* MongoDB是由C++语言编写的，是一个基于**分布式文件存储**的数据库。**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是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。
* **CAP定理**（布鲁尔定理）：对于一个分布式计算系统来说，不可能同时满足以下三点：（1）**一致性（Consistency）**，所有节点在同一时间具有相同的数。（2）**可用性（Availability）**，保证每个请求不管成功或者失败都有响应。（3）**分区容错性（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地满足一致性、可用性和分区容错性这三个需求，最多只能同时较好地满足两个**。
* 根据CAP原理将NoSQL分成了满足CA原则、满足CP原则和满足AP原则三大类：（1）**CA**，单点集群，满足一致性，可用性，通常在可扩展性上不太强大。（2）**CP**，满足一致性，分区容错性，通常性能不是特别高。（3）**AP**，满足可用性，分区容错性的系统，通常可能对一致性要求低一些。
* **BASE**：**Basically Available**，**Soft-state**，**Eventually Consistent**。BASE是NoSQL数据库通常对可用性及一致性的弱要求原则：（1）**Basically Available**，基本可用。（2）**Soft-State**，软状态/柔性事务。“Soft-state”可用理解为“无连接”的，而“Hard-state”是“面向连接”的。（3）**Eventual Consistency**，最终一致性，也是ACID的最终目的。
* MongoDB是**文档存储**型NoSQL。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。这样也就有机会对某些字段建立**索引**，实现关系数据库的某些功能。



* 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。字段值可以包含其他文档，数组及文档数组。

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



一个mongodb中可以建立多个独立的数据库。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。

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。

1. admin：从权限的角度来看，这时“root”数据库。要是将一根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，臂如列出所有数据库或者关闭服务器。
2. 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。
3. config：当MongoDB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**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。一个简单的文档例子如下：

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

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。

文档命名规范：

1. 键不能含有\0（空字符）。这个字符用来表示键的结尾。
2. .和$有特别的意义，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）。

**集合**是MongoDB文档组，类似于RDBMS（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**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有一定的关联性。臂如，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}

当第一个文档插入时，集合就会被创建。

集合命名规范：

1. 集合名不能是空字符串“”。
2. 集合名不能含有\0字符，这个字符表示集合名的结尾。
3. 集合名不能以“system.”开头，这是为系统集合保留的前缀。
4. 用户创建的集合名字不能含有保留字符。

**Capped collections**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）。Capped collections能高性能**自动地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不能，你必须要显式地创建一个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，这样就可以确保所有文档在磁盘上的位置一直保持不变**。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Capped collections来实现的。

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

在Capped collections中，你能添加新的对象。

能进行更新，然而，对象不能增加存储空间。如果增加，更新就会失败。

使用Capped collections不能删除一个文档，可以使用drop()删除collection所有的行。

删除之后，你必须显式地重新创建这个collection。

在32位机器中，capped collection最大存储为1e9个字节。