# 基本介绍

## 分布式系统

1. 分布式系统是由一组通过网络进行通信、为了完成共同的任务而协调工作的计算机节点组成的系统。
2. 分布式系统是建立在网络之上的软件系统。正是因为软件的特性，所以分布式系统具有高度的内聚性和透明性。因此，网络和分布式系统之间的区别更多的在于高层软件（特别是操作系统），而不是硬件。

## 分布式计算的优点

1. 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。
2. 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。
3. 资源共享：共享数据是必不可少的应用，如银行，预订系统。
4. 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。
5. 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。
6. 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。
7. 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

1. 故障排除：故障排除和诊断问题。
2. 软件：更少的软件支持是分布式计算系统的主要缺点。
3. 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。
4. 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。

## NoSQL介绍

1. 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。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。**

## CAP定理

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。

1. 一致性 (Consistency) (所有节点在同一时间具有相同的数据)
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)

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：

1. 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。
2. CP - 满足一致性，分区容忍性的系统，通常性能不是特别高。
3. 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。

## NoSQL数据库的BASE原则

ACID是针对关系型数据库，BASE是针对非关系型数据库



## NoSQL数据库的分类



# MongoDB简介

## 基本介绍

* 1. 概述

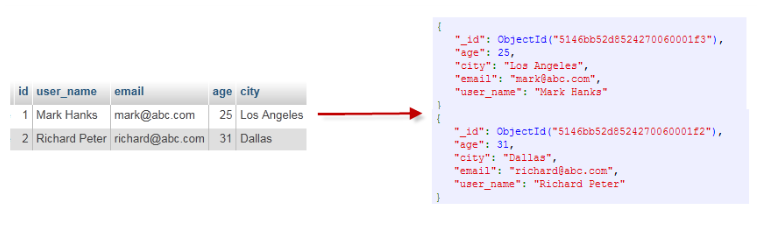
1. MongoDB 是由C++语言编写的，是一个基于**分布式文件存储**的开源数据库系统。
2. 在高负载的情况下，添加更多的节点，可以保证服务器性能。
3. MongoDB 旨在为WEB应用提供可扩展的高性能数据存储解决方案。
4. MongoDB **将数据存储为一个文档**，数据结构由键值(key=>value)对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档，数组及文档数组。
   1. 特点
5. MongoDB 是一个面向文档存储的数据库，操作起来比较简单和容易。
6. 你可以在MongoDB记录中设置任何属性的索引 (如：FirstName="Sameer",Address="8 Gandhi Road")来实现更快的排序。
7. 你可以通过本地或者网络创建数据镜像，这使得MongoDB有更强的扩展性。
8. 如果负载的增加（需要更多的存储空间和更强的处理能力） ，它可以分布在计算机网络中的其他节点上这就是所谓的分片。
9. Mongo支持丰富的查询表达式。查询指令使用JSON形式的标记，可轻易查询文档中内嵌的对象及数组。
10. **MongoDb 使用update()命令可以实现替换完成的文档（数据）或者一些指定的数据字段 。**
11. Mongodb中的**Map/reduce**主要是用来对数据进行批量处理和聚合操作。
12. Map和Reduce。Map函数调用emit(key,value)遍历集合中所有的记录，将key与value传给Reduce函数进行处理。
13. Map函数和Reduce函数是使用Javascript编写的，并可以通过db.runCommand或mapreduce命令来执行MapReduce操作。
14. **GridFS是MongoDB中的一个内置功能，可以用于存放大量小文件。**
15. MongoDB允许在服务端执行脚本，可以用Javascript编写某个函数，直接在服务端执行，也可以把函数的定义存储在服务端，下次直接调用即可。
16. MongoDB支持各种编程语言:RUBY，PYTHON，JAVA，C++，PHP，C#等多种语言。
17. MongoDB安装简单。

## 基本概念/结构

* 1. MongoDB和MySQL概念对比



* 1. 实例图



## 分类架构

* 1. 数据库

1. 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。
2. 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。

1. 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。

* 1. 集合

1. 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。
2. 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。
3. 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$。

* + 1. Capped Collections

1. Capped collections 就是**固定大小的collection**。
2. 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。
3. 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。
4. 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，这样话就可以确保所有文档在磁盘上的位置一直保持不变。
5. 由于 Capped collection 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 Collection 来实现的。
6. 要注意的是指定的存储大小包含了数据库的头信息。
7. 创建语句

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，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 所有的行。**

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。

* 1. 文档

1. **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。

3.2.1 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)。

3.2.2 注意要点

(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。

* 1. 元数据

1. 一组自带的数据库，使用命名空间：dbname.system.\*
2. 如下：



1. 修改集合的限制

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



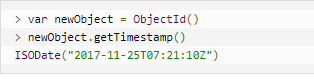
* 1. ObjectId
     1. ObjectId 类似唯一主键，可以很快的去生成和排序，包含 12 bytes，含义是：

1. 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
2. 接下来的 3 个字节是机器标识码
3. 紧接的两个字节由进程 id 组成 PID
4. 最后三个字节是随机数



* + 1. 注意要点

1. **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**
2. 由于 ObjectId 中保存了创建的时间戳，所以你不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:



1. ObjectId 转为字符串



* 1. 字符串 –

BSON 字符串都是 UTF-8 编码。

* 1. 时间戳

1. 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。
2. 时间戳值是一个 64 位的值。前32位是一个time\_t值（与Unix新纪元相差的秒数）, 后32位是在某秒中操作的一个递增的序数。
3. 在单个 mongod 实例中，时间戳值通常是唯一的。
4. BSON 时间戳类型主要用于 MongoDB 内部使用。在大多数情况下的应用开发中，你可以使用 BSON 日期类型。

4.4 日期

(1) 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。

1. 创建语句



1. 返回时间的字符串

