**2021**年阿里精选面试题及答案

1. 使用mysql索引都有哪些原则？索引什么数据结构？

B+tree和B tree什么区别？

1、 对于查询频率高的字段创建索引；

2、 对排序、分组、联合查询频率高的字段创建索引；

3、 索引的数目不宜太多

原因：a、每创建一个索引都会占用相应的物理控件；

b、过多的索引会导致insert, update、delete语句的执行效率降低；

4、 若在实际中，需要将多个列设置索引时，可以采用多列索引

如：某个表(假设表名为Student),存在多个字段(StudentNo, StudentName, Sex, Address, Phone, BirthDate),其中需要71 StudentNo, StudentName 字段进行查询，对 Sex字段进行分组，对BirthDate字段逹行排序，此时可以创建多列索引index index\_najne (StudentNo, *S*tudentName, *S*ex, BirthDate) ;#index\_najne 为索引名

在上面的语句中只创建了一个索引，但是对4个字段都赋予了豪引的功能。 创建多列索引，需要遵循BTree类型，即第一列使用时，才启用索引。

在上面的创建语句中，只有mysql语句在使用到StudentNo字段时，索引才会被启 用。

如： select \* from Student wheie StudentNo = 1000; #使用到了 StudentNo 字段，索引被启用。

以使用explain检测索引是否被启用如:explain select \* from Student where StudentNo = 1000;

5、 选择唯一性索引

唯一性索引的值是唯一的，可以更快速的通过该索引来确定某条记录。例如，学生 表中学号是具有唯一性的字段。为该字段建立唯一性索引可以很快的确定某个学生的 信息。如果使用姓名的话，可能存 在同名现象，从而降低查询速度。

6、 尽量使用数据量少的索引

如果索引的值很长，那么查询的速度会受到影响。例如，对一个CHAR (100)类型的 字段进行全文检索 需要的时间肯定要比对CHAR (10)类型的字段

需要的时间要多。

7、 尽量使用前缀来索引

如果索引字段的值很长，最好使用催的前缀来索引。例如，TEXT和BLOG类型的字 段，进行全文检索会很浪费时间。如果只检索字段的前面的若干个字符，这样可以提 高检索速度。

8、 删除不再使用或者很少使用的索引.

表中的数据被大量更新，或者数据的使用方式被改变后，原有的一些索引可能不再 需要。数据库管理员应当定期找出这些索引，将它们删除，从而减少索引对更新操作 的影响B+ tree树索引，B tree,散列

1. Mysql有哪些存储引擎？请详细列举其区别？

InnoDB：事务型存储引擎，并且有较高的并发读取频率 MEMORY:存储引擎，存放在内存中，数据量小，速度快

Merge：

ARCHIVE:归档，有很好的压缩机制

1. 设计高并发系统数据库层面该如何设计？数据库锁有哪

些类型？如何实现？

1. 分库分表：同样量的数据平均存储在不同数据库相同表（或不同表）中，减轻单表 压力，如果还是很大，就可以每个库在分多张表，根据hash取值或者其他逻辑判断将 数据存储在哪张表中
2. 读写分离：数据库原本就有主从数据库之分，查询在从服务器*，*増删改在主服务器*，*
3. 归档和操作表区分：建一张归档表，将历史数据放入，需要操作的表数据单独存储
4. 索引啊之类的创建，对于数据量很大，百万级别以上的单表，如果増删改操作不频 繁的话*，*可以创建bitMap索引，速度要快得多
5. 共享锁：要等第一个人操作完，释放锁，才能操作
6. 更新锁：解决死锁，别人可以读，但不能操作
7. 排他锁：读写都被禁用
8. 意向锁（xlock）:对表中部分数据加锁，查询时，可以跳过
9. 计划锁：操作时，别的表连接不了这张表，
10. 数据库事务有哪些？

原子性：所有操作要么全部成功，要么全部失败

—致性：例如转账，一个事务执行前和执行后必须一致 隔离性：防止脏读，重复读问题

持久性：永久性提交数据応

1. Oracle常用函数有哪些?

Concat：字符串拼接，或者|| KConcat：字符串拼接，或者|| Instr：指定字符串位置

Length：长度

Trim：去空格

Lower:小写

Upper：大写

Nvl：判断空

Replace：替换

Substr：截取

Floor：向下取整

To\_nmber:

To\_char:

To\_date：

Decode：判断函数等等

6.Sql中哪些情况可能不会走索引？

1. 查询谓词没有使用索引的主要边界，换句话说就是select \*,可能会导致不走索 引
2. 单键值的b树索引列上存在null值，导致COUNTS）不能走索引。索引列存在空 值
3. 索引列上有函数运算，导致不走索引
4. 隐式类型转换导致不走索引。
5. 表的数据库小或者需要选择大部分数据，不走索引
6. !=或者◊（不等于），可能导致不走索引
7. 表字段的属性导致不走索引，字符型的索引列会导致优化器认为需要扫描索引大 部分数据且聚簇因子很大，最终导致弃用索引扫描而改用全表扫描方式，
8. 使用like, in等，可能导致不走素引
9. 讲讲分布式唯一 ID?

确定ID存储用64位，1个64位二进制6是这样的00000000……1100 0101,切

割64位，某段二进制表示成1个约束条件，前41位为臺秒时间，后紧接9位为IP, IP之后为自増的二进制，记录当前面位数相同情况下是第几个id,如现在有10台机器, 这个id生成器生成id极限是同台机器Ims内生成2的14次方个ID。

分布式唯一ID =时间戳« 41位，int类型服务器编号« 10,序列自増sequence o 每个时间戳内只能生成固定数量如（1。万）个自増号，达到最大值则同步等待下个时 间戳，白増从。开始。将臺秒数放在最高位，保证生成的ID是趋势逢増的，每个业务 统、每个机房、每个机器生成的ID都是不同的。如39bit臺秒数|4bit业务线12bit 机房I预留17bit序列号。高位职2016年1月1耳1到现在的臺秒数，系统运行1曾, 至少需要10年x365天x24小时x3600愁xl000臺秒=320x10^9,差不多39bit给臺秒 数，每秒单机高峰并发小于10。，差不多7bit给每臺秒的自増号，5年内机房小于100 台机器，预留2bit给机房，每个机房小二100台机器，预留7bit给每个机房，业务线 小于10个，预留4bit给业务线标识。

64bit分布式ID (42bit臺秒+5bit机器ID+12位自増)等

生成分布式ID的方式：A, 2个自増表，步长相互隔开B,时间的臺秒或者纳秒C, UUID D, 64位约束条件(如上)

1. NIO和IO的区别？

第一点，町0少了1次从内核空间到用户空间的拷贝。

Byt eBuff er. allocat eDir ect ()分配的内芍使用的是本机内存而不是Java堆上的内存， 和网络或者磁盘交互都在操作系统的内核空间中发生oallocateDirectO的区别在于这 块内存不由java堆管理，但仍然在同一用户进程内。

第二点，町0以块处理数据，10以流处理数据

第三点，非阻塞，阿。1个线程可以管理多个输入输出通道

1. Redis内存数据上升到一定大小会执行数据淘汰策略，

Redis提供了哪6种数据淘汰策略？

LRU:从已设置过期时间的数据集合中挑选最近最少使用的数据淘汰

random:从已设置过期时间的数据中挑选任意数据淘汰

ttl:从已设置过期时间的数据集合中挑选将要过期的数据淘汰。

notenvision:禁止駆逐数据

如mysql中有2千万数据,redis只存储20万的热门数据。LRU或者TTL都满足热点数 据读取较多，不太可能超时特点。

redis特点:速度块,0 (1)，丰富的数据类型，支持事物原子性，可用于緩存，比memecache 速度块，可以持久化数据。

常见问题和解决：Master最好不做持久亿如RDB快照和AOF日志文件；如果数据比较 重要，某分slave幵启AOF备份数据，漩各为每秒1次，为了主从复制速度及稳定，MS 主从在同一局域网内;主从复制不要用图状结构,用单向链表更为稳定M-S-S-S-So。。。: redis过期采用懒汉+ 定期，懒汉即get/set时候检查key是否过期，过期则删除key, 定期遍历每个DB,检查制定个数个key;结合服务器性能调节并发情况。

过期淘汰，数据写入redis会附带1个有效时间，这个有效时间内该数据被认为是正确 的并不关心真实情况，例如对支付等业务采用版本号实现，redis中每一份数据都维持 1个版本号，DB中也维持1份，只有当redis的与DB中的版本一致时，才会认为redis 为有效的，不过仍然每次都要访问DB,只需要查询version版本字段即可。

1. 请描述 MyISM 和 InnoDB?

MyISM采用表级锁，对Myism表读不会阻塞读，会阻塞同表写，对Myism写则会阻塞读 和写，即一个线程获得1个表的写锁后，只有持有锁的线程可以对表更新操作，其他线 程的读和写都会等待。

InnoDB,采用行级锁，支持事务，例如只对a列加索引，如果update ... where a=l and b=2其实也会锁整个表，select使用共享锁，update insert delete采用排它锁， commit会把锁职消，当然select by id for update也可以制定排它锁。

1. 请描述实时队列？

实时队列采用双队列模式，生产者将行为记录写入Queuel, worker服务从Queuel消费 新鲜数据，如果异常则写入Queue 2 (主要保存异常数据), RetryWorker会监听Queue 2, 消费异常数据，如果还未处理成功按照一定的策略等待或者将异常数据再写入Queue2, 如果数据发生积压可以调整worker的消费游标，从最新数据重新开始消费，保证了最 新data得到处理，中间未处理的一段则可以启动backupworker指定起止游标在消费完 指定区间的数据后*，*backupWorker会自动停止。

DB降级开关后，可直接写入redis (stoim),同时将数据写入一份到Retry队列，在 幵启DB降级开关后消费Retry队列中的数据，从而把数据写入到mysql中，达到最终 一致性。MYSQL切分为分片为2的N次方，例如原来分为两个库d0和dl均放在s0服 务器上，s0同时有备机si,扩容只要几步骤：确保s0到si服务器同步顺利，没有明 显延迟；s0暂时关闭读写权限；确保si已经完全同步到s0更新；si开放读写权限； dl的dns由sO切换到si; sO开放读写权限。

1. DB的特性和隔离级别？

4大特性：原子性，一致性，分离性，持久性

隔离级别：

读提交：写事务禁止读

读未提交：写事务允许读

可重复读：写事务禁止读事务，读禁止与

序列化：全部禁止

详细说明：读提交1个事务开始写则全咅噤止其他事务访问该行。读未提交1个事务开 始写则不允许其他事务同时写，但可以读。可重复读读事务会禁止写事务，写事物则 禁止其他任何事务。序列化性能最低，全部禁止，串行执行。MYSQL默认的是可重复 读。

1. ICMP是什么协议,处于哪一层？

Internet控制报文协议，处于网络层（IP层）

1. 讲一下NIO和网络传输.

NIO Reactor反应器模式，例如汽车是乘客访问的实体reactor,乘客上车后到售票员 处Acceptor登记，之后乘客便可休息睡觉了，到达乘客目的地后，售票员Aceptor将其唤 ■即可。持久TCP长链接每个client和曙匕如工之间有存在一个持久连接，当CCU （用户 并发数量）上升，阻塞server无法为每个连接运行1个线程，自己开发1个二进制协议， 将message压缩至3-6倍，传输双向且消息频率高，假设gver链接了 2000个client, 每个client平均每分钟传输1T0个message, 1个messaged的大小为几百字节/几千字节, 而server也要向client广播其他玩家的当前信息，需要高速处理消息的能力。Buffer, 网络字节存放传输的地方，从channel中读与，从buff er作为中间存储格式，channel是 网络连接与buffer间数据通道，像之前的socket的stream。

1. 内存泄漏

未对作废数据内存单元置为null,尽早释放无用对象的引用，使用临时变量时，让引 用变量在推出活动域后自动设置为null,暗示垃圾收集器收集；程序避免用String拼 接，用StringBuffer,因为每个String会占用内存一块区域；尽量少用静态变量（全 局不会回收）；不要集中创建对象尤其大对象，可以使用流操作；尽量使用对象池，不 再循环中创建对象，优化配置；创建对象到单例getlnstance中，对象无法回收被单例 引用；服务器session时间设置过长也会引起内存泄漏。

1. 请描述平衡二叉树.

平衡二叉树，左右高度之差不超过1, Add/delete可能造成高度＞1,此时要旋转，维持 平衡状态，避免二叉树退化为链表，让AM/Delete时间复杂度但控制在0 （log2N）,旋 转算法2个方法，1是求树的高度，2是求2个高度最大值，1个空树高度为-1,只有1 个根节点的树的高度为。，以后每一层+1,平衡树任意节点最多有2个儿子，因此高度 不平衡时，此节点的2棵子树高度差为2。例如单旋转，双旋转，插入等。

红黑树放弃完全平衡，追求大钱平衡，保祉每次插入最多要3次旋转就能平衡。

1. 请问溢岀的原因?

是否逢归的调用；大量循环；全局变量是否过多；数组，List, Map数据是否过大；用 DDMS工具检查地方。

内存溢出的原因