# MongoDB基本概念

【<https://www.runoob.com/mongodb/mongodb-databases-documents-collections.html>】

## 术语比较：

|  |  |  |
| --- | --- | --- |
| SQL术语概念 | MongoDB术语概念 | 解释说明 |
| database | databse | 数据库 |
| table | collection | 数据库表/集合 |
| row | document | 数据记录行/文档 |
| column | field | 数据字段/域 |
| index | index | 索引 |
| table joins |  | 表连接,MongoDB不支持(MongoDB推荐使用内嵌文档的方式) |
| primary key | primary key | 主键,MongoDB自动将\_id字段设置为主键 |

通过下面实例，我们可以更加直观的了解MongoDB中的一些概念



## 数据库：

一个MongoDB实例(一个MongoDB实例就是运行了一台MongoDB服务器)中可以建立多个数据库。

MongoDB的单个实例可以容纳多个独立的数据库，每一个都有自己的集合和权限，不同的数据库也放置在不同的文件中。

数据库也通过名字来标识，数据库名可以是满足以下条件的任意UTF-8字符串：

* 不能是空串；
* 应全部小写；
* 最多64字节；

有一些数据库名是保留的，可以直接访问这些有特殊作用的数据库：

* admin：从权限的角度来看，这是“root”数据库。要是将一个用户添加到这个数据库，这个用户自动继承所有数据库的权限。一些特定的服务器端命令也只能从这个数据库运行，比如列出所有的数据库或者关闭服务器。
* local：这个数据库永远不会被复制，可以用来存储限于本地单台服务器的任意集合。
* config：当Mongo用于分片设置时，config数据库在内部使用，用于保存分片的相关信息。

要记住一点，数据库最终会变成文件系统里的文件，而数据库名就是相应的文件名，这是数据库名有如此多限制的原因。

## 集合：

集合即使MongoDB文档组，类似于RDBMS中的表格。

集合存在于数据库中，集合没有固定的结构，这意味着你可以对集合插入不同格式和类型的数据，但通常情况下我们插入集合的数据都会有一定的关联性。

比如，我们可以将以下不同数据结构的文档插入到集合中：

|  |
| --- |
| {"site":"www.baidu.com"}  {"site":"www.google.com","name":"Google"}  {"site":"www.runoob.com","name":"菜鸟教程","num":5} |

**当第一个文档被插入时，集合就会被创建。**

### 1、合法的集合名：

* 集合名不能是空字符串；
* 集合名不能含有\0字符(空字符)，这个字符表示集合名的结尾；
* 集合名不能以“system.”开头，这是为系统集合保留的前缀；
* 用户创建的集合名不能含有保留字符，有些驱动程序的确支持在集合名里面包含，这是因为某些系统生成的集合中包含该字符。除非你要访问这种系统创建的集合，否则千万不要在名字里出现$。

### capped collections：

Capped collections就是固定大小的collection。

它有很高的性能以及队列过期的特性（过期按照插入的顺序），有点和“RRD”概念类似。

Capped collections是高性能自动的维护对象的插入顺序，它非常适合类似记录日志的功能，和标准的collection不同，你必须要显式的创建一个capped collection，指定一个collection的大小，单位是字节。collection中的数据存储空间值是提前分配的。

Capped collections可以按照文档的插入顺序保存到集合中，而且这些文档在磁盘上存放位置也是按照插入顺序来保存的，所以当我们更新capped collections中的文档的时候，更新后的文档不可以超过之前文档的大小，这样的话就可以确保所有文档在磁盘上的位置一直不变。

由于Capped Collections是按照文档的插入顺序而不是使用索引确定插入位置，这样的话可以提高增添数据的效率。MongoDB的操作日志文件oplog.rs就是利用Capped collections来实现的。

|  |
| --- |
| db.createCollection("mycoll", {capped:true, size:100000}) |

* 在capped collection中，你能添加新的对象。
* 能进行更新，然而，对象不会增加存储空间。如果增加，刚更新就会失败。
* 使用capped collection不能删除一个文档，可以使用drop()方法删除collection所有的文档对象。
* 删除之后，你必须显式的重新创建这个collection。
* 在32位的机器中，capped collection最大存储为1e9个字节。

### 3、动态模式：

集合是动态模式的，这意味着一个集合里面的文档可以是各式各样的。

### 4、子集合：

组织集合的一种惯例是使用“.” 分隔不同命名空间的子集合。

例如，一个具有博客功能的应用可能包含两个集合，分别是“blog.posts”和“blog.authors”。这是为了使组织结构更清晰，这里的blog集合（这个集合甚至不需要存在）跟它的子集合没有任何关系。

虽然子集合没有任何特别的属性，但它们却非常有用，因而很多MongoDB 工具都

使用了子集合。

* GridFS (— 种用于存储大文件的协议）使用子集合来存储文件的元数据，这样就

可以与文件内容块很好地隔离开来。（第6 章会详细介绍GridFS）

* 数驱动程序都提供了一些语法糖，用于访问指定集合的子集合。例如，在数

据库shell中，db \_blog 代表blog 集合，而db.blog.posts 代表blog.posts 集合。

在MongoDB中，使用子集合来组织数据非常高效，值得推荐。

## 文档：

文档是一组键值对，MongoDB的文档不需要设置相同的字段，并且相同的字段不需要相同的数据类型，这与关系型数据库有很大的区别，也是MongoDB非常突出的特点。

### 注意点：

1. 文档中的键/值对是有序的；
2. 文档中的值不仅可以是基本类型，也可以是嵌入的文档；
3. MongoDB区分类型和大小写；
4. 文档的键是字符串，除少数例外情况，键可以使用任意UTF-8字符；

### 文档键命令规范：

1. 键不能含有\0(空字符)，这个字符用来表示键的结尾；
2. . 和 $ 有特别的含义，只有在特定环境下才能使用；
3. 以下划线“\_”开头的键是保留的（不是严格要求的）；

## 元数据：

数据库的信息是存储在“特定”的集合中，它们使用了系统的命名空间：

|  |
| --- |
| dbname.system.\* |

在MongoDB数据库中名字空间<dbname>.system.\*是包含多种系统信息的特殊集合（Collection），如下：

|  |  |
| --- | --- |
| 集合命名空间 | 描述 |
| dbname.system.namespaces | 列出所有名字空间 |
| dbname.system.indexes | 列出所有索引 |
| dbname.system.profile | 包含数据库概要(profile)信息 |
| dbname.system.users | 列出所有可访问数据库的用户 |
| dbname.local.sources | 包含复制对端(slave)的服务器信息和状态 |

对于修改系统集合中的对象有如下限制：

在{{system.indexes}}插入数据，可以创建索引。但除此之外该表信息是不可改变的（特殊的drop index命令将自动更新相关信息）。

{{system.users}}是可修改的

{{system.profile}}是可删除的

## MongoDB数据类型：

下表为MongoDB中常用的几种数据类型：

|  |  |
| --- | --- |
| 数据类型 | 描述 |
| 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,根据你所采用的的服务器,可分为32位和64位。 |
| Boolean | 布尔值。用于存储布尔值。 |
| 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 |
| Array | 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 |

下面说明下几种重要的数据类型：

### 1、ObjectId：

ObjectId类似唯一主键，可以很快的去生成和排序，包含12bytes，含义是：

* 前4个字节表示创建unix时间戳，格林尼治时间UTC时间，比北京时间晚了8个小时。
* 接下来的3个字节是机器标识码。
* 紧接的2个字节由进程Id组成PID。
* 最后3个字节是随机数。



MongoDB中存储的文档必须有一个\_id键，这个键的值可以是任何类型的，默认是个ObjectId对象。

由于ObjectId中保存了创建的时间戳，所以你需要为你的文档保存时间戳字段，你可以通过getTimestamp函数来获取文档的创建时间。

|  |
| --- |
| var objectId = ObjectId("5edf96daecb4490f9498d76a");  objectId.getTimestamp(); |

ObjectId转为字符串

|  |
| --- |
| var objectId = ObjectId("5edf96daecb4490f9498d76a");  objectId.getTimestamp();  objectId.str |

### 2、字符串：

BSON字符串都是UTF-8编码

### 时间戳：

BSON有一个特殊的时间戳类型用于MongoDB内部使用，与普通的日期类型不相关。时间戳值是一个64位的值，其中：

* 前32位是一个time\_t值（与Unix新纪元相差的秒数）。
* 后32位是在某秒钟操作的一个递增的序数。

在单个MongoDB实例中，时间戳值通常是唯一的。

在复制集中，oplog有一个ts字段，这个字段中的值使用BSON时间戳表示了操作时间。

### 日期：

表示当前距离Unix新纪元（1970年1月1日）的毫秒数，日期类型是有符号的，负数表示1970年之前的日期。

|  |
| --- |
| > var mydate1 = new Date() //格林尼治时间  > mydate1  ISODate("2018-03-04T14:58:51.233Z")  > typeof mydate1  object |

|  |
| --- |
| > var mydate2 = ISODate() //格林尼治时间  > mydate2  ISODate("2018-03-04T15:00:45.479Z")  > typeof mydate2  object |

这样创建的时间是日期类型，可以使用JS中的Date类型的方法。

返回一个时间类型的字符串：

|  |
| --- |
| > var mydate1str = mydate1.toString()  > mydate1str  Sun Mar 04 2018 14:58:51 GMT+0000 (UTC)  > typeof mydate1str  string |

或者

|  |
| --- |
| > Date()  Sun Mar 04 2018 15:02:59 GMT+0000 (UTC) |

## 七、命名空间：

把数据库名添加到集合名前，得到集合的完全限定名，即命名空间（namespace）。

例如，如果要使用cms数据库中的blog.posts集合，这个集合的命名空间就是cms.blog.posts。

命名空间的长度不得超过121字节，且在实际使用中应小于100字节。