葵花宝典之吊打Java面试官

说说你的职业规划？

答：初级工程师->中级工程师->高级工程师。说说对每个级别的理解，每个级别应该具备怎么样的能力。

自我介绍：

介绍一下你的项目吧：

项目介绍：

1. 项目的功能，主要模块有什么？
2. 项目采用了哪些技术栈？简单说说用这些技术栈做什么，并且为什么选择他们？
3. 自己负责哪些模块？有什么难点和新点
4. 遇到了什么困难？
5. 等待提问，并解释。

项目介绍

在线办公系统是一款面向公司内部工作人员的在线协同办公小程序。采用目前主流的前后端分离开发模式，主要核心业务包括人脸签到，公告通知，会议管理，视频会议等。项目构建在Spring Boot和SSM框架之上，数据层采用了MySQL和MongoDB两种关系型数据库，MongoDB用来存储公告消息，因为消息模块要存储的数据量比较大，普通MySQL单表超过2000万记录，MySQL数据库就难以支撑。而 MongoDB能够储存TB级别的数据，符合消息模块的存储要求。缓存层采用的是Redis，在这里我用来缓存用户的Token令牌，主要用来解决令牌过期的问题。消息中间件采用的是RabbitMQ，我将系统消息保存在消息队列中，在用户登陆系统的时候，从消息队列接收该用户的系统消息，把系统消息存储在数据库中，最后消息队列中的该条消息自动删除。这样做的目的是通过中间件来实现削峰填谷。

1.说说人脸签到模块？/用Docker做什么？/用JSOUP做什么？

人脸签到模块，我通过Docker容器部署了一个人脸识别程序来识别员工上传的照片是否为本人，在此基础上还增加了疫情风险等级判断，如果该签到地点是高风险地区，则异步向公司领导，HR发送告警邮件。疫情风险等级判断我是这样实现的，由于员工签到的时候，前端会将员工的定位信息和照片发送给JAVA程序，我们就可以利用这个定位信息。本地宝这个网站提供了新冠疫情地区风险等级的查询，通过输入地址，就能看到具体的风险等级。由于本地宝并没有提供Web接口让我们调用，所以我就通过URL地址传参的方式获取本地宝返回的响应。然后通过JSOUP从HTML中解析出的风险等级信息。

2.用消息中间件来做什么?

答：用消息中间件来削峰填谷。如果一个电商系统有800万注册用户，那么系统发出一条公告消息之后，意味着要在消息表中插入800万条消息记录，每条记录是发给某位用户的公告消息。瞬间数据库的负载达到顶峰，数据库被大量的写入操作占用，导致电商系统正常的业务无法展开。

3.你用MongoDB做什么？/为什么不直接用MySQL，而选择MongoDB呢？

答:因为消息模块要存储的数据量太大，普通MySQL单表超过2000万记录，MySQL数据库就难以支撑，所以我们要换能存储海量数据的数据库产品，于是MongoDB数据库就进入到我的视线里。MongoDB适合存储海量低价值的数据，正好符合消息模块的存储要求。

4.但是消息数据日积月累，MongoDB也会有撑不下来的时候。怎么解决呢？

答：系统消息保存在消息队列中，我用它来做削峰填谷，系统消息最终还是要存储在数据库上面。于是我是这样设计的，将公告消息先写入消息中间件RabbitMq中。然后在用户登陆系统的时候，用异步线程从消息队列MQ中，接收该用户的系统消息，最后把系统消息存储在数据库中，最后消息队列MQ中的该条消息自动删除。这么设计之后，所有用户不可能同时登陆Web系统，所以我们就把往数据库中写入系统消息的任务，变成了错峰写入。

5.为什么选择 RabbitMQ而不选择Kafka？

答：但是消息收发的可靠性上，Kafka不如RabbitMQ。RabbitMQ还有另外一个优点，就是既支持消息异步收发，又支持同步收发。虽然我们现在大部分的场景对应的是消息异步收发，但是有的场合要支持消息的同步收发，这时候RabbitMQ能适应各种业务场景的优点就显现出来了。所以在项目立项的时候，选择RabbitMQ是最稳妥的方案。

6.刚才你说为什么RabbitMQ可靠性更好，那 RabbitMQ是如何保证消息可靠性的呢？

答：1、就是消息持久化。持久化可以防止在异常情况下丢失数据。除了消息持久化之外，甚至交换器和队列都能持久化。

7.你还说RabbitMQ能同时支持同步收发和异步收发，你的项目有体现吗？

答：有的，由于小程序和后端项目之间并不是长连接，所以后端项目异步方式接收到队列中的消息，也无法推送给移动端的小程序。我们可以让后端的Java项目采用同步的方式接收队列中的消息。在移动端，我们创建定时器，然后向后端Java项目发出轮询请求。后端Java项目接收到轮询请求之后，用同步方式接收队列中的消息，然后把消息存储在MongoDB上面，最后向小程序返回接收了多少条新消息，移动端则弹出提示框告知用户有信息的消息通知。

8．如何解决消息重复消费的问题？

答：我的项目中有遇到过这个问题。小程序每隔5分钟轮询是否有新的消息，如果积压的消息太多，Java系统没有接收完消息，这时候新的轮询到来，就会产生两个消费者共同接收同一个消息的情况，这会造成数据库中添加了重复的记录。我是这么解决的，我给每条MQ消息设置了唯一的UUID值，第一个消费者把消息保存到数据库，那么第二个消费者就无法再把这条消息保存到数据库，解决了消息的重复消费问题。

9.说一说令牌过期难题？

答：我是这么想的，因为我给每个token都设置了过期时间。假设token的过期时间是15天，用户在第14天的时候，还可以免登陆正常使用系统。但是第15天，用户的token就过期了，这个时候用户就得重新登录系统了。这样用户的体验感就不是很好，无论你有什么使用系统，每隔一段时间你就得重新登录。所以我想要实现只要用户间隔操作时间不超过15天，系统就不需要用户重新登录系统。

10.那你是怎么实现的？

答：我这里用到的是token缓存方案，我把token缓存到了Redis中，然后把token的过期时间设置为正常token过期时间的一倍，然后根据情况刷新token的过期时间。第一种情况：Token失效，缓存也不存在的情况 ：当第15天，用户的Token失效以后，到Redis查看是否存在缓存的Token，如果这个Token不存在于Redis里面，就说明用户的操作间隔了15天，需要重新登录。

第二种情况：Token失效，但是缓存还存在的情况。如果Redis中存在缓存的Token，说明当前Token失效后，间隔时间还没有超过15天，不应该让用户重新登录。所以要生成新的 Token返回给客户端，并且把这个Token缓存到Redis里面。

11客服端怎么更新令牌呢？/客户端怎么知道这次响应带回来的Token是更新过的呢？

答：只要用户成功登陆系统，当后端服务器更新Token的时候，就在响应中添加Token。客户端那边判断每次Ajax响应里面是否包含Token，如果包含，就把Token保存起来就可以了。

12.怎么在响应中添加令牌？/有用过AOP吗？/ThreadLocal有用过吗？

答：这里我用到了过滤器，AOP和ThreadLocal。首先，我使用了一个过滤器拦截所有的HTTP请求，一方面它会把请求中的Token字符串提取出来，封装成对象交给Shiro框架；另一方面，它会检查Token的有效性。如果Token过期，那么会生成新的Token，分别存储在一个ThreadLocal变量里（ThreadLocalToken）和Redis中。之所以把新令牌保存到ThreadLocalToken里面，是因为我使用了AOP切面类传递这个新令牌 。

虽然过滤器中有 doFilterInternal() 方法，可以得到响应并且写入新令牌。但是我觉得这样做比较麻烦，首先要通过IO流读取响应中的数据，然后还要把数据解析成JSON对象，最后再放入这个新令牌。我则定义了AOP切面类 ，拦截所有Web方法返回的 R对象 ，然后在 R对象里面添加新令牌。但是过滤器和AOP切面类之间没有调用关系，所以我们很难把新令牌传给AOP切面类 。这里我想到了ThreadLocal，只要是同一个线程，往ThreadLocal里面写入数据和读取数据是完全相同的。在Web项目中，从过滤器到AOP切面类 ，都是由同一个线程来执行的，中途不会更换线程。所以可以放心的把新令牌保存都在 ThreadLocal 里面， AOP切面类可以成功的取出新令牌，然后往R对象里面添加新令牌即可。

ThreadLocalToken 是我自定义的类，里面包含了 ThreadLocal 类型的变量，可以用来保存线程安全的数据，而且避免了使用线程锁。

13.你用到了过滤器，你能说说过滤器和拦截器的区别吗？

答：过滤器是Javaweb三大组件之一（servlet、Listener、Filter）。一般完成一些比较通用的工作。比如，登录检验，统一编码处理，敏感字符处理。主要通过实现Filter接口实现，主要有init方法（初始化方法，web服务器启动，创建filter时调用，只调用一次）,doFilter（拦截到请求时，调用该方法，可以调用多次）,dest ory方法（服务器关闭时调用，只调用一次）。

拦截器跟过滤器有点类似，是spring框架提供的。主要通过HandlerInterceptor接口实现，主要有preHandle,postHandle,afterHandle分别在目标方法执行前执行，目标方法执行后执行，视图渲染结束后执行，最后执行。过滤器的执行在dispatchersevlet之前，拦截器在其之后，过滤器能拦截所有资源请求，拦截前只能拦截Spring中的所有资源。

什么叫过滤器链？

答：一个web服务中，可以配置多个过滤器，这多个过滤器就形成了一个过滤器链。顺序是按照类名的首字母排序的，也可以使用@order注解

14.AOP相关问题？

15.你简单说说ThreadLocal？

答：ThreadLocal是多线程中为了解决线程安全的一个操作类，他是线程安全的。它为每个线程都分配了一个独立的线程副本从而解决了变量并发访问冲突的问题。它还同时实现了线程内的资源共享。

15.你可以说说ThreadLocal的实现原理吗？

答：ThreadLocal本质上是一个线程内部存储类，让多个线程只操作自己内部的值。从而实现线程数据隔离。我记得我看过源码：ThreadLocal里面有个内部类ThreadLocalMap,这个类里面有一个table数组，这个数组是真正存数据的地方。当我们调用set方法的时候，ThreadLocal以自己作为key,资源对象作为 value存储在ThreadLocalMap里面。Get方法和Remove的方法也是一样的。

15.ThreadLocal会导致内存溢出，你了解吗？

答：了解的。ThreadLocalMap的key被设计为弱引用，会被GC。但是只有key会被释放，value不会。因为value是一个强引用。因为在使用的时候都会用static修饰（既强引用），所以不会被释放。因此建议主动使用remove释放Key,以防止内存溢出。

16.简单介绍一下Shiro?\为什么选择Shiro而不选择spring security?

答：Shiro是Java领域非常知名的认证（ Authentication ）与授权（ Authorization ）框架，用以替代JavaEE中的JAAS功能。相较于其他认证与授权框架，Shiro设计的非常简单，所以广受好评。任意JavaWeb项目都可以使用Shiro框架，而Spring Security必须要使用在Spring项目中。所以Shiro的适用性更加广泛。

17.解释一下RBAC权限模型？

答：RBAC的基本思想是，对系统操作的各种权限不是直接授予具体的用户，而是在用户集合与权限集合之间建立一个角色集合。每一种角色对应一组相应的权限。一旦用户被分配了适当的角色后，该用户就拥有此角色的所有操作权限。这样做的好处是，不必在每次创建用户时都进行分配权限的操作，只要分配用户相应的角色即可，而且角色的权限变更比用户的权限变更要少得多，这样将简化用户的权限管理，减少系统的开销。

18.什么是JWT?/知道cookie,session吗？跟他们相比有什么优势？

答：JWT的全称是JSON Web Token。主要有三个部分组成有效载荷，信息，签名。

19.你的视频会议功能具体是怎么实现的？

20.跟我说说你是怎么实现多设备登录的实现方案？

21.浏览器插件实现扫码登录是怎么实现的？跟我说一下扫码登录

答：主要有三个部分，分别是移动端，PC端，服务端。1.首先PC端会向服务端发起生成二维码的请求，此时服务端会生成一个二维码id（类似uuid是唯一的），并与设备信心绑定，二维码id作为 key,状态作为value。然后将二维码id发送给PC端。PC端生成二维码待手机扫描，并每隔一段时间向服务端轮询二维码的状态。这个时候二维码的状态是待扫描

2.移动端扫描二维码，得到二维码id，并将token和二维码id作为参数发送给服务端，服务端生成临时token。状态变为待确认，并将临时token发送给移动端。移动端的页面变为待确认，此时移动端将临时token作为凭证进行确认。服务端验证通过，临时token失效。状态更新为已登录，并生成一个token发送给PC端，PC端的后续请求就通过这个令牌。

22.并行和并发的区别？

答：并发是指在同一时间段内具有处理多个任务的能力。比如说：多个线程被一个cpu轮流执行，这就是并发。并行是指在同一时刻具有处理多个任务的能力，比如说多个线程被分配到多个cpu执行，这就是并行。

23.进程，线程，协程的区别？

答：进程是应用程序的启动实例，进程包括线程且至少包含一个线程。线程比进程更加的轻量，因为线程的上下文切换成本比进程的上下文切换成本要低。协程是比线程还要轻量，协程不受操作系统的调度，协程的调度是通过用户程序实现的，调度器会通过调度策略将协程分配给线程执行。

24.Java中创建线程有几种方式？分别是什么？/go创建协程呢？

答：四种。分别是继承Thread，实现runnable接口，实现callable接口，线程池创建。

协程的创建是通过go关键字。

25. runnable，callable两个接口创建线程有什么不同？

答：runnable没有返回值，callable有返回值。对异常的处理方式也不一样，runnable可以跑出异常，但是不同捕获异常。Callable不仅可以抛出异常，还可以捕获异常。

26.线程包括哪些状态？状态之间如何变化？

答：线程的状态包括新建，可执行，终结，阻塞，等待。有时限等待6种。

1.当一个线程对象被创建但是还没有执行start方法的时候，线程为新建状态。当执行了start方法之后，转化为可执行状态。当run方法里面的程序执行完之后，就转化为终结状态。

2.其中，如果线程获取锁失败，将进入阻塞队列里进行阻塞，状态由可执行变为阻塞状态。当持有锁的线程释放锁之后，会按照一定的规则唤醒阻塞队列里面的线程，唤醒后变为可执行状态。

3.当条件不满足时，调用了wait方法，此时线程由可执行状态变为等待状态，当其他持有锁线程调用notify方法或notifyall方法，会恢复为可行性状态。

4.还有一种情况就是调用了sleep方法，线程会从可执行状态变为有时限等待状态。超时时间到后会自转化为可执行状态。

27.wait方法和sleep方法有什么区别？

答: sleep是thread的静态方法，线程执行sleep后在等待相应时间后醒来。Wait是Object的成员方法，每个对象都有。Wait需要notify唤醒，如果不唤醒将会一直等待下去。

28.新建T1、T2、T3三个线程，怎样才能让他们顺序执行？

答：可以使用join方法，让T3调用T2，T2调用T1。

29.start方法和run方法有什么区别？

答：start是用来启动线程的，只能调用一次。Run方法封装了线程要执行的代码，可以被执行多次。

30.怎么中断一个正在运行的线程？

答：stop方法，但是不推荐，已经被弃用了。Interrupt方法。

31.线程池的种类有哪些？

答:有4种，分别是：newFixedthreadpool,创建固定线程数的线程池（适合任务量已知的情况），newsingelthreadpool创建单例化的线程池，只有一个线程来执行任务，保证所有任务按照指定顺序执行（适合用于按照指定顺序执行的任务），newcachethreadpool可缓存线程池，核心线程数为0，适合任务比较密集的情况。Newscheduledthreadpool可以执行延迟任务的线程池，支持定时及周期性任务执行。

32.线程池的核心参数有哪些？

答：有7个。分别是线程池的核心线程数量（corepoolsize）、最大线程数量（maximumpoolsize）、keepalivetime(当线程数大于核心线程数时，多余的空闲线程的最长生存时间)、时间单位（timeunit）、blockingquene（任务队列）、threadfatory(线程工厂)、RejectExecutionHandler拒绝策略。其中，当新任务来到时会先判断当前运行的线程数是不是大于核心线程数，如果是，任务则会放入这个blockingquene（任务队列）中。如果满了，则会创建救急线程执行。

拒绝策略有四种，1直接抛异常2当前调度者执行任务3丢弃当前任务4丢弃最早排队的任务，默认是直接抛异常。

33.线程池中有哪些常见的阻塞队列？

答：四种，arrayblockqueue(基于数组),linkedblockqueue（基于链表）,delayworkqueue（优先级队列，保证每次出队的任务都是当前队列中执行时间最靠前的）,synchronousqueue（不存储元素的阻塞队列，每个插入操作必须等待一个移出操作）

33.synchronized了解吗？

答：了解，synchronized是java的一个关键字，是同步的意思。主要是用来解决多个线程之间访问资源的同步性，它能够保证被synchronized修饰的代码块或者方法在同一个时刻只有一个线程执行。

34.说一下synchronized的底层实现？

答：它的底层是通过Monitor(监视器)来决定当前线程是否获得了锁。Monintor对象存在于每个java对象的对象头中。Mointor内部维护了三个变量，分别是ower表示目前持有锁的线程，只有一个线程能获得锁。waitset表示等待队列，entrylist表示阻塞队列。

1.并发事务会带来什么问题？具体说一下

答：脏读，不可重复读，幻读。脏读：一个事务修改数据但是还没有提交至数据库，此时另外一个事物读取到了这个还没有提交的数据。这就是脏读。不可重复读：一个事务多次读取同一个数据，在读取两个数据之间另外一个事务修改了这个数据。导致两次读取同一个数据不一致，这就是不可重复读。幻读就是一个事务读取了几行数据，此时有另外一个事务插入了几条数据，导致前后查询的数据行数不一致，这就是幻读。

2.那 Mysql是怎么解决这个问题？

答：对事务进行隔离，Mysql支持四种事务隔离级别。分别是未提交读，已提交读，可重复读，串行化。默认可重复读。

3.事务的特性说一下？

答：一致性，原子性，持久性，隔离性。原子性是指：一个事务要么全部执行，要么全不执行，不会停顿在某个地方。如果事务在执行过程中发生了异常就会回滚到事务开始的地方，就好像事务没有执行过一样。持久性是指事务处理结束后，对数据的修改是永久的，即便系统故障也不会影响。一致性是指事务执行前后，数据是一致的，就比如A，B各有五百块，A给B转了500块后，总的金额还是1000块，不会多也不会少。隔离性是指防止多个并发事务同时操作同一个数据导致数据不一致，隔离让让每个事务都有自己的数据空间，对其他的事务隔离的，不会互相干扰。

4.这4个特性分别是怎么实现的？

答：持久性是通过redolog(重做日志)实现的，原子性是通过undolog（回滚日志）实现的。隔离性是通过MVVC实现的，一致性是通过持久性，原子性，隔离性实现的。

1. 解释一下MVVC?

答：MVVC的意思是多版本并发控制。它通过维护数据的多个版本，使得事务对数据的读写没有冲突。底层有三个部分：隐藏字段，undolog日志，视图。隐藏字段在每个表中都有，有一个事务 id和一个回滚指针，用来记录上一个版本的地址。Undolog主要记录回滚日志 ，存储老版本数据，在内部会形成一个版本链，在多个事务并行操作某一行记录时，记录不同事务修改数据的版本，通过回滚指针形成一个链表。视图解决的是一个事务查询选择版本的问题。通过内部定义的匹配规则和当前的事务id选择对应的版本的数据。不同隔离级别快照读是不一样的，最终访问的结果也不一样。RC级别每一个执行快照读是生成视图，RR级别仅仅在事务第一次执行快照读时生成视图。

1. 什么是当前读和快照读？/二者之间有什么区别？

答：当前读是读取数据的最新版本，读之前会加锁。快照读读取的是数据的可见版本，可能是历史数据，不加锁。

8.匹配规则知道吗？

答：我记得有四个字段，分别是当前活跃事务id集合，最小的活跃事务id，预分配事务id，视图创建者的事务id.当前事务id等于视图创建者的事务id是可以访问。当前事务id小于最小事务id可以访问。大于最大事务id不可以访问。当前事务id大于最小事务id小于最大事务id时且不在活跃集合里面是允许访问的，说明数据已经提交了。

9.怎么定位慢查询？

答：可以用一些监控工具，比如说skywalking.它可以看到哪些接口比较慢，可以分析这个接口哪个部分比较慢，还能看到sql的具体执行时间，我们就可以定位哪个sql有问题。如果没有监控工具的话，mysql提供了慢日志查询功能，可以在配置文件开启慢日志，可以设置sql超过多少时间后被记录到日志中去。

Show\_query\_log:标记慢查询日志是否开启的参数，默认是OFF，默认不开启

Show\_query\_log\_file:标记存放慢查询日志文件的完整路径。

Long\_query\_time:控制慢查询的时间阈值参数，我用的两秒。

Long\_queries\_not\_using\_indexes:标记是否记录未使用索引的查询，默认是关闭的。

Mysqldumpslow工具解读慢查询日志

Mysqldumpslow -s (什么方式？c=记录次数，t=时间，) -t 数据条数 -g 正则匹配模式 慢查询日志地址

10.如果这个sql很慢该怎么分析呢/

答：通过explain指令查看sql的执行情况，首先我们看key和ley\_len两个字段，检查一下是否命中了索引，如果本身已经添加了索引，那么要分析一下索引是否失效了。还有就是type类型，是不是index和all,是否走了全索引扫描和全盘扫描。最后就是extra建议，看看是否回表了。回表了可以尝试添加索引和修改返回的字段。

11.索引是什么？

答：索引就是一种高效获取数据的数据结构，就像我们书籍的目录一样，可以快速定位到我们想要获取的数据。

12.索引的底层了解过吗？

答：了解的，底层就是通过B+树实现的。B+树阶数更多，路径更短。非叶子节点上存储指针，叶子节点上存储数据。叶子节点是一个双向的链表，便于区间查询和扫库，它的磁盘io次数更低。

13.B+树和B树有什么区别？

答：B树上叶子节点非叶子节点都会存储数据，B+树只在叶子节点上存储数据，B+树查询效率更稳定。范围查询B+树的效率高于B树。

14.聚簇索引和非聚簇索引？

答：聚簇索引就是指数据和索引存放在一块，比如说叶子节点存放的是数据，比如主键。非聚簇索引是指数据和索引分开存放。叶子节点只存放数据的地址而不存放数据本身。

15.回表查询呢？

答：就是通过非聚簇索引找到主键值，再通过主键值找到聚簇索引中叶子节点存放的数据。

16.覆盖查询呢？

答：覆盖查询就是查询的字段刚好是索引的字段，这样就不会回表查询了，效率高。

17.说说你平时对sql的优化？

答：

1.任何时候都不要讲表属性设置为null，容易引起问题（null的长度并不是0，null只能通过is null来查询 ==、！=null都会失效。索引也是不同is null 会失效。Null跟谁计算都是自己，升序在第一个，降序在最后。聚合函数不会加上null）

2.使用存储所需要的最小数据类型

3.选择简单的数据类型

4.存储小数直接选择decimal

5.尽量避免使用text和blob(二进制)

6.不要使用枚举类型。

7.使用select语句的时候务必声明字段，不要使用select \*。

8.聚合查询尽量使用union all 而不是union因为union会多过滤一次。

9.如果是表关联的话，尽量使用inner join,不要使用left join\right join

7.索引加的不要，效果可能适得其反。

一．索引加的正确但是没有写出适用索引的查询语句

I．字符串类型在查询时没有使用引号，不会使用表的索引

II.where条件左边字段参与了函数或者数学运算，不会使用表索引

III．联合索引最左前缀顺序不匹配，不会使用表索引

二．索引加不好：

不再使用的索引没有及时的删除：空间浪费，插入删除效率受影响。

索引选择性太低，索引列的意义不大。索引选择性=不重复的索引值/表记录数，选择索引选择性高的。

列值过长，可以选择部分前缀作为索引（区分度高的情况下），而不是整列都加上索引。（网站）

表记录少就不要添加联合索引啦（1000行）

存在联合索引的情况下，再去对前缀部分加索引是没有意义的。

一张表中间建立的索引过多，根据业务创建并不是越多越好。太多会浪费空间，影响额外的查询效率。不应该超过五个。

18.数据库锁的分类？

答：按照锁数据的粒度分：全局锁，表级锁和行级锁。

全局锁：使用了全局锁的话，整个数据库就处于制度状态了。会话断开的时候，全局锁会释放。主要用来全库逻辑备份。

行级锁：开销大，加锁慢，粒度最小，可能会死锁，并发度最高。

表级锁：开销小，加锁快，不会死锁，粒度大。

//共享锁：select … lock in share mode;

//排它锁：select…for update

行级锁分为：记录锁：也就是仅仅把一条记录锁上。间隙锁：锁定一个范围，但是不包含记录本身。Next-key lock:锁定一个范围，并且锁定记录本身。插入意向锁：一个事务在插入一条记录的时候，需要判断该记录是否加了间隙锁。如果有，插入操作就会发生阻塞，直到拥有间隙锁的事务释放了间隙锁，在此时会生成一个插入意向锁，表明现有有事务想在某个区间插入数据，但是现在处于等待状态。

表级锁又分为：表锁，元数据锁，意向锁。表锁不仅限制其他线程对表读写还会限制本线程接下来的读写。元数据锁:对一张表进行操作时，加的是元数据读锁，为了防止其他的线程对这个表结构进行更改。意向锁：为了判断表里面是否有记录被加锁了。

按照数据的锁定方式分：乐观锁和悲观锁

19．行锁的并发性能远高于表锁，但是要小心粒度升级

InnoDB只有在通过索引条件检索数据时使用行级锁，否则使用表锁。这样就会出现两个事务查询一个表里不同的数据出问题，因为升级为表级锁了。得加索引。

对于UPDATE,DELETE,INSERT，InnoDB会自动给涉及数据集加排他锁；对于select,InnoDB不会加任何锁。

20.分库分表常见的问题？

全局主键问题

UUID，额外的自增表，分布式全局唯一ID生成算法

事务一致性问题

XA协议，2PC（两阶段提交）,3PC（三阶段提交）

关联查询问题（join）

字段冗余设计，数据组装，拆分查询

答：了解，但是没有这么干过。先垂直切分但是当单表数据量很大时只能水平切分，取模就行了。

21.什么是读写分离？/怎么实现？

答：将数据库的读写操作分散到不同的数据库节点上。一般情况下，都是选择一主多从，主节点负责写数据，从节点负责读数据。为了保证数据一致性，主从之间会进行数据同步。代理实现（mycat），组件实现(sharding-JDBC)

22.有什么问题？该怎么解决？

答：数据同步需要时间，可能会出现数据不一致的情况。一般两个方法：强制将读请求路由到主库二，延迟读：写请求后不立刻进行读请求，比如说我支付成功后，跳转到一个支付成功的页面，当我点击返回之后才返回自己的账互。

23.常见的分片算法？

答：哈希分片，范围分片，地理位置分片，融合分片（比如哈希加范围）

24.分库分表有什么问题？

1.无法join

2.事务问题

3.分布式ID

4.跨库聚合查询

Redis

1.解释一下缓存穿透，缓存击穿，缓存雪崩？

答：缓存击穿是指查询一个一定不存在的数据，由于数据不存在则不会缓存，那么每次请求都会向DB查询，这种请求多了会导致DB挂掉。解决方案有向缓存存储空值。或者是使用布隆过滤器。缓存击穿是指有个热点key过期了，这个时候有大量的请求直接访问DB,导致DB挂掉。解决方案有热点数永不过期或者加锁 setnx,或者设置逻辑过期。缓存雪崩是指大量的key同时过期，这时候大量的请求指向使得DB挂。击穿是一个key，雪崩是多个key,解决有给过期时间加个随机值。

2.如何保证双写一致性？

答：加上读写锁，用redission实现。读时是共享锁，读读不互斥，读写互斥。写的时候是排他锁。读写互斥，读读互斥。这样可以保证双写一致。或者采用延时双删，但是这个延时时间不好把握。所以还有要根据场景选择，强一致的情况采用读写锁。双写一致要求不那么高的就采用延迟双删。

3.redis持久化的两种方式？

答：RDB快照文件和AOF追加文件. RDB快但是会丢失数据，AOF慢但是丢数据的风险比较小。

4.过期策略？

答：定期删除和惰性删除。一般采用两种结合的方式。

5.数据淘汰策略？

答：默认是noevition,内存不足直接报错。还有LRU(最少最近)和LFU（最少使用频率），好像还有随机。

6. 数据库有1000万数据 ,Redis只能缓存20w数据, 如何保证Redis中的数据都是热点数据?

答：使用LRU淘汰策略，剩下来的就是热点数据了。

7.redis分布式锁怎么实现？

答：setnx(set if not exist).

1. 那你怎么控制有效时长呢？

答：这个指令确实不能这么做。可以采用redisson,它需要手动加锁，可以控制锁的失效时间和等待时间。当锁住的一个业务还没有完成，redisson引入了一个看门狗机制，每个一段时间看看是否还持有锁，有就增加持锁时间，业务结束后需要使用释放锁就可以了。

1. redis集群方式？

答：三种，主从复制。哨兵模式。分片集群

1. redis为啥这么快？

答：基于内存，单线程避免了多线程之间上下文切换的成本，使用了io多路复用

1. 说一下io多路复用？

答：利用单个线程来同时监听多个SOCKET，并在某刻socket就绪时得到通知，避免无效的等待。目前的IO多路复用采用的都是epoll模式，在通知用互进程socket准备就绪的同时，还会把已就绪的socket写入用户空间，不用挨个遍历socket来判断是否就绪。Select/poll需要挨个遍历文件描述符集合还要拷贝两次，内核态和用户态之间，epoll直接内核态。Select数组，poll是动态数组，链表好像。Epoll是红黑树和链表。

1. RabbitMQ有几种队列模式?

答：5种。分别是：

1.简单模式：一个生产者（发送方）对应一个消费者（接收方）。

2.Work模式：一个生产者对应多个消费者（但是只能有一个消费者能获得消息、排他）

3.发布/订阅模式：一个生产者将消息发送至fanout交换器，fanout交换器绑定到多队列，然后与之对应的所有消费者都能够接收消息（不排他）

4.路由模式：一个生产者将消息发送至direct交换器，交换器按照关键字（key）将消息路由至对应的队列。

5.主题模式：一个生产者将消息发送至topic交换器，交换器按照规则将消息路由至对应的队列。

我采用的是主题模式，将userId作为topic。

1. 怎么保证消息不会丢失？

答：主要从三个方面考虑，第一个就是开启生产者确认机制，确保生产者的消息能够到达队列，如果报错先记录到日志中，再去修复数据。第二个就是开启消息持久化，确保消息在未被消费前不会丢失。其中交换机，队列，消息都要做持久化。第三个就是开启消费者确认机制为auto,有spring确认消息处理成功完成ack,当然也要设置重试次数，一般为三次。如果超过三次还没有收到消息，就将失败的消息投递到异常交换机，交给人工处理。

（kafka）主要从三个方面考虑，第一个就是异步回调发送，如果消息发送失败，可以通过回调获失败后的消息信息。可以考虑重试或者记录日志。生产者可以设置消息重试，有时候是因为网络抖动造成的。第二个就是开启消息持久化，采用复制机制，发送者发送消息的时候，发送一个ack，其实就是确认机制，设置为all,就是说当生产者发送消息到了分区之后，不仅仅只在leader分区保存确认，在follwer分区也会保存确认，只有当所有的副本都保存确认以后才算是成功发送了消息，所以，这样设置就很大程度了保证了消息不会在broker丢失。第三个有可能是在消费者端丢失消息，kafka消费消息都是按照offset进行标记消费的，消费者默认是自动按期提交已经消费的偏移量，默认是每隔5s提交一次，如果出现重平衡的情况，可能会重复消费或丢失数据。我们一般都会禁用掉自动提价偏移量，改为手动提交，当消费成功以后再报告给broker 消费的位置，这样就可以避免消息丢失和重复消费了

1. 死信交换机或者延迟队列有听说过吗？

答：首先我说说死信吧，当消息被拒绝或者消息过期了，队列满了就会成为死信。这时死信就会被重新发送另一个交换机，这个交换机就是死信交换机。跟这个死信交换机相连的队列就叫做死信队列。延迟队列的话就是说我有一个消息我不想马上给消费者消费，我想过一段时间给再给消费者消费。可以使用死信队列和ttl（消息存活时间）实现。我记得rabbitmq有个插件可以直接使用，我们直接指定一个交换器是死信交换器就行了，然后发送消息的时候直接指定超时时间就行。

1. 怎么处理消息堆积？

答：第一提高消费者的消费能力，比如说可以使用多线程消费任务。

第二增加更多的消费者。还有扩大队列的容量，比如说可以使用惰性队列。

1. rabbitmq的高可用机制？

答：采用镜像集群机制，一主多从，所有操作都是主节点完成，从节点向主节点同步即可。如果主节点宕机了，镜像节点就会替代成新的主节点，如果在主从同步完成前发生宕机就会丢失数据。用仲裁队列即可，采用raft协议，强一致。

1. 软件设计原则有哪些？

答：开闭原则、单一职责原则、里氏替换原则、依赖倒置原则、接口隔离原则，合成/聚合复用原则，迪米特法则。

开闭原则：对拓展开放，对修改关闭。

单一职责原则：一个类只负责一个功能领域的相应职责。

里氏替换原则：所有引用基类的地方必须能透明的使用其子类的对象。

依赖倒置原则：依赖于抽象，不能依赖于具体实现。

接口隔离原则：类之间的依赖关系应该建立在最小的接口上

合成/聚合原则：尽量使用合成/聚合，而不是通过继承达到复用的目的。

迪米特原则：一个软件实体应当尽可能少的与其他实体发生相互作用。

1. 你觉得为什么要有设计模式/

答：我觉得设计模式是软件开发人员在软件开发过程中总结出来的一般性问题的解决方案。合理的使用设计模式可以重用代码，让代码更容易被他人理解，实现代码低耦合。针对很多一般性问题我们都可以利用设计模式中对应的模式来解决。

1. 你知道的那些设计模式？项目中遇到了哪些？

答：工厂模式：工厂模式包括简单工厂模式，工厂模式、抽象工厂模式。简单工厂模式就是指由一个工厂对象来创建实例，客户端不需要关注创建逻辑，只需要提供传入工厂的参数。由于所有的产品都由一个工厂来创建，这就会导致如果要增加新产品，就需要修改工厂类的判断逻辑，这违背了开闭原则，而且产品太多工厂类就会比较复杂。

工厂模式跟简答工厂模式相比，他讲每个产品都交给具体的工厂来生产。就是定义一个抽象工厂，其定义了产品的生产接口，但是不负责具体的产品，讲产品任务交给不同的派生类工厂。

抽象工厂模式是指工厂模式只能针对一类产品，如果要生成另一种产品甚至多种产品就比较难办了。他在抽象工厂中增加创建产品的接口，并在具体子工厂中实现新家产品的创建。

单例模式是指一个单例类在任何情况下都只存在一个实例，他的构造方法必须是私有的，由自己创建的一个静态变量存储实例，对外提供一个静态公有方法获取实例。优点就是内存中只有一个实例，减少开销。由饿汉式单例（线程安全）和懒汉式单例（线程不安全）。懒汉模式是在第一次调用实例方法的时候才创建实例，就是延迟加载所有会有线程安全问题。饿汉模式在类加载时就立刻创建实例，在任何线程访问之前都已经存在了，不存在线程安全问题。

适配器模式：我们的应用程序中可能会遇到要将两个不同接口的类进行通信，在不修改这两个的前提下我们就需要某刻中间件来完成这个衔接的过程。这个中间件就是适配器。所谓的适配器模式就是将一个类的接口，转换成客户期望的另一个接口，他可以让原本两个不兼容的接口无缝对接。

代理模式：它允许一个对象（代理）充当另外一个对象的接口，已控制对该对象的访问。代理经常用于增强或者修改别被代理对象的行为，而不需要修改代码。代理模式可以用于各种情况，比如延迟加载、访问控制、日志记录等等。

观察者模式：用于定义对象之间的一对多的依赖关系，当一个对象的状态发生改变时，其所有依赖对象都会收到通知并进行更新。观察者模式也叫做发布订阅模式。

责任链模式：为了避免请求发送者与多个请求处理者耦合在一起，将所有请求的处理者通过前一个对象记住下一个对象的引用而形成一条链；当有请求发生时，可将请求沿着这条链传递，直到有对象处理他为止。

策略模式：该模式定义了一系列算法，并将每个算法封装起来，使他们可以相互替代，且算法的变化不会影响使用算法的客户。他通过对算法进行封装，把使用算法的责任和算法的实现分割开来，并委派给不同的对象对这些算法进行管理。只要代码中有冗长的if-else和switch分支判断都可以采用策略模式进行优化。

装饰器模式：主要对现有的类对象进行包装，以期望在不改变类对象及类定义的情况下，为对象添加额外的功能。（就是说在不改变原有类的基础上，给类对象增加新的功能）。可以通过一个或者多个装饰器来实现这么目的，每个装饰器都实现了相同的接口，以便与原始对象保持一致。

21．Spring中使用了哪些设计模式/

工厂模式：beanfactory、applicationcontext创建bean对象

单例模式：创建bean默认是单例的

适配器模式：springMVC就用到了适配器模式来适配controller.

22.Mybatis的执行流程？

答：1.读取配置文件，加载运行环境和映射文件

2.构造会化工厂sqlsessionfactory

3.会话工厂创建会话对象

4.操作数据库接口，executor执行器，同时负责查询缓存的维护。

5.executor接口的执行方法中有一个mappedstatement类型的参数，封装了映射信息。

6.输入参数映射

7.输出参数映射。

23. Mybatis支持延迟加载吗/

答：支持，但是默认关闭

24. Mybatis一级缓存、二级缓存？

答：作用域为sqlsession，默认打开一级缓存。二级缓存需要单独开启作用域为namespace和mapper

25.内存管理主要负责什么？

答：内存分配、内存释放、内存优化、地址转换（主要是虚拟地址转换为物理地址）、内村扩充、内存映射（将一个文件直接映射到用户进程的进程空间中，这样就可以直接通过内存指针来读写内存直接存取文件，速度更快）

26．什么是内存碎片？

答：内部内存碎片，外部内存碎片。已经分配给进程使用但是未被使用的内存。内部碎片的原因：采用固定比例比如二的幂次方进行内存分配时，进程所分配的可能会比实际的要大。外部内存碎片的原因：由于未分配的连续内存区域太小，以至于不能满足任何进程所需要的内存分配请求。就是未分配给进程但是又不能够使用的进程。

27.常见的内存管理方式？

答：连续内存管理（块式管理）和非连续内存管理（段式管理、页式管理、段页式管理机制）

28.说说虚拟内存？

答：虚拟内存本质上只是逻辑存在的，是一个假想出来的内存空间，主要作用就是作为进程访问物理内存的桥梁。虚拟内存可以隔离进程，物理内存通过虚拟地址空间访问，虚拟地址空间与进程一一对应。那个进程都拥有了完整的物理地址，进程之间彼此隔离。多个进程共享物理内存（进程运行过程中，回加载许多操作系统的动态库。这些库对每一个进程而言都是公共的，他在实际内存中只会加载一份）。提供系统的使用安全。

没有虚拟内存的话用户程序就可以访问任意物理内存，可能会误操作到系统运行必须的内存，进而造成系统崩溃，严重影响系统的安全。同时运行多个程序崩溃，比如一个用户内存封盖另一个的。

MMU内存管理单元：将虚拟地址转化为物理地址。

28．文件链接是一种特殊的文件类型。可以在文件系统中指向另一个文件。

29.inode就是用来维护某个文件被分成几块，每一块在的地址，文件拥有者，创建时间，权限，大小等信息。Inode文件的属性信息。Block实际的文件内容。

30.硬链接和软链接有什么区别？

答：硬链接通过inode节点号建立链接，硬链接和源文件的inode节点号相同，两者对文件系统来说是完全平等的，删除一个对另一个完全没有影响，可以通过给文件设置硬链接文件来防止重要的文件被误删。只有删除了源文件和硬链接，该文件才会被真正删除。

软链接和源文件的inode节点号不同，而是指向一个文件路径。源文件删除后，软连接依然存在，但是指向的是一个无效的文件路径。类似于windows系统中的快捷方式。

Java的优点和特点？

答：一次编译到处运行。

31.说说ELK实现日志监控系统？

答：ELK是elastic search、logstash、kibana三个组件的首字母缩写。 其中，logstash的主要工作是日志的收集，通过input、filter、output三部分，把日志收集，过滤，输出到elasticsearch中；elasticsearch作为开源的分布式引擎，提供了搜集、分析、存储数据的功能。Kibana则是作为elasticsearch分析数据的可视化展示，可以对日志进行分析，汇总，监控和搜索日志用。项目结合的话就是通过springboot项目中引入logbash这个jar包。在logback-spring.xml文件进行配置。比如说logstash的ip和端口。

6666