目录

[NoSQL 简介 1](#_Toc521342470)

[MongoDB 概念解析 3](#_Toc521342471)

[数据库 4](#_Toc521342472)

[文档 4](#_Toc521342473)

[集合 4](#_Toc521342474)

[MongoDB 数据类型 5](#_Toc521342475)

[MongoDB使用 6](#_Toc521342476)

[MongoDB 连接 6](#_Toc521342477)

[MongoDB 创建数据库 7](#_Toc521342478)

[MongoDB 删除数据库 7](#_Toc521342479)

[MongoDB 创建集合 7](#_Toc521342480)

[MongoDB 删除集合 7](#_Toc521342481)

[MongoDB 插入文档 7](#_Toc521342482)

[MongoDB 查询文档 8](#_Toc521342483)

[MongoDB 更新文档 9](#_Toc521342484)

# NoSQL 简介

NoSQL，指的是非关系型的数据库。NoSQL有时也称作Not Only SQL的缩写，是对不同于传统的关系型数据库的数据库管理系统的统称。

NoSQL用于超大规模数据的存储（例如谷歌或Facebook每天为他们的用户收集万亿比特的数据）。这些类型的数据存储不需要固定的模式，无需多余操作就可以横向扩展。

分布式计算的优点

可靠性（容错） ：

分布式计算系统中的一个重要的优点是可靠性。一台服务器的系统崩溃并不影响到其余的服务器。

可扩展性：

在分布式计算系统可以根据需要增加更多的机器。

资源共享：

共享数据是必不可少的应用，如银行，预订系统。

灵活性：

由于该系统是非常灵活的，它很容易安装，实施和调试新的服务。

更快的速度：

分布式计算系统可以有多台计算机的计算能力，使得它比其他系统有更快的处理速度。

开放系统：

由于它是开放的系统，本地或者远程都可以访问到该服务。

更高的性能：

相较于集中式计算机网络集群可以提供更高的性能（及更好的性价比）。

分布式计算的缺点

故障排除：

故障排除和诊断问题。

软件：

更少的软件支持是分布式计算系统的主要缺点。

网络：

网络基础设施的问题，包括：传输问题，高负载，信息丢失等。

安全性：

开放系统的特性让分布式计算系统存在着数据的安全性和共享的风险等问题。

RDBMS vs NoSQL

RDBMS   
- 高度组织化结构化数据   
- 结构化查询语言（SQL） (SQL)   
- 数据和关系都存储在单独的表中。   
- 数据操纵语言，数据定义语言   
- 严格的一致性  
- 基础事务

NoSQL   
- 代表着不仅仅是SQL  
- 没有声明性查询语言  
- 没有预定义的模式  
-键 - 值对存储，列存储，文档存储，图形数据库  
- 最终一致性，而非ACID属性  
- 非结构化和不可预知的数据  
- CAP定理   
- 高性能，高可用性和可伸缩性

CAP定理（CAP theorem）

在计算机科学中, CAP定理（CAP theorem）, 又被称作 布鲁尔定理（Brewer's theorem）, 它指出对于一个分布式计算系统来说，不可能同时满足以下三点:

* 一致性(Consistency) (所有节点在同一时间具有相同的数据)
* 可用性(Availability) (保证每个请求不管成功或者失败都有响应)
* 分隔容忍(Partition tolerance) (系统中任意信息的丢失或失败不会影响系统的继续运作)

CAP理论的核心是：一个分布式系统不可能同时很好的满足一致性，可用性和分区容错性这三个需求，最多只能同时较好的满足两个。

因此，根据 CAP 原理将 NoSQL 数据库分成了满足 CA 原则、满足 CP 原则和满足 AP 原则三 大类：

* CA - 单点集群，满足一致性，可用性的系统，通常在可扩展性上不太强大。
* CP - 满足一致性，分区容忍性的系统，通常性能不是特别高。
* AP - 满足可用性，分区容忍性的系统，通常可能对一致性要求低一些。

BASE

BASE：Basically Available, Soft-state, Eventually Consistent。 由 Eric Brewer 定义。

CAP理论的核心是：一个分布式系统不可能同时很好的满足一致性，可用性和分区容错性这三个需求，最多只能同时较好的满足两个。

BASE是NoSQL数据库通常对可用性及一致性的弱要求原则:

Basically Availble --基本可用

Soft-state --软状态/柔性事务。 "Soft state" 可以理解为"无连接"的, 而 "Hard state" 是"面向连接"的

Eventual Consistency -- 最终一致性， 也是是 ACID 的最终目的。

NoSQL 数据库分类

|  |  |  |
| --- | --- | --- |
| 类型 | 部分代表 | 特点 |
| 列存储 | Hbase  Cassandra  Hypertable | 顾名思义，是按列存储数据的。最大的特点是方便存储结构化和半结构化数据，方便做数据压缩，对针对某一列或者某几列的查询有非常大的IO优势。 |
| 文档存储 | MongoDB  CouchDB | 文档存储一般用类似json的格式存储，存储的内容是文档型的。这样也就有有机会对某些字段建立索引，实现关系数据库的某些功能。 |
| key-value  存储 | Tokyo Cabinet / Tyrant  Berkeley DB  MemcacheDB  Redis | 可以通过key快速查询到其value。一般来说，存储不管value的格式，照单全收。（Redis包含了其他功能） |
| 图存储 | Neo4J  FlockDB | 图形关系的最佳存储。使用传统关系数据库来解决的话性能低下，而且设计使用不方便。 |
| 对象存储 | db4o  Versant | 通过类似面向对象语言的语法操作数据库，通过对象的方式存取数据。 |
| xml数据库 | Berkeley DB XML  BaseX | 高效的存储XML数据，并支持XML的内部查询语法，比如XQuery,Xpath。 |

# MongoDB 概念解析

MongoDB 是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统。在高负载的情况下，添加更多的节点，可以保证服务器性能。MongoDB 旨在为WEB应用提供可扩展的高性能数据存储解决方案。

MongoDB 将数据存储为一个文档，数据结构由键值(key=>value)对组成。MongoDB 文档类似于 JSON 对象。字段值value可以包含其他文档，列表及文档列表。列表与python列表格式相似。

如下为一个MongoDB的集合示例：

{"site":"www.baidu.com"}

{"site":"www.google.com","name":"Google"}

{"site":"www.runoob.com","name":"菜鸟教程","num":5}

SQL vs MongoDB

|  |  |  |
| --- | --- | --- |
| SQL术语/概念 | MongoDB术语/概念 | 解释/说明 |
| database | database | 数据库 |
| table | collection | 数据库表/集合 |
| row | document | 数据记录行/文档 |
| column | field | 数据字段/域 |
| index | index | 索引 |
| table joins |  | 表连接,MongoDB不支持 |
| primary key | primary key | 主键,MongoDB自动将\_id字段设置为主键 |

## 数据库

一个mongodb中可以建立多个数据库。

MongoDB的默认数据库为"db"，该数据库存储在data目录中。

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。

·"show dbs" 命令可以显示所有数据的列表。

·执行 "db" 命令可以显示当前数据库对象或集合。

·运行"use"命令，可以连接到一个指定的数据库。

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。

* 不能是空字符串（"")。
* 不得含有' '（空格)、.、$、/、\和\0 (空字符)。
* 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。

## 文档

文档是一组键值(key-value)对(即BSON)。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。

需要注意的是：

1. 文档中的键/值对是有序的。
2. 文档中的值不仅可以是在双引号里面的字符串，还可以是其他几种数据类型（甚至可以是整个嵌入的文档)。
3. MongoDB区分类型和大小写。
4. MongoDB的文档不能有重复的键。
5. 文档的键是字符串。除了少数例外情况，键可以使用任意UTF-8字符。

文档键命名规范：

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

## 集合

集合就是 MongoDB 文档组，类似于 RDBMS （关系数据库管理系统：Relational Database Management System)中的表格。

集合存在于数据库中，集合没有固定的结构，这意味着你在对集合可以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。当第一个文档插入时，集合就会被创建。

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。
* 用户创建的集合名字不能含有保留字符。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的数据存储空间值提前分配的。

要注意的是指定的存储大小包含了数据库的头信息。

* 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，更新就会失败 。
* 数据库不允许进行删除。使用drop()方法删除collection所有的行。
* 注意: 删除之后，你必须显式的重新创建这个collection。
* 在32bit机器中，capped collection最大存储为1e9( 1X109)个字节。

元数据

数据库的信息是存储在集合中。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。 |
| 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。 |

# MongoDB使用

## MongoDB 连接

标准 URI 连接语法：

mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[database][?options]]

* mongodb:// 这是固定的格式，必须要指定。
* username:password@ 可选项，如果设置，在连接数据库服务器之后，驱动都会尝试登陆这个数据库
* host1 必须的指定至少一个host, host1 是这个URI唯一要填写的。它指定了要连接服务器的地址。如果要连接复制集，请指定多个主机地址。
* portX 可选的指定端口，如果不填，默认为27017
* /database 如果指定username:password@，连接并验证登陆指定数据库。若不指定，默认打开 test 数据库。
* ?options 是连接选项。如果不使用/database，则前面需要加上/。所有连接选项都是键值对name=value，键值对之间通过&或;（分号）隔开

标准的连接格式包含了多个选项(options)，如下所示：

|  |  |
| --- | --- |
| 选项 | 描述 |
| replicaSet=name | 验证replica set的名称。 Impliesconnect=replicaSet. |
| slaveOk=true|false | true:在connect=direct模式下，驱动会连接第一台机器，即使这台服务器不是主。在connect=replicaSet模式下，驱动会发送所有的写请求到主并且把读取操作分布在其他从服务器。  false: 在 connect=direct模式下，驱动会自动找寻主服务器. 在connect=replicaSet 模式下，驱动仅仅连接主服务器，并且所有的读写命令都连接到主服务器。 |
| safe=true|false | true: 在执行更新操作之后，驱动都会发送getLastError命令来确保更新成功。(还要参考 wtimeoutMS).  false: 在每次更新之后，驱动不会发送getLastError来确保更新成功。 |
| w=n | 驱动添加 { w : n } 到getLastError命令. 应用于safe=true。 |
| wtimeoutMS=ms | 驱动添加 { wtimeout : ms } 到 getlasterror 命令. 应用于 safe=true. |
| fsync=true|false | true: 驱动添加 { fsync : true } 到 getlasterror 命令.应用于 safe=true.  false: 驱动不会添加到getLastError命令中。 |
| journal=true|false | 如果设置为 true, 同步到 journal (在提交到数据库前写入到实体中). 应用于 safe=true |
| connectTimeoutMS=ms | 可以打开连接的时间。 |
| socketTimeoutMS=ms | 发送和接受sockets的时间。 |

## MongoDB 创建数据库

MongoDB 创建数据库的语法格式如下：

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。

MongoDB 中默认的数据库为 test，如果你没有创建新的数据库，集合将存放在 test 数据库中。

## MongoDB 删除数据库

MongoDB 删除数据库的语法格式如下：

db.dropDatabase()

删除当前数据库，默认为 test，你可以使用 db 命令查看当前数据库名。

## MongoDB 创建集合

MongoDB 中使用 **createCollection()** 方法来创建集合。

语法格式：

db.createCollection(name, options)

参数说明：

* 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

options 可以是如下参数：

|  |  |  |
| --- | --- | --- |
| 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。 |
| autoIndexId | 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。

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。

## MongoDB 删除集合

集合删除语法格式如下：

db.COLLECTION\_NAME.drop()

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

## MongoDB 插入文档

文档的数据结构和JSON基本一样。

所有存储在集合中的数据都是BSON格式。

BSON是一种类json的一种二进制形式的存储格式,简称Binary JSON。

MongoDB使用insert()或save()方法向集合中插入文档，语法如下：

db.COLLECTION\_NAME.insert(document)

document的书写格式类似于python中的字典，如：

{title: 'MongoDB 教程',

description: 'MongoDB 是一个 Nosql 数据库',

by: '菜鸟教程',

url: 'http://www.runoob.com',

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

likes: 100}

也可以将一个变量赋予想要创建的文档值，然后将该（字典）变量作为insert参数。

插入文档你也可以使用 db.COLLECTION\_NAME.save(document)命令。如果不指定\_id字段save()方法类似于insert()方法。如果指定\_id字段，则会更新该\_id的数据。

## MongoDB 查询文档

MongoDB 查询文档使用 find() 方法。

find() 方法以非结构化的方式来显示所有文档。

语法

MongoDB 查询数据的语法格式如下：

db.collection.find(query, projection)

* query ：可选，使用查询操作符指定查询条件
* projection ：可选，使用投影操作符指定返回的键。查询时返回文档中所有键值， 只需省略该参数即可（默认省略）。

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

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

pretty() 方法以格式化的方式来显示所有文档。

MongoDB 与 RDBMS Where 语句比较

如果你熟悉常规的 SQL 数据，通过下表可以更好的理解 MongoDB 的条件语句查询：

|  |  |  |  |
| --- | --- | --- | --- |
| 操作 | 格式 | 范例 | RDBMS中的类似语句 |
| 等于 | {<key>:<value>} | db.col.find({"by":"菜鸟教程"}).pretty() | where by = '菜鸟教程' |
| 小于 | {<key>:{$lt:<value>}} | db.col.find({"likes":{$lt:50}}).pretty() | where likes < 50 |
| 小于或  等于 | {<key>:{$lte:<value>}} | db.col.find({"likes":{$lte:50}}).pretty() | where likes <= 50 |
| 大于 | {<key>:{$gt:<value>}} | db.col.find({"likes":{$gt:50}}).pretty() | where likes > 50 |
| 大于或  等于 | {<key>:{$gte:<value>}} | db.col.find({"likes":{$gte:50}}).pretty() | where likes >= 50 |
| 不等于 | {<key>:{$ne:<value>}} | db.col.find({"likes":{$ne:50}}).pretty() | where likes != 50 |

MongoDB AND 条件

MongoDB 的 find() 方法可以传入多个键(key)，每个键(key)以逗号隔开，即常规 SQL 的 AND 条件。

语法格式如下：

>db.col.find({key1:value1, key2:value2}).pretty()

MongoDB OR 条件

MongoDB OR 条件语句使用了关键字 $or,语法格式如下：

>db.col.find(

{

$or: [

     {key1: value1}, {key2:value2}

]

}

).pretty()

## MongoDB 更新文档