# 面试常见

## 数据库事务

### 定义

指作为单个逻辑工作单元执行的一系列操作，要么完全执行要么完全不执行。

符合

* 原子性：事务中的操作要么都做要么都不做
* 一致性：事务的执行结果必须从数据库从数据库的一个一致行状态转化为另一个一致性状态 <https://www.zhihu.com/question/31346392>
* 隔离性：一个事务的执行状态不能被另一个事务干扰
* 持久性：一个事务一旦提交，他对数据库的改变是永久的

### 事务的隔离级别

1. 读未提交：引发脏读，读到还没有提交的数据
2. 读已提交：解决了脏读，但是有不可重复读的问题，导致这个问题的原因有：一，有一个交叉事务有新的commit，导致只读一次得到的数据不一致，二，有多个实例并发操作数据库，同一个事务多个实例提交；
3. 可重复读：mysql的默认事务隔离级别，他保证了多个实例并发读取数据时，读到的数据时一致的。但是会造成幻读，当用户读取某一行时，另一个事务又在你读取的范围内插入了新行，当用户再次读取该范围行时，就会看见新的幻影行。
4. 可串行化：强制要求事务串行执行，从而解决了幻读，一般做法就时加上共享锁。但是会导致大量的超时现象和锁竞争。

住：事务的隔离级别只对当前链接起作用

### 事务运行的三种模式

1. 自动提交事务：每条单独的语句都是一个事务，每个语句都隐含一个commit
2. 显式事务：以begin transaction开始，以commit/rollback结束
3. 隐性事务：在前一个事务完成后，新的事务隐式启动，但是都要进行显示的commit或rollback结束。

## 数据库三范式

什么是范式：数据库的设计会影响数据库的性能，从而带来效率问题。所以建立科学的数据库结构需要符合规范，从而优化数据的存储方式。这个规范就是范式。

1. 第一范式：每一列属性都是不可再分割，确保属性的原子性。（地址->省+市）
2. 第二范式：满足第一范式的前提下，每一行中的数据只能与一列相关，就是一行数据只做一件事（姓名，性别，地址，民族，房间号，这样一个人定多间房间就会产生很多冗余，应该改为两个表，一个事登记信息，一个是房间信息，通过id关联）
3. 第三范式：数据不存在传递关系，就是每个属性都和主键有直接关系而不是间接关系。（学号，姓名，年龄，性别，院校，院校地址，院校电话），这种主键是学号，关联院校，院校又关联地址和电话，要吧他们拆分成学生信息和院校信息。

## 反相代理和正向代理

### 正向代理

定义：正向代理就是顺着请求的方向进行代理。就是代理服务器是由你配置为你服务的。去请求目标地址。

比如我们要访问谷歌网站，我们直接访问不通，那么我们就可以找一个代理服务器为我们服务，我们通过代理服务器请求到谷歌网站。对于谷歌而言，他只知道有一个服务器访问了自己，并不知道是你访问不了他找了一个代理服务器访问自己。

### 反相代理

定义：与正向代理相反，代理服务器是为目标服务器服务的，虽然整体访问路线依然是client->Proxy->server。

比如我们要访问百度网站，百度代理服务器对外的域名是[www.baidu.com](http://www.baidu.com)，具体的内部服务器节点我们不知道。现实中我们通过访问百度代理服务器，代理服务器给我们转发请求到他们N多个服务器节点中的一个，给我们返回请求结果。

反相代理的实现：

1. 需要一个负载均衡设备负责分发用户的请求，将用户的请求分发到空闲的服务器上。
2. 服务器返回自己的服务到负载均衡设备上。
3. 负载均衡设备将服务器的返回结果返回给用户。

作用：防止外网对内网的恶性攻击，缓存以减少服务器的压力，访问安全控制，进行负载均衡，将用户的请求分配给多个服务器。

正向代理隐藏真实客户端，反向代理隐藏真实服务端

#### Nginx

## Elasticsearch

### 什么是es

elasticsearch是一款分布式，高性能，高可用，可伸缩的搜索和分析系统。

### 与mysql的区别

es通常是进行某个词条进行检索，mysql通常会通过模糊查询的方法去查，效率低，而且不支持相关搜索。也就是说es一半适合文本搜索。

基本概念：

1. 文档：es属于面向文档的存储，文档用json体现，每个字段都有自己就的类型。一个文档对应了关系型数据库的条数据。
2. 索引：es中的索引相当于关系型数据库中的数据库，索引通过mapping来定义类型，setting制定分片。
3. 类型：相当于关系型数据库中的表。
4. 节点：一个es实例，又分为：

* Master eligible 候选主节点
* Data：数据节点
* Coordinating：代理节点，接受用户请求

1. 分片：有主分片和副分片两种，主分片存储数据，副分片保留数据的副本，分片将数据分到了多个节点上解决了存储空间不足的问题，也实现了数据的并行访问。

### es数据的写入过程

1. 用户选择一个节点发送请求，这个节点就是协调节点。
2. 协调节点对发送过来的文档进行路由，将请求转发给对应的node。
3. 对应的节点上的主分片处理数据，将数据存储在主分片当中，并且向副分片复制副本
4. 协调节点检测到已经完成主分片存储和副分片拷贝后，返回响应结果返回到客户端。

### Es读数据的过程

1. 用户发送请求到任意一个节点，成为协调节点
2. 该节点用过doc id进行hash路由，将请求转发到对应的node当中，此时会采用round-robin随机轮询算法，在他的主分片和所有的副分片中随机选择一个，让请求负载均衡。
3. 查询到数据的节点将返回结果返回至协调节点
4. 协调节点将返回结果返回到客户端

### Es的数据搜索过程

es最强大的就是做全文检索，假如你的es数据库当中有三条数据

java真好玩

java真有趣

j2ee好难学

你根据java关键词进行搜索，将包含java的文档给搜索出来，es就会返回给你：java真好玩和java真有趣

1. 客户端发送请求到一个节点，使之成为协调节点
2. 协调节点将搜索请求发送到所有的主分片和副分上
3. 每个节点将自己搜索的结果（此时是doc id）返回给协调节点，由协调节点进行合并，排序，分页等操作，产出最终结果
4. 接着由协调节点根据doc id去各个节点上拉去实际的文档数据，并返回给客户端

## 线程池

### 为什么会使用线程池

服务器会遇到这种请求：单个请求处理需要的时间很短，但是会有大量的这种请求。因为一般服务应用程序会每接收到一个请求就会新建一个线程，然后在新建的线程里处理该请求，但是当请求量很大的时候会频繁得创建销毁线程，造成开销很大，在创建和销毁线程话费的时间和消耗的系统资源比实际处理用户请求还要大。另外过多的活动的线程也会消耗大量的系统资源，可能会导致内存用完或者切换过度。

### 线程池的风险

1. 死锁： 当线程A持有对象x的独占锁，并且等待对象Y的锁，线程B持有对象Y的独占锁，并且等待对象x的锁。此时就会发生死锁。这是所有多线程都可能出现的，但是线程池会有特有的死锁情况发生：所有池线程都在执行已阻塞的等待队列中等待另一任务的执行结果，但是这个任务却因为没有未被占用的线程来运行。
2. 资源不足：我们只有恰当得调整了线程池的大小，才会发挥它的优势，因为线程会消耗内存以及其他大量的系统资源，除了thread对象所需要的内存之外，每个线程都需要两个可能很大的执行调用堆栈，除此之外jvm会为每个java线程创建一个本地线程，这些本地线程也会消耗额外的系统资源。此外线程之间的切换开销虽然很小，但是如果存在很多线程的话环境切换也可能严重的影响系统性能。
3. 并发错误：线程池和其他的排队机制依靠使用wait()和notify()方法，这俩个方法都难于使用，如果编码不正确，可能会导致同志丢失，从而导致线程有空闲的，但是队列中依然有等待的。
4. 线程泄漏：线程泄漏的一种情况是在任务抛出一个runtimeexception或者error时，如果池类没有捕捉到它们，那么线程只会退出，而线程池的数量会永久得减少一个，当这种情况发生很多次时，线程池最终会变成空，系统将会停止。另一种情况是有些任务可能会永久得等待某些资源或着等待用户输入，这些资源又不能保证变得可用，用户可能已经回家了，像这些情况任务会永久得停止，而这些停止的任务也会引起线程池泄漏问题。对于这样的任务，我们要么给他自己的线程，要么限制线程等待时间。
5. 请求过载：过量的请求排在工作队列，这些等待执行的任务可能会消耗大量的系统资源，引起资源匮乏。我们可以返回一个系统繁忙的响应该客户，来拒绝请求。

### 几种常见的线程池

1. newCachedThreadPool：可缓存线程池，如果线程池的大小超过处理需要，则可以灵活的回收空闲线程，若无可回收则新建线程。
2. newFixedThreadPool：创建一个指定工作线程数量的线程池，每当提交一个任务就是创建一个工作线程，如果工作线程数量达到线程池的初始的最大数，则将提交的任务存到池对列当中。
3. newSingleThreadExecutor：创建一个单线程化的Executor，只创建唯一的工作者线程来执行任务，保证所有任务按照指定顺序（FIFO，LIFO，优先级），如果这个线程异常结束，会有另一个取代它，保证顺序执行。
4. newScheduleThreadPool：创建一个定长的线程池，而且支持定时以及周期性得执行任务。

## Mysql中的索引原理