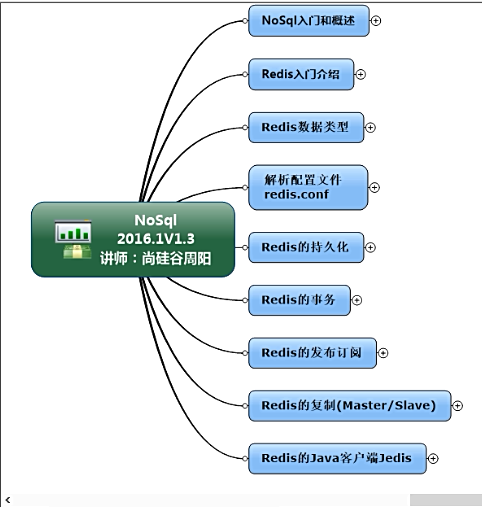
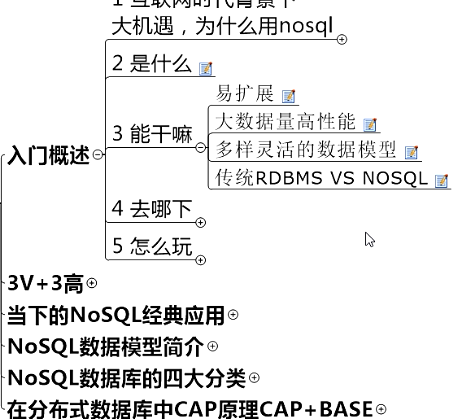
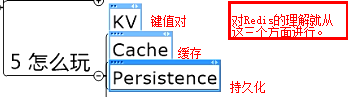
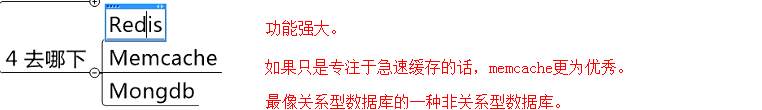
# Redis



# Nosql入门和概述





## 1.1 为啥用Nosql

|  |  |
| --- | --- |
| **笔记** | **截图** |
| 一： |  |
| 二： |  |
| 三：  ①主库只负责写入，从库只负责读取。  主从分离，读写分离。 |  |
| 四：  ①MySql数据库默认使用InnoDB引擎。  InnoDB:行锁，支持高并发；  MyISAM:表锁，高并发下会出现严重锁问题。  ②分库分表，比如根据数据是否经常更改（比如身份证信息），将数据进行分库分表，来缓解数据库压力。 |  |
| 五：  ①一些很大的图片或视频是不能放到MySql数据库里面的，这样数据库会非常非常慢。  ②大对象的4种类型  1.BLOB数据类型  1）它是用来存储二进制数据。可以存储的最大数据量是(4GB-1)\*db\_block\_size(最大32kb),也就是128TB.  2.CLOB数据类型  1）存储字符数据。可以存储的最大数据量是(4GB-1)\*db\_block\_size(最大32kb),也就是128TB.  3.NCLOB数据类型，用来存储多字节字符的数据，一般用于非英文的字符。可以存储的最大数据量是(4GB-1)\*db\_block\_size(最大32kb),也就是128TB.  4.BFILE数据类型,存储文件指针。数据文件可以存储在数据库之外，数据库只存储对该文件的引用。其最多也可以存储4GB的数据。 |  |
| 六：  p①负载均衡：  所谓负载均衡，就是说如果一组计算机节点（或者一组进程）提供相同的（同质的）服务，那么对服务的请求就应该均匀的分摊到这些节点上。 负载均衡的意义在于，让所有节点以最小的代价、最好的状态对外提供服务，这样系统吞吐量最大，性能更高，对于用户而言请求的时间也更小。  常见互联网分布式架构如上，分为客户端层、反向代理nginx层、站点层、服务层、数据层。可以看到，每一个下游都有多个上游调用，只需要做到，每一个上游都均匀访问每一个下游，就能实现“将请求/数据【均匀】分摊到多个操作单元上执行”。 | (1)【客户端层】到【反向代理层】的负载均衡，是通过“DNS轮询”实现的；  (2)【反向代理层】到【站点层】的负载均衡，是通过“nginx”实现的；  (3)【站点层】到【服务层】的负载均衡，是通过“服务连接池”实现的；  (4)【数据层】的负载均衡，要考虑“数据的均衡”与“请求的均衡”两个点，常见的方式有“按照范围水平切分”与“hash水平切分”。  https://blog.csdn.net/j2IaYU7Y/article/details/78891182 |
| 七： |  |

## 1.2Nosql，ACID

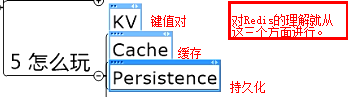
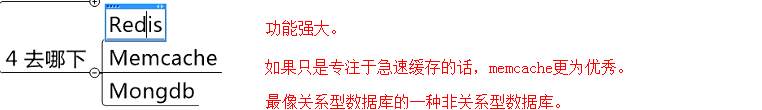
|  |  |
| --- | --- |
| ①：关系型数据库有个经典的公式：ACID.  ACID是事务的四大特性，想要成为事务，必须具备这四点。  **Atomicity：原子性**体现在对于一个事务来讲，要么一起执行成功要么一起失败，执行的过程中是不能被打断或者执行其他操作的。  **Consistency：一致性**表现为事务进行过后和执行前，整体系统都是稳定的，比如对于入账出账操作是不会有总资金的变化的。  **Isolation：隔离性**表示各个事务之间不会互相影响，数据库一般会提供多种级别的隔离。实际上多个事务是并发执行的，但是他们之间不会互相影响。  **Durability：持久性**表示一旦一个事务成功了，那么他的改变是永久性的被记录和操作。  ②：  **脏读：**所谓的脏读，其实就是读到了别的事务回滚前的脏数据。比如事务B执行过程中修改了数据X，在未提交前，事务A读取了X，而事务B却回滚了，这样事务A就形成了脏读。也就是说，当前事务读到的数据是别的事务想要修改成为的但是没有修改成功的数据。  **不可重复读：**事务A首先读取了一条数据，然后执行逻辑的时候，事务B将这条数据改变了，然后事务A再次读取的时候，发现数据不匹配了，就是所谓的不可重复读了。也就是说，当前事务先进行了一次数据读取，然后再次读取到的数据是别的事务修改成功的数据，导致两次读取到的数据不匹配，也就照应了不可重复读的语义。  **幻读：**事务A首先根据条件索引得到N条数据，然后事务B改变了这N条数据之外的M条或者增添了M条符合事务A搜索条件的数据，导致事务A再次搜索发现有N+M条数据了，就产生了幻读。也就是说，当前事务读第一次取到的数据比后来读取到数据条目少。  **不可重复读和幻读比较：**  两者有些相似，但是前者针对的是update或delete，后者针对的insert。 |  |
| 关系型数据库提供了四种级别的事务隔离，来保证不同的效果。**数据库锁是为了构建这些隔离级别存在的**。    1、Serializable（串行化）：最严格的级别，事务串行执行，资源消耗最大；  2、REPEATABLE READ（重复读）：保证了一个事务不会修改已经由另一个事务读取但未提交（回滚）的数据。避免了“脏读取”和“不可重复读取”的情况，但不能避免“幻读”，但是带来了更多的性能损失。  3、READ COMMITTED（提交读）：大多数主流数据库的默认事务等级，保证了一个事务不会读到另一个并行事务已修改但未提交的数据，避免了“脏读取”，但不能避免“幻读”和“不可重复读取”。该级别适用于大多数系统。  4、Read Uncommitted（未提交读）：事务中的修改，即使没有提交，其他事务也可以看得到，会导致“脏读”、“幻读”和“不可重复读取”。 |

## 1.3Nsql能干嘛



|  |  |
| --- | --- |
| 数据之间无关系，后面会发现他就是一大堆K-V键值对。 |  |
| Redis的性能（20160429）：写8W次/S,读11W次/S。 |  |
|  |  |
|  | 传统RDBMS：关系型数据库。 |

## 1.4去哪下，怎么玩



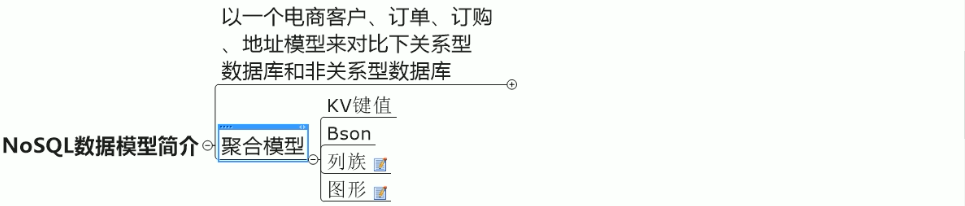
## 1.5 3V+3高

|  |  |
| --- | --- |
|  | 高可扩：横向+纵向，此处高可扩主要是指的横向方面的扩展。  横向：多台计算机，集群等。  纵向：对于单一计算机进行硬件等方面的扩展，有其极限。 |

## 1.6当下Nosql经典应用

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
|  | |  | |
| 阿里提出了第五代网站架构使命后，支付接口工程师（跟钱打交道的工程师）将会越来越重要。 |  | | |
|  | | | Mysql很重要。 |

## 1.7 Nosql数据模型简介



## 1.8 Nosql数据库的四大分类

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

下面这些举例不可能全部精通，举一反三即可，说到Redis，就能想到Oracle BDB，Tokyo等技术大概是什么样子。



## 1.9NoSql在分布式数据库中CAP原理

## CAP+BASE+分布式简介

|  |  |
| --- | --- |
|  | |
|  | NoSql中的CAP这三个特性，最多只能满足两个，不能同时满足。  P（分区容错性）：比如淘宝的服务器不可能都放在一个地区，上海，杭州，北京等都有分布，所以P是必须要满足的一个特性。  ACID:**Atomicity：原子性,Consistency：一致性**,**Isolation：隔离性,Durability：持久性** |
|  | 传统的关系型数据库（RDBMS），使用的特性是CA；  Redis使用的特性是：CP. |
| CAP的3进2：  Redis等产品的出现主要就是给传统型数据库(Oracle等)减负，所以最终要保证的强一致性通过Redis等产品来实现。CP |  |
| BASE是什么：  是为了解决 关系型数据库的强一致性要求导致可用性降低的问题 的解决方案。  为了保证可用性，数据可临时满足弱一致性，能够做到最终一致性即可。（**临时牺牲C，换取AP，最终满足C即可**） |  |
| 分布式+集群简介 |  |

# Redis入门介绍



## 2.1入门概况

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  | |
| 中/英文 | |
|  | |
|  | | Redis的企业级应用，99%都是linux版的运用和安装。 |