|  |
| --- |
| 技术问题(至少50个) |
| 问题1：线程与进程区别?  回答：进程是所有线程的集合，每一个线程是进程中的一条执行路径。  问题2：什么是多线程？  回答：及时提高程序的效率（同时执行，互不影响）。  问题3：多线程应用场景？  回答：主要能体现到多线程提高程序效率。  举例: 迅雷多线程下载、分批发送短信、爬虫等。  问题4：创建多线程有哪些方式？  回答： 1、使用继承Therad类方式 继承Thread类重写run方法 2、使用实现runable接口方式3、使用匿名内部类方式 4、callable 5、使用线程池创建线程。  问题5：实现多线程使用继承thread类好还是实现runable接口好？  回答：使用实现Runnable接口好，原因实现了接口还可以继承，继承了类不能再继承。  问题6：启动线程是使用调用start方法还是run方法？  回答：开始执行线程 注意 开启线程不是调用run方法，而是start方法  调用run只是使用实例调用方法。  问题7 ：什么是守护线程  回答：Java中有两种线程，一种是用户线程，另一种是守护线程(gc线程)。用户线程是指用户自定义创建的线程，主线程停止，用户线程不会停止，守护线程当进程不存在或主线程停止，守护线程也会被停止。使用setDaemon(true)方法设置为守护线程。  问题8：如何解决多线程之间线程安全问题?  回答：使用多线程之间同步synchronized或使用锁(lock)。  问题9：为什么使用线程同步或使用锁能解决线程安全问题呢？  回答：将可能会发生数据冲突问题(线程不安全问题)，只能让当前一个线程进行执行。代码执行完成后释放锁，然后才能让其他线程进行执行。这样的话就可以解决线程不安全问题。  问题10：volatile与synchronized区别  回答：①volatile轻量级，只能修饰变量。synchronized重量级，还可修饰方法  ②volatile只能保证数据的可见性，不能用来同步，因为多个线程并发访问volatile修饰的变量不会阻塞。  synchronized不仅保证可见性，而且还保证原子性，因为，只有获得了锁的线程才能进入临界区，从而保证临界区中的所有语句都全部执行。多个线程争抢synchronized锁对象时，会出现阻塞。  问题11：wait与sleep区别?  回答：对于sleep()方法，我们首先要知道该方法是属于Thread类中的。而wait()方法，则是属于Object类中的。  sleep()方法导致了程序暂停执行指定的时间，让出cpu给其他线程，但是他的监控状态依然保持者，当指定的时间到了又会自动恢复运行状态。  在调用sleep()方法的过程中，线程不会释放对象锁。  而当调用wait()方法的时候，线程会放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象调用notify()方法后本线程才进入对象锁定池准备  获取对象锁进入运行状态。  问题 12:Lock 接口与 synchronized 关键字的区别?  回答：Lock 接口可以尝试非阻塞地获取锁 当前线程尝试获取锁。如果这一时刻锁没有被其他线程获取到，则成功获取并持有锁。  Lock 接口能被中断地获取锁 与 synchronized 不同，获取到锁的线程能够响应中断，当获取到的锁的线程被中断时，中断异常将会被抛出，同时锁会被释放。  Lock 接口在指定的截止时间之前获取锁，如果截止时间到了依旧无法获取锁，则返回。  问题 13:什么是线程池?及作用?  回答:Java中的线程池是运用场景最多的并发框架，几乎所有需要异步或并发执行任务的程序  都可以使用线程池。在开发过程中，合理地使用线程池能够带来3个好处。  第一：降低资源消耗。通过重复利用已创建的线程降低线程创建和销毁造成的消耗。  第二：提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。  第三：提高线程的可管理性。线程是稀缺资源，如果无限制地创建，不仅会消耗系统资源，  还会降低系统的稳定性，使用线程池可以进行统一分配、调优和监控。但是，要做到合理利用  作用：  线程池是为突然大量爆发的线程设计的，通过有限的几个固定线程为大量的操作服务，减少了创建和销毁线程所需的时间，从而提高效率。  如果一个线程的时间非常长，就没必要用线程池了(不是不能作长时间操作，而是不宜。)，况且我们还不能控制线程池中线程的开始、挂起、和中止。  问题 14:线程池四种创建方式?  回答:Java通过Executors（jdk1.5并发包）提供四种线程池，分别为：  newCachedThreadPool创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。  newFixedThreadPool 创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待。  newScheduledThreadPool 创建一个定长线程池，支持定时及周期性任务执行。  newSingleThreadExecutor 创建一个单线程化的线程池，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。  问题 15:线程池原理剖析?  回答:提交一个任务到线程池中，线程池的处理流程如下：  1、判断线程池里的核心线程是否都在执行任务，如果不是（核心线程空闲或者还有核心线程没有被创建）则创建一个新的工作线程来执行任务。如果核心线程都在执行任务，则进入下个流程。  2、线程池判断工作队列是否已满，如果工作队列没有满，则将新提交的任务存储在这个工作队列里。如果工作队列满了，则进入下个流程。  3、判断线程池里的线程是否都处于工作状态，如果没有，则创建一个新的工作线程来执行任务。如果已经满了，则交给饱和策略来处理这个任务。  问题 16:Spring中的IOC和AOP如何理解？  回答:1.IOC就是控制反转,控制权的反转  这个话题比较有意思，我之前在一家单位中维护过一个基于jdbc+servlet的派出所人员管理的项目，这个项目比较老旧，没有详细的mvc层级拆分，在servlet中大量使用new指令创建对象，这给后期维护人员无论是在代码查看和维护上带来不便。其实这呢没有引入ioc的理念。  IOC就是把创建调用对象的权力由对象本身转换给spring容器管理,由spring容器来管理对象之间的关系依赖,这就达到了控制反转的目的,这样也实现了对象之间的解耦，方便统一维护。  2.IOC注入的几种方式  在IOC中bean的注入方式有三种,setter注入,构造器注入,接口注入  Setter注入,这个用的比较多,给对象提供一个set方法就可以,不需要get方法  其次是构造器注入,给对象提供构造方法  接口注入也是一种  IOC底层实现原理:反射+工厂模式+xml配置  3.AOP实现  AOP,面向切面编程,类比OOP面向对象编程OOP之前有一个C语言的面向过程编程  AOP并不是来替代OOP的,AOP而是在OOP的基础上演变诞生的  AOP,也就是面向切面编程,它们指定了一些预编译的方式和动态代理,在不改变原程序业务代码的情况下来实现新增功能.  AOP通知有五种:前置通知,后置通知,环绕通知,异常通知,最终通知.  比如登录QQ需要登录用户名密码,这个时候就可以用到前置通知,在业务之前给一个通知  还有事务管理的功能,在数据的增删改的时候需要启动事务  AOP也就是做一些除了业务功能以外的功能,比如日志记录,事务管理,安全  事务的ACID四大特性:原子性,持久性,隔离性,一致性  动态代理:jdk cglib 默认的是jdk,jdk要有接口,cglib必须要有实现类  Spring提倡面向接口编程,所以spring默认的动态代理方式是jdk动态代理,  AOP的底层原理就是:反射+动态代理  问题17 :SpringMVC运行原理？  回答:首先，Tomcat启动时加载SpringMVC的核心控制器DispatcherServlet。当用户向服务器发送请求时，核心控制器将请求交给HandlerMapping做解析，HandlerMapper将要访问的Controller返回给核心控制器。核心控制器将用户的请求发送给指定的Controller做业务处理。当业务处理完成后，SpringMVC将需要传递的数据和跳转的视图名称封装为一个ModelAndView，将ModelAndView对象发送给核心控制器。核心控制器从ModelAndView对象中取出视图名称，交给视图解析器做解析，视图解析器中配置着页面路径的前缀和后缀，解析之后将要跳转的页面反馈给核心控制器。最终核心控制器将数据发送给页面通过Response对象响应给用户。  问题 18:spring事务的传播特性和隔离级别吗?  回答: 这块了解一些，首先spring一共定义了七种事务传播属性，【分别是propagation\_required(支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择)、  propagation\_supports(支持当前事务，如果当前没有事务，就以非事务方式执行)、propagation\_mandatory(支持当前事务，如果当前没有事务，就抛出异常)、propagation\_requires\_new(新建事务，如果当前存在事务，把当前事务挂起)、propagation\_not\_supported(以非事务方式执行操作，如果当前存在事务，就把当前事务挂起)、  propagation\_never(以非事务方式执行，如果当前存在事务，则抛出异常)、propagation\_nested(如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则进行与PROPAGATION\_REQUIRED类似的操作)】  但是在我个人平时的开发中接触比较多的就是那个propagation\_required,这种事务传播特性可支持当前事务，如果当前没有事务则会新创建一个事务，打个比方说：有两个方法methodA和methodB，如果这两个方法都配置了该事务，当methodA方法内调用methodB方法时methodB是支持methodA事务的,同时如果methodA没有事务,methodB有事务,那么就会新建一个事务。同时spring还定义了如propagation\_supports、propagetion\_never、propagation\_requires\_new等其他事物传播特性。  问题 19:spring事务的隔离级别?  回答: 事物隔级别这块，spring共支持5种事物隔离级别，分别是  默认隔离级别（Default:使用数据库设置的隔离级别(默认),由DBA默认的设置来觉得隔离级别）  读未提交数据（read uncommited读未提交数据：所有的事务都可以“看到”未提交事务的执行结果；会导致脏读、不可重复读和幻读的问题的出现）  读已提交数据（read commited读已提交数据：一个事务在开始时，只能"看见"已提交事务所做的改变；Oracle和sql server默认的级别，可以避免脏读，但不可重复读和幻读问题仍然会出现）  可重复读（Repeatable read可重读：确保同一事务的多个实例在并发读取数据时，会看到同样的行，事务在读取某数据的瞬间（就是开始读取的瞬间），必须先对其加 行级共享锁，直到事务结束才释放；  事务在更新某数据的瞬间（就是发生更新的瞬间），必须先对其加 行级排他锁，直到事务结束才释放）  可串行化（Serializable可串行化：最高的事务隔离级别，每个读的数据上加上共享锁  事务在读取数据时，必须先对其加 表级共享锁 ，直到事务结束才释放；  事务在更新数据时，必须先对其加 表级排他锁 ，直到事务结束才释放）  这里，不同的隔离级别，会出现不同的影响：  对于读取未提交数据（read uncommited）来说，可能会出现脏读、幻读和不可重复读，打个比方说：事务A执行update操作将字段name的值从小明更改为小黄,但是它没有提交,这时事务B来读数据库,读到的name等于小黄,如果事务A进行了事物回滚,这就对于事物B来说小黄是无效数据，也造成了脏读。  对于读已提交数据（read commited）来说，虽然避免了数据脏读，但是也可能会造成幻读的发生，比如：事务A对user表某一行进行读取操作,读到name=小明,这时事务B执行了update操作将字段name的值从小明更改为小黄,并在事务A未提交前已成功提交事务,此时对于事务A来说name真实的值不再是小明,这样造成了数据幻读。  但是对于可重复读（Repeatable read）来说，在同事务多实例的并发环境下可避免幻读的产生，它是使用锁机制实现的，比如说：在并发环境下，事务A在读取某行数据时,会对这一行加行级共享锁,该锁仅仅对读操作事务共享，这样保证了当其他事务读取到这行记录时数据可达到一致，避免幻读发生, 同时如果事务A对表中某一行数据进行更改操作,那么会对该行添加行级排它锁,这把锁会排斥其他事务对该行的修改操作，做到了数据一致性。  串行化（Serializable）:与可重读的类似,它不是使用行级共享锁或者行级排它锁实现的，而是使用了表级共享锁和表级排它锁，相对可重读的来说，支持的并发效率不高。  问题 20:HashMap底层是如何实现的？或者HashMap是如何去重的？  回答:首先HashMap底层维护了一个数组加链表的数据结构（此时可以用笔画一下），这个数组为entry类型的数组，entry类中有object类型的key和value属性，还有一个entry类型的next属性，这个next指向下一个entry，这样就完成了一个链表结构。HashMap在添加一个{key，value}时，首先会对key进行hash运算（这个运算调用了object的hashcode（）方法，获取hash码，为了让数据更均匀的分布在数组上，HashMap对该值又进行了异或和位运算的一个复合运算，来将该值得复杂度提高）  static final int hash(Object key) {  int h;  return (key == null) ? 0 : (h = key.hashCode()) ^ (h >>> 16);  }  获取一个hash值，该值与数组长度取余后即可定位该值存储下标位置，这时会判断数组在该下标下是否有Entry对象，如果有，会调用equals()方法比较两个key(没有经过hashcode()计算)是否相同，如果相同，则做覆盖操作，如果不相同，则查看Entry对象中next是否为空，如果为空，就直接将存入的{key,value}封装到一个entry对象中，然后把该对象赋值给next变量，如果next不为空，重复以上操作。在这个过程中，hashmap会根据负载因子（loadfactor）与数组长度的乘积做是否扩容的判断，比如hashmap负载因子为0.75，默认初始化数组长度为16，那么如果数组中存储的元素个数超过0.75\*16=12的话，就是触发一次扩容，其实这么处理是为了让hashmap处于高效存储检索状态，是对hashmap的一次优化。那么每次扩容都会将数组长度做向左位移一位的操作，也就是扩大一倍。  都知道，链表结构长度越大，在检索查询时，效率相对不高，在jdk1.8之后HashMap做了相关优化，跟负载因子类似，它定义了两个变量，一个是树阈值，一个是树还值，分别为8和6，当链表长度大于等于8时，会将链表结构转换成一个平衡的二叉树结构，也就是红黑树结构，那如果做删除操作时，当树节点数小于等于6时，则还原为链表结构，这样，提高了检索速度。  问题21 :对arraylist理解？  回答:1、ArrayList简述  首先，众所周知，ArrayList是一个线程不安全的容器，其底层维护了一个数组，它最大的特点就是按照插入顺序来保存元素，可以利用下标来查找值，因此查询速度十分的快，其缺点也正是因为其数组的特性在中间插入元素很慢、删除下标靠前的元素很慢。  2、ArrayList的构造器  接下来说一说ArrayList的构造器，我们常用的构造函数有两个，分别是有参和无参的，无参的构造函数十分简单，只是初始化了一个空的Object型数组。  再来说有参的，其实参数的含义就是这个集合的容量，也就是说如果我们传入的参数为10，那么ArrayList将会初始化一个长度为10的空的Object型数组，这两个构造器究竟有何区别，接下来才是重点。  3、ArrayList的添加方法  在添加元素之前，会先判断是否声明的是无参构造函数，在这里就显现出了两个构造器的区别了，如果我们一开始声明的是一个空参的构造函数，这是ArrayList就会初始化数组长度为10，试想，如果我们只需要存储3个元素，而我们声明了一个空参的构造函数，那么集合的长度会是10，也就是说，数组有7个长度的空间被浪费了，这就是对内存的一种浪费,所以如果在使用ArrayList时在确认放置元素数量的情况下,再在构造器中传入合适的参数,这样可避免内存的浪费。  接下来会再去需要确保容量是否够用，如果够用，那么不需要扩容，反之，则进行扩容，它是通过Arrays.copyOf(elementData, newCapacity)方法实现扩容的，这个方法需要传入两个参数，一个是原数组，一个是原数组容量1.5倍的一个数值(这是数值是通过原来数组长度经过位移运算向右位移一位+原来长度实现的代码如下:int newCapacity = oldCapacity + (oldCapacity >> 1))，这样就生成一个新的数组，并将原有的数组内容全复制进去。试想一下，我们需要保存1000个元素，而一开始默认长度是10，那么集合需要很多次扩容，每次扩容是上一次容量的1.5倍，每次扩容还要进行复制。如果不事先声明一个长度的话，使用效率会大大降低，即便是不知道具体数字，也可以指定一个大概的容量。这也说明了两个构造器的另一个区别。  问题 22:java中的集合框架？  回答: 首先java的集合架构分为两个根接口collection和map，因为集合都需要进行迭代遍历，所以他们都继承了Iterator接口，那，collection根接口下有两个子接口，一个是有序可重复的list接口，另一个为无序不可重复的set接口。  List接口下有这么几个常用的实现类，分别为ArrayList、LinkedList、Vector, ArrayList呢底层维护了一个object类型的数组，但是比较有意思的是在调用new指令创建空参对象时，它底层的数组为null，也就是没有被初始化，但是一旦调用add方法，ArrayList会进行扩容，初次扩容默认长度为10，这是ArrayList的一种初始化方式，Vector相对ArrayList是线程安全的，它的方法被synchronized关键字修饰，是线程安全的。剩下的一个LinkedList数据结构为双向链表结构。  Set接口也有这么几个常用的实现类，分别是HashSet（原理跟HashMap一样，只要将map中的value值去掉即可）、TreeSet（平衡的二叉树，即红黑树）。  Map接口下常用的实现类有HashMap、TreeMap、HashTable, HashMap实现原理略，HashTable相对HashMap是线程安全的。  问题 23:ConcurrentHashMap 如何实现高效地线程安全？  回答:HashMap是非线程安全的，Hashtable是线程安全的，但是由于Hashtable是采用synchronized进行同步，相当于所有线程进行读写时都去竞争一把锁，导致效率非常低下。  　　ConcurrentHashMap可以做到读取数据不加锁(进行写的操作加锁)，并且其内部的结构可以让其在进行写操作的时候能够将锁的粒度保持地尽量地小，不用对整个ConcurrentHashMap加锁(只对每一个segment加锁)。  ConcurrentHashMap为了提高本身的并发能力，在内部采用了一个叫做Segment的结构，一个Segment其实就是一个类Hash Table的结构，Segment内部维护了一个链表数组，我们用下面这一幅图来看下ConcurrentHashMap的内部结构：  　　从上面的结构我们可以了解到，ConcurrentHashMap定位一个元素的过程需要进行两次Hash操作，第一次Hash定位到Segment，第二次Hash定位到元素所在的链表的头部，因此，这一种结构的带来的副作用是Hash的过程要比普通的HashMap要长，但是带来的好处是写操作的时候可以只对元素所在的Segment进行加锁即可，不会影响到其他的Segment，这样，在最理想的情况下，ConcurrentHashMap可以最高同时支持Segment数量大小的写操作（刚好这些写操作都非常平均地分布在所有的Segment上），所以，通过这一种结构，ConcurrentHashMap的并发能力可以大大的提高。  问题 24:hibernate与mybatis区别  回答:Hibernate和Mybatis都是一个开放源代码的对象关系映射框架,Hibernate对JDBC进行了非常轻量级的对象封装,建立对象与数据库表的映射，它是一个全自动的、完全面向对象的持久层框架。Mybatis原名：ibatis,2010年由谷歌接管以后更名，是一个半自动化的持久层框架。  相同点:  （1）Hibernate与MyBatis都是通过SessionFactoryBuider由XML配置文件生成SessionFactory，由SessionFactory 生成Session，由Session来开启执行事务和SQL语句。  （2）Hibernate和MyBatis都支持JDBC和JTA事务处理。  （3）基于ORM思想解决了entity和数据库的映射问题  不同点:  Hibernate:  Hibernate的DAO层开发比MyBatis简单，Mybatis需要维护SQL和结果映射。  Hibernate对对象的维护和缓存要比MyBatis好，对增删改查的对象的维护要方便。  Hibernate数据库移植性很好，MyBatis的数据库移植性不好，不同的数据库需要写不同SQL。  Hibernate有更好的二级缓存机制，可以使用第三方缓存。MyBatis本身提供的缓存机制不佳。  Mybatis:  MyBatis可以进行更为细致的SQL优化，可以减少查询字段。  MyBatis容易掌握，而Hibernate门槛较高。  问题25 :MySQL优化？  回答:1、创建索引  对于查询占主要的应用来说，索引显得尤为重要。很多时候性能问题很简单的就是因为我们忘了添加索 引而造成的，或者说没有添加更为有效的索引导致。如果不加索引的话，那么查找任何哪怕只是一条特定的数据都会进行一次全表扫描，如果一张表的数据量很大而 符合条件的结果又很少，那么不加索引会引起致命的性能下降。但是也不是什么情况都非得建索引不可，比如性别可能就只有两个值，建索引不仅没什么优势，还会 影响到更新速度，这被称为过度索引。  2、复合索引  比如有一条语句是这样的：select \* from users where area='beijing' and age=22;  如 果我们是在area和age上分别创建单个索引的话，由于mysql查询每次只能使用一个索引，所以虽然这样已经相对不做索引时全表扫描提高了很多效率， 但是如果在area、age两列上创建复合索引的话将带来更高的效率。如果我们创建了(area, age, salary)的复合索引，那么其实相当于创建了(area,age,salary)、(area,age)、(area)三个索引，这被称为最佳左前缀 特性。因此我们在创建复合索引时应该将最常用作限制条件的列放在最左边，依次递减。  3、索引不会包含有NULL值的列  只要列中包含有NULL值都将不会被包含在索引中，复合索引中只要有一列含有NULL值，那么这一列对于此复合索引就是无效的。所以我们在数据库设计时不要让字段的默认值为NULL。  4、使用短索引  对串列进行索引，如果可能应该指定一个前缀长度。例如，如果有一个CHAR(255)的 列，如果在前10 个或20 个字符内，多数值是惟一的，那么就不要对整个列进行索引。短索引不仅可以提高查询速度而且可以节省磁盘空间和I/O操作。  5、排序的索引问题  mysql查询只使用一个索引，因此如果where子句中已经使用了索引的话，那么order by中的列是不会使用索引的。因此数据库默认排序可以符合要求的情况下不要使用排序操作；尽量不要包含多个列的排序，如果需要最好给这些列创建复合索引。  6、like语句操作  一般情况下不鼓励使用like操作，如果非使用不可，如何使用也是一个问题。like “%aaa%” 不会使用索引而like “aaa%”可以使用索引。  7、不要在列上进行运算  select \* from users where YEAR(adddate)<2007;  将在每个行上进行运算，这将导致索引失效而进行全表扫描，因此我们可以改成  select \* from users where adddate<‘2007-01-01';  8、不使用NOT IN和<>操作  NOT IN和<>操作都不会使用索引将进行全表扫描。NOT IN可以NOT EXISTS代替，id<>3则可使用id>3 or id<3来代替。  问题 26:什么是数据库范式？  回答:为了建立冗余较小、结构合理的数据库，设计数据库时必须遵循一定的规则。在关系型数据库中这种规则就称为范式。范式是符合某一种设计要求的总结。要想设计一个结构合理的关系型数据库，必须满足一定的范式。  问题 27:数据库三大范式？  回答:第一范式：1NF是对属性的原子性约束，要求属性(列)具有原子性，不可再分解；(只要是关系型数据库都满足1NF)  第二范式：2NF是对记录的惟一性约束，表中的记录是唯一的, 就满足2NF, 通常我们设计一个主键来实现，主键不能包含业务逻辑。  第三范式：3NF是对字段冗余性的约束，它要求字段没有冗余。 没有冗余的数据库设计可以做到。  但是，没有冗余的数据库未必是最好的数据库，有时为了提高运行效率，就必须降低范式标准，适当保留冗余数据。具体做法是： 在概念数据模型设计时遵守第三范式，降低范式标准的工作放到物理数据模型设计时考虑。降低范式就是增加字段，允许冗余。  问题 28:分表分库？  回答:垂直拆分  垂直拆分就是要把表按模块划分到不同数据库表中（当然原则还是不破坏第三范式），这种拆分在大型网站的演变过程中是很常见的。当一个网站还在很小的时候，只有小量的人来开发和维护，各模块和表都在一起，当网站不断丰富和壮大的时候，也会变成多个子系统来支撑，这时就有按模块和功能把表划分出来的需求。其实，相对于垂直切分更进一步的是服务化改造，说得简单就是要把原来强耦合的系统拆分成多个弱耦合的服务，通过服务间的调用来满足业务需求看，因此表拆出来后要通过服务的形式暴露出去，而不是直接调用不同模块的表，淘宝在架构不断演变过程，最重要的一环就是服务化改造，把用户、交易、店铺、宝贝这些核心的概念抽取成独立的服务，也非常有利于进行局部的优化和治理，保障核心模块的稳定性  垂直拆分用于分布式场景。  水平拆分  上面谈到垂直切分只是把表按模块划分到不同数据库，但没有解决单表大数据量的问题，而水平切分就是要把一个表按照某种规则把数据划分到不同表或数据库里。例如像计费系统，通过按时间来划分表就比较合适，因为系统都是处理某一时间段的数据。而像SaaS应用，通过按用户维度来划分数据比较合适，因为用户与用户之间的隔离的，一般不存在处理多个用户数据的情况，简单的按user\_id范围来水平切分  通俗理解：水平拆分行，行数据拆分到不同表中， 垂直拆分列，表数据拆分到不同表中  问题 29:如何使用水平拆分数据库？  回答:使用水平分割拆分表，具体根据业务需求，有的按照注册时间、取摸、账号规则、年份等。  问题 30：什么是索引  回答：索引用来快速地寻找那些具有特定值的记录，所有MySQL索引都以B-树的形式保存。如果没有索引，执行查询时MySQL必须从第一个记录开始扫描整个表的所有记录，直至找到符合要求的记录。表里面的记录数量越多，这个操作的代价就越高。如果作为搜索条件的列上已经创建了索引，MySQL无需扫描任何记录即可迅速得到目标记录所在的位置。如果表有1000个记录，通过索引查找记录至少要比顺序扫描记录快100倍。  问题 31:索引的分类？  回答:1.普通索引  2.唯一索引  3.主键索引  4.组合索引  5.全文索引  问题 32：那些列上适合添加索引  回答：肯定在where条件经常使用  该字段的内容不是唯一的几个值  字段内容不是频繁变化  问题 33:MySQL数据引擎？  回答:使用的存储引擎 myisam / innodb/ memory  myisam 存储: 如果表对事务要求不高，同时是以查询和添加为主的，我们考虑使用myisam存储引擎. ,比如 bbs 中的 发帖表，回复表.  INNODB 存储: 对事务要求高，保存的数据都是重要数据，我们建议使用INNODB,比如订单表，账号表.  MyISAM 和 INNODB的区别  1. 事务安全（MyISAM不支持事务，INNODB支持事务）  2. 查询和添加速度（MyISAM批量插入速度快）  3. 支持全文索引（MyISAM支持全文索引，INNODB不支持全文索引）  4. 锁机制（MyISAM时表锁，innodb是行锁）  5. 外键 MyISAM 不支持外键， INNODB支持外键. (在PHP开发中，通常不设置外键，通常是在程序中保证数据的一致)  Memory 存储，比如我们数据变化频繁，不需要入库，同时又频繁的查询和修改，我们考虑使用memory, 速度极快. （如果mysql重启的话，数据就不存在了）  问题 34：Myisam注意事项  回答：如果你的数据库的存储引擎是myisam,请一定记住要定时进行碎片整理  问题 35:HasTable与HasMap？  回答:1.HashMap不是线程安全的  HastMap是一个接口 是map接口的子接口，是将键映射到值的对象，其中键和值都是对象，并且不能包含重复键，但可以包含重复值。HashMap允许null key和null value，而hashtable不允许。  2.HashTable是线程安全的一个Collection。  3.HashMap是Hashtable的轻量级实现（非线程安全的实现），他们都完成了Map接口，主要区别在于HashMap允许空（null）键值（key）,由于非线程安全，效率上可能高于Hashtable。  HashMap允许将null作为一个entry的key或者value，而Hashtable不允许。  HashMap把Hashtable的contains方法去掉了，改成containsvalue和containsKey。  注意: HashTable线程安全，HashMap线程不安全。  问题 36:悲观锁与乐观锁？  回答:悲观锁:悲观锁悲观的认为每一次操作都会造成更新丢失问题，在每次查询时加上排他锁。  每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会block直到它拿到锁。传统的关系型数据库里边就用到了很多这种锁机制，比如行锁，表锁等，读锁，写锁等，都是在做操作之前先上锁。  Select \* from xxx for update;  乐观锁:乐观锁会乐观的认为每次查询都不会造成更新丢失,利用版本字段控制  问题 37:乐观锁实现方式  回答:就是给数据增加一个版本标识，在数据库上就是表中增加一个version字段，每次更新把这个字段加1，读取数据的时候把version读出来，更新的时候比较version，如果还是开始读取的version就可以更新了，如果现在的version比老的version大，说明有其他事务更新了该数据，并增加了版本号，这时候得到一个无法更新的通知，用户自行根据这个通知来决定怎么处理，比如重新开始一遍。这里的关键是判断version和更新两个动作需要作为一个原子单元执行，否则在你判断可以更新以后正式更新之前有别的事务修改了version，这个时候你再去更新就可能会覆盖前一个事务做的更新，造成第二类丢失更新，所以你可以使用update … where … and version=”old version”这样的语句，根据返回结果是0还是非0来得到通知，如果是0说明更新没有成功，因为version被改了，如果返回非0说明更新成功。  问题 38:悲观锁按照使用性质划分？  回答:共享锁  共享锁：也称读锁，事务A对对象T加s锁，其他事务也只能对T加S，多个事务可以同时读，但不能有写操作，直到A释放S锁。  排它锁  排它锁（Exclusivelocks简记为X锁）：也称写锁，事务A对对象T加X锁以后，其他事务不能对T加任何锁，只有事务A可以读写对象T直到A释放X锁。  更新锁  更新锁（简记为U锁）：用来预定要对此对象施加X锁，它允许其他事务读，但不允许再施加U锁或X锁；当被读取的对象将要被更新时，则升级为X锁，主要是用来防止死锁的。因为使用共享锁时，修改数据的操作分为两步，首先获得一个共享锁，读取数据，然后将共享锁升级为排它锁，然后再执行修改操作。这样如果同时有两个或多个事务同时对一个对象申请了共享锁，在修改数据的时候，这些事务都要将共享锁升级为排它锁。这些事务都不会释放共享锁而是一直等待对方释放，这样就造成了死锁。如果一个数据在修改前直接申请更新锁，在数据修改的时候再升级为排它锁，就可以避免死锁。  问题 39:悲观锁按照作用范围划分？  回答:行锁  锁的作用范围是行级别，数据库能够确定那些行需要锁的情况下使用行锁，如果不知道会影响哪些行的时候就会使用表锁。举个例子，一个用户表user，有主键id和用户生日birthday当你使用update … where id=?这样的语句数据库明确知道会影响哪一行，它就会使用行锁，当你使用update … where birthday=?这样的的语句的时候因为事先不知道会影响哪些行就可能会使用表锁。  表锁  锁的作用范围是整张表。  问题40 :Java虚拟机原理  回答:所谓虚拟机，就是一台虚拟的机器。他是一款软件，用来执行一系列虚拟计算指令，大体上虚拟机可以分为  系统虚拟机和程序虚拟机， 大名鼎鼎的Visual Box、Vmare就属于系统虚拟机，他们完全是对物理计算的仿真，  提供了一个可以运行完整操作系统的软件平台。  程序虚拟机典型代码就是Java虚拟机，它专门为执行单个计算程序而计算，在Java虚拟机中执行的指令我们成为Java  自己码指令。无论是系统虚拟机还是程序虚拟机，在上面运行的软件都被限制于虚拟机提供的资源中。  Java发展至今，出现过很多虚拟机，做初Sun使用的一款叫ClassIc的Java虚拟机，到现在引用最广泛的是HotSpot虚拟  机，除了Sum意外，还有BEA的Jrockit，目前Jrockit和HostSopt都被oralce收入旗下，大有整合的趋势。  问题 41:Java堆  回答:堆内存用于存放由new创建的对象和数组。在堆中分配的内存，由java虚拟机自动垃圾回收器来管理。在堆中产生了一个数组或者对象后，还可以在栈中定义一个特殊的变量，这个变量的取值等于数组或者对象在堆内存中的首地址，在栈中的这个特殊的变量就变成了数组或者对象的引用变量，以后就可以在程序中使用栈内存中的引用变量来访问堆中的数组或者对象，引用变量相当于为数组或者对象起的一个别名，或者代号。  根据垃圾回收机制的不同，Java堆有可能拥有不同的结构，最为常见的就是将整个Java堆分为  新生代和老年代。其中新声带存放新生的对象或者年龄不大的对象，老年代则存放老年对象。  新生代分为den区、s0区、s1区，s0和s1也被称为from和to区域，他们是两块大小相等并且可以互相角色的空间。  绝大多数情况下，对象首先分配在eden区，在新生代回收后，如果对象还存活，则进入s0或s1区，之后每经过一次  新生代回收，如果对象存活则它的年龄就加1，对象达到一定的年龄后，则进入老年代。  问题 42:Java栈  回答:Java栈是一块线程私有的空间，一个栈，一般由三部分组成:局部变量表、操作数据栈和帧数据区  局部变量表：用于报错函数的参数及局部变量  操作数栈：主要保存计算过程的中间结果，同时作为计算过程中的变量临时的存储空间。  帧数据区:除了局部变量表和操作数据栈以外，栈还需要一些数据来支持常量池的解析，这里帧数据区保存着  访问常量池的指针，方便计程序访问常量池，另外当函数返回或出现异常时卖虚拟机子必须有一个异常处理表，方便发送异常  的时候找到异常的代码，因此异常处理表也是帧数据区的一部分。  问题 43:内存溢出解决办法  回答:1.设置堆内存大小 -Xms1m –Xmx10m  2.栈溢出 产生于递归调用，循环遍历是不会的，但是循环方法里面产生递归调用， 也会发生栈溢出，解决办法:设置线程最大调用深度  -Xss5m 设置最大调用深度  3.Tomcat内存溢出在catalina.sh 修改JVM堆内存大小  问题44 :如何防止内存泄露  回答:特别注意一些像HashMap、ArrayList的集合对象，它们经常会引发内存泄漏。当它们被声明为static时，它们的生命周期就会和应用程序一样长。  特别注意事件监听和回调函数。当一个监听器在使用的时候被注册，但不再使用之后却未被反注册。“如果一个类自己管理内存，那开发人员就得小心内存泄漏问题了。” 通常一些成员变量引用其他对象，初始化的时候需要置空。  问题 45:垃圾回收机制算法  回答:GC ：Garbage Collections 字面意思是垃圾回收器，对于一个Java程序员来说，一个对象的内存分配和回收是JVM自动进行的。那么我先说一下JVM的内存模型结构，它主要包括：栈内存、堆内存、方法区。堆内存分为新生代和老年代，新生代又划分为Eden区和survivor区。 GC一般就发生在堆内存中。  发起GC之前，它会先确定哪些对象可以被回收，有两种方法可以确定回收的对象，分别是引用计数法和可达性分析法。如果一个对象没有与任何的引用有关，就会被作为垃圾回收，这种方法实现简单，效率快，但它的缺点是没有办法解决循环引用的问题。而可达性分析法可以解决这个问题，它通过“GC Roots”作为对象进行搜索，来判断它与对象之间有没有可达路径。  确定可回收对象之后，jvm就开始进行垃圾回收，有四种算法，第一种是标记-清除法，它标记出回收对象后直接进行清除，这种算法容易造成内存碎片。第二种算法复制算法可以解决第一种算法的缺点，它是把内存分成了相等的两个部分，每次只是用其中的一部分，这部分内存用完后，把存活的对象移动到另一部分上，把已使用的内存一次清除掉，这种算法的缺陷就是降低了内存的使用率。接下来就是第三种算法标记-整理法，同样它解决了第二种算法的缺陷，它是把可回收的对象进行标记之后，把存活的对象挪向一端，清理掉端界以外的内存，这种算法相比较而言还是比较合理的，在进行垃圾回收的同时还能尽可能的使用内存。最后一种算法是分代收集算法，它也是目前JVM 进行GC通用的算法，它对堆内存中的新生代和老年代进行不同的回收算法，因为新生代每次都有大量的对象要被回收，所以适合使用复制算法，只进行少量复制操作即可。在实际中新生代并不是按照1:1来分配Eden区和Survivor区，它会划出较大的一块空间给Eden区，把Survivor区分为两块区，分别为 From Space和To Space,每次只是用Eden区和Survivor区其中的一块，假设使用的为From Space，当Eden区和From Space区的空间不足，它就会进行 一次GC，把存活的对象复制到To Space区，然后对Eden区和From Space区进行清理，依次类推，当对象躲过一次GC，他的年龄就会加一，直到15岁的时候就会被移动到老年代。老年代存放的一般是需要大量存储空间的对象，所以说它每次只回收很少的对象，比较适合编辑-整理法。  问题 46:redis是什么？  回答:redis是一种内存型、非关系型数据库；redis的数据会存储到硬盘上；它保存数据一种采用的是rdb(快照的形式进行数据的保存，是redis默认开启的方案，它将内存中的数据都保存到磁盘上，放到rdb文件中，重新启动服务器是将rdb的内容重新取出来,再回复到内存中。还有一种是aof默认是不开启的，因为他会频繁的操作磁盘，每秒钟同步一次，所以默认是不开启的。开启之后数据的恢复就是用aof。  问题 47:redis主要有以下几个特点？  回答:1、速度快  Redis是用C语言实现的，所有数据存储在内存中。  2、持久化  Redis的所有数据存储在内存中，对数据的更新将异步地保存到磁盘上。  3、支持多种数据结构  Redis支持五种数据结构：String、List、Set、Hash、Zset  4、支持多种编程语言  Java、php、Python、Ruby、Lua、Node.js  5、功能丰富  除了支持五种数据结构之外，还支持事务、流水线、发布/订阅、消息队列等功能。  6、源码简单  约23000行C语言源代码。  7、主从复制  主服务器（master）执行添加、修改、删除，从服务器执行查询。  8、高可用及分布式  Redis-Sentinel（v2.8）支持高可用  Redis-Cluster（v3.0）支持分布式  问题 48:redis持久化方式？  回答:redis的数据会存储到硬盘上，它保存数据一种采用的是rdb(快照的形式进行数据的保存，快照是采用一种机制保存，这种机制你可以自己在redis的配置文件中进行配置，隔多长时间保存多少数据,这种方法是比较快了，数据会被保存到硬盘上，当redis启动的时候，硬盘上的数据会在写到数据库中，这种方法是大大提高了数据的存储和读取的效率。但是性能上还是会有一定的影响，毕竟他还要去占用cpu去做保存快照的操作；redis保存数据的另一种方法就是采用aof集群的方式，它会使每个服务器内都有对方服务器上的数据，这中方式避免了像rdb那种在特定时间内用快照保存数据的机制，减少内存的占用，从而提高性能。而且这种方法还有另一个好处就是，当一个服务器荡掉以后，数据不会因此丢失，而是当你在操作数据的时候会从其他的服务器上读到你想要操作的数据。aof默认是不开启的，因为他会频繁的操作磁盘，每秒钟同步一次，所以默认是不开启的。开启之后数据的恢复就是用aof。  RDB 和 AOF ,一般怎么应用呢？一般来说,如果想达到极高的数据安全性， 应该同时使用两种持久化功能。如果你非常关心你的数据,但仍然可以承受数分钟以内的数据丢失， 那么你可以只使用 RDB 持久化。有很多用户都只使用 AOF 持久化， 但我们并不推荐这种方式： 因为定时生成 RDB 快照（snapshot）非常便于进行数据库备份， 并且 RDB 恢复数据集的速度也要比 AOF 恢复的速度要快， 除此之外， 使用 RDB 还可以避免之前提到的 AOF 程序的 bug 。  问题 49:分布式集群？  回答:Redis集群中至少应该有三个节点。为了实现容错机制（所有的redis节点彼此互联(PING-PONG机制)每个节点都有一个FAIL，因为是通过ping-pong机制来进行互联，所以当一个节点没有返回pong的时候，标记为pfail,通过集群中超过半数的节点检测失效时才生效变为PFAIL,就代表本节点失效），要保证集群的高可用，需要每个节点有一个备份机。Redis 集群中内置了 16384 个哈希槽，集群所搭建的服务器平均分配了这16384个hash槽，所以每当我们添加一个key-value 时，redis 先对 key 使用 crc16 算法算出一个结果，然后把结果对 16384 求余数，这样每个 key 都会对应一个编号在 0-16383 之间的哈希槽，然后重定向到包含这个hash槽的节点，继续添加的时候，又会算相应的hash槽，概率基本上是每添加一个key跳转到不同的节点，数据均匀的分布在不同的节点上，这样实现了每个节点都有一定数量的数据，如果主节点挂了那从节点就顶上来，这样就实现了高可用。  问题 50:什么是高可用？  回答:高可用（High Availability），是当一台服务器停止服务后，对于业务及用户毫无影响。 停止服务的原因可能由于网卡、路由器、机房、CPU负载过高、内存溢出、自然灾害等不可预期的原因导致，在很多时候也称单点问题。  问题 51:Redis数据淘汰策略？  回答:Redis在已使用内存达到设定的上限时，提供了6种数据淘汰策略（也就是maxmemory-policy可能的值）：  在已经设置了过期时间的数据中：  1、淘汰最近未使用的一些数据，也就是volatile-lru策略  2、随机淘汰一些数据，也就是volatile-random策略  3、淘汰距离过期时间最近的一些数据，也就是volatile-ttl策略  在全部数据中  4、淘汰最近未使用的一些数据，也就是allkeys-lru策略  5、随机淘汰一些数据，也就是allkeys-random策略  不淘汰数据  6、内存使用达到设定的内存上限时，用户试图存储新数据时会直接返回错误，也就是noeviction策略。  问题 :  回答:  问题 :  回答: |
| **人资问题(至少30个)** |

|  |
| --- |
| 问题1：你得离职原因是啥  回答：主要原因是因为，公司的项目运作比较稳定，公司在19年的规划要做开展其它的业务，所以不需要太多的技术岗。  问题2：那个是主动离职、还是被动离职  回答：算是主动吧，因为领导也沟通征求我们意见了，我当时想如果在公司维护项目，是挺稳定的工作，可能对自己技术方面没有提升，所以选择离职。  问题3：公司是做什么业务的？  回答：公司有自主研发的项目，也有承接外包业务。领域涉及金融、电商、医疗等等  问题4：你的职业规划？  回答：首先是自己会在技术的岗位一直发展下去，同时也通过自己的学习和工作积累让技术方面不段的提升，另一方面，也希望自己先把工作稳定下来，先更好的工作开展起来，能过自己技术的提升，工作能力的提升，也让自己在工作中发展更好。  问题5： 你在xxx有限公司这家公司工作两年是吧？  回答：对，两年时间，其实在这个公司让我也成长挺多的。  问题6：加班可以接受吗？  回答：可以接受，做这一行加班很正常，尤其是项目要上线的时候，完全可以接受。  问题7：能接受出差吗？  回答：可以接受，这都是工作中的一部分。  问题8：你的优点？  回答：我的优点是抗压能力比较强，适应环境能力比较快，能狗很快的适应环境并融入团队。  问题9：你的缺点？  回答：我的缺点是不太喜欢运动，休息的时候也能对着电脑坐一天。  问题10：上份工作的薪资是多少？税前还是税后?  回答：12k,税前，税后大概一万多点  问题11：期望薪资多少？  回答：15k  问题12：有其他补助吗?  回答：有交通补和餐补，一个月这两个加起来600块钱  问题13：缴纳五险一金了吗？  回答：没有，因为当时比较年轻，想多拿一些钱，折现了  问题14：现居住地址？  回答：住在昌平回龙观，来咱公司不到一个小时，挺快的  问题15：如果你来到我们公司如何来提升自己？  回答： 1.首先对本公司所属的业务进行了解和掌握本公司的工作流程以及业务体系；  2.在做一个项目时更加注重用户体验和完善的服务流程；  3.培养自己敏锐的市场洞察能力，提高自己的用户需求转换能力；  4.跟团队人员进行讨论，彼此分享一些比较好的想法相互进步；  5.关注相关行业的书籍以及新闻，提高自己的行业知识；  6.不断尝试新的想法和方法，不断自学天提升专业知识；  7.提高自己的执行能力；  问题16：现在处于离职状态还是在职状态？  回答： 1.自己正在办理离职；  2.已经离职；  问题17：你有什么想要问的吗？  回答：咱们公司是做什么业务的？若是有幸能够入职咱们公司我最主要是负责哪些方面的工作？  问题18：你的学历学信网可以查到吗？  回答：不好意思，学信网查不到，可以在民教网查到？  问题19：我看你这个学校是本科为什么查不到呢？  回答：因为我上的是三本，而且这个学院是学校和企业联合开办的，所以只能在民教网上查  问题20：休息时间和工作时间发生冲突时，怎么办?  回答: 我会把工作处理完了再去考虑休息时间，因为我觉得热爱生活才会热爱工作嘛!(父母都很支持我的工作，我想他们应该可以理解的)  问题21：你的朋友都怎么评价你?  回答：细心、认真、负责、乐于助人  问题22：如果项目经理给你分配的任务过重，你怎么办?  回答：我会申请项目经理重新分配一下，给我分的任务可能赶不上进度  　如果非常相信我：非常感谢您的信任，这样吧，我尽自己最大的努力，如果在工作中遇到些困难希望可以得到一些帮助。您看可以吗?  问题23：你通常如何处理別人的批评？  回答：善意的批评要接受。当别人指出自己身上的缺点或者自己做错了事情要挨骂的时候，千万不要动怒。如果是善意的批评，对方只是希望我们可以改正，这不是无端指责，因而不要去顶撞别人，而是要本着“有则改之无则加勉”的心态来对待。  问题24：你最擅长的技术方向是什么？  回答：我最擅长的是spring 全家桶,因为我上家公司一直用的就是这个技术,而且spring也是一种发展趋势,当然我还是有好多地方需要学习的.  问题25：谈谈你对跳槽的看法？  回答：（1）正常的"跳槽"能促进人才合理流动，应该支持；（2）频繁的跳槽对单位和个人双方都不利，应该反对。  问题26：工作中你难以和同事、上司相处，你该怎么办？  回答：①我会服从领导的指挥，配合同事的工作。  ②我会从自身找原因，仔细分析是不是自己工作做得不好让领导不满意，同事看不惯。还要看看是不是  为人处世方面做得不好，如果是这样的话我会努力改正。  ③如果我找不到原因，我会找机会跟他们沟通，请他们指出我的不足，有问题就及时改正。  ④作为优秀的员工，应该时刻以大局为重，即使在一段时间内，领导和同事对我不理解，我也会做好本职工作，虚心向他们学习，我相信，他们会看见我在努力，总有一天会对我微笑的。  问题27：希望和什么样的领导合作?  回答：作为新员工，没机会选择领导，应该自己适应领导。  问题28：你这是第几家面试?  回答：我到贵公司是第二次面试，昨天在XX公司面试过，感觉不错，他们说明天之前给我答复。  问题29：和上级发生矛盾怎么处理?  回答：我个人性格很开朗，也比较随和，善于沟通。我想我会和我的上级相处的很融洽。当然工作中也难免会发生分歧，我会私底下和上级沟通。大家都是年青人如果主动的去沟通，去解决就不会有什么问题了。  解决不了：如果我觉得对公司不会有损失的话，我会保留意见的，当然我相信通过沟通问题应该可以解决吧。  问题30：为什么的年龄和身份证不一致呢？  回答：当时计划生育查的比较严，户口晚上了几年。  ........ |