用下列命令将序列文档插入到counters 集合中。

|  |  |
| --- | --- |
| |  | | --- | | >db.counters.insert({\_id:"productid",sequence\_value:0}) | |

第 2 章 高级教程 | 82

创建 JavaScript 函数

接着，我们创建一个getNextSequenceValue 函数，该函数以序列名为输入，按照 1 的幅度增加序列数，返回

更新的序列数。在该例中，序列名称为productid productid。

|  |
| --- |
| >function getNextSequenceValue(sequenceName){ |

|  |
| --- |
| var sequenceDocument = db.counters.findAndModify( |

|  |
| --- |
| { |

|  |
| --- |
| query:{\_id: sequenceName }, |

|  |
| --- |
| update: {$inc:{sequence\_value:1}}, |

|  |
| --- |
| new:true |

|  |
| --- |
| }); |

|  |
| --- |
| return sequenceDocument.sequence\_value; |

|  |
| --- |
| } |

使用 JavaScript 函数

下面，在创建新文档以及将返回的序列值赋予文档的 \_id 字段过程中，我们使用getNextSequenceValue 函

数。

使用下列代码插入两个范例文档：

|  |
| --- |
| >db.products.insert({ |

|  |
| --- |
| "\_id":getNextSequenceValue("productid"), |

|  |
| --- |
| "product\_name":"Apple iPhone", |

|  |
| --- |
| "category":"mobiles"}) |

|  |
| --- |
| >db.products.insert({ |

|  |
| --- |
| "\_id":getNextSequenceValue("productid"), |

|  |
| --- |
| "product\_name":"Samsung S3", |

|  |
| --- |
| "category":"mobiles"}) |

如上所示，使用getNextSequenceValue 函数为 \_id 字段设定值。

为了验证该功能，使用 find 命令来获取文档：

|  |
| --- |
| >db.prodcuts.find() |

在上面的查询返回的结果文档中，\_id 字段值果然是自动增长的：

第 2 章 高级教程 | 83

|  |
| --- |
| { "\_id" : 1, "product\_name" : "Apple iPhone", "category" : "mobiles"} |

|  |
| --- |
| { "\_id" : 2, "product\_name" : "Samsung S3", "category" : "mobiles" } |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| 更多信息请访问 |  |

|  |
| --- |
| http://wiki.jikexueyuan.com/project/mongodb/ |