问题：

1. 三次握手中，最后一次回复丢失，会发生什么？

答：客户端第三次丢失是感应不到的，所以，会当作正常情况发送报文，Server 端将以 RST 包响应要求，此时，客户端知道，第三次握手失败。而服务器经过一段时间后，没有得到第三次握手的响应，则重新发送第二次握手的包。默认为 5 次重发。间隔时间为 3，6，12 秒，之后就放弃。

2. 消息队列的确保机制：

答：

- 发送端的可靠性

发送端完成操作后一定能将消息成功发送到消息队列中。

实现方法：在本地数据库建一张消息表，将消息数据与业务数据保存在同一数据库实例里，这样就可以利用本地数据库的事务机制。事务提交成功后，将消息表中的消息转移到消息队列中，若转移消息成功则删除消息表中的数据，否则继续重传。

- 接收端的可靠性

接收端能够从消息队列成功消费一次消息。

两种实现方法：

保证接收端处理消息的业务逻辑具有幂等性：只要具有幂等性，那么消费多少次消息，最后处理的结果都是一样的。

保证消息具有唯一编号，并使用一张日志表来记录已经消费的消息编号。

3. redis 高并发的原因：

答：

- 它是单线程，没有进程竞争，锁等设置，所以少了切换上下文的时间，相对快了很多。
- 同时，数据存储在内存中，他可以快速处理数据。
- 同时，它又是 epoll 的多路复用模式，异步的读取信息，自己要进行的逻辑处理也相对很少。并且可以涉及单机多 redis，充分利用其他 cpu 核心。

4. threadlocal 的作用：

答：确保线程中有独享的变量副本，可以为自己服务。这类副本本身是不用多个线程维护状态的。所以可以放入线程本地。

5. 微服务注册中心，如何主动删除服务？

参考： https://blog.csdn.net/xiaobao5214/article/details/81263445?utm_source=blogxgwz1

答：

- 一个是客户端自己关闭后，eureka 检查心跳，发现服务已经下线就关闭。
- 一个是发送 delete 请求：/eureka/apps/app-name 到注册中心。
- 还有就是客户端如果是 springboot，那么调用方法 DiscoveryManager.getInstance().shutdownComponent();即可。

6. http 状态码，及其对应的 304 状态码的含义：

答：客户端在请求一个文件的时候，发现自己缓存的文件有 Last Modified （截止时间），那么在请求中会包含 If Modified Since ，这个时间就是缓存文件的 Last Modified 。因此，如果请求中包含 If Modified Since，就说明已经有缓存在客户端。服务端只要判断这个时间和当前请求的文件的修改时间就可以确定是返回 304 还是 200 。返回 304，则代表浏览器可以继续使用缓存数据。如此，可以减少传输的数据量，同时，服务器也不需要进一步执行相关数据的查询。

7. 两种数据库引擎的差异

答：6 大差别：
- Innodb 支持行级锁，和外键，myisam 不支持行级锁和外键。
- innodb 删除表的数据是一行一行的删除的，而 myisam 是直接删除一个表。
- innodb 表中，没有缓存这个表的数量（即 count（*））；而 myisam 缓存了。
- innodb 支持事务，而 myisam 不支持。
- innodb 的存储文件是一个大文件，而 myisam 是三个文件，分别存储数据，索引，表的定义。在迁移数据库的时候更加方便。
- 5.6 之前，InnoDB 不支持全文索引，MyISAM 支持全文类型索引

8. b+树的优点：
答：
- 数据均存储在叶子节点中，非叶子节点只有索引值。这样的设置，非叶子节点小，一次性读入内存的节点多，io 新能消耗低。
- B+树的查询效率更加稳定，路径长度一致。
- B+树的数据都存储在叶子结点中，各个叶子节点有兄弟指针，所以，扫库的时候很方便，只需要扫一遍叶子结点即可。

9.聚簇索引的特点：
https://blog.csdn.net/qq_29373285/article/details/85254407
https://www.cnblogs.com/wangkaihua/p/10220462.html
答：索引的位置和其对应的物理内存顺序是一致的，即，索引排在前的，其数据存储的物理地址也在前面。同时，聚簇索引的叶子节点就是数据，而非聚簇的叶子节点是主键 id，还要根据这个 id 去对应的聚簇索引中查询相关的数据。
9. 复合索引的最左原则
答：最左匹配四个点：
- 建立一个复合索引（a，b，c），相当于建立了多个索引(a),(a,b),(a,b,c);
- 索引排序是根据从左向右匹配的。
- 查询的时候，优化器会优化 sql 语句为对应索引的顺序，即：（a,c,b）也是可以用索引的。
- 单独查询 b，会采用 index 类型索引，即：不符合最左，直接查询所有的节点，找到对应的数据指针，回表，这样很慢。

补充：复合索引的三个好处：可以减小索引树的个数，节省空间开销，建立一个树，相当于建立三棵树。可以提高效率，索引越多，筛选出的剩余数据越少，回表的时候越快。可以实现覆盖索引。即：查询的数据刚好是我们的树的索引项，比如就要找（a,b,c），那么就可以不用回表，返回索引的数据即可。
10. springmvc 的过程： https://www.cnblogs.com/fengquan-blog/p/11161084.html
答：
- 请求给 dispetchServlet 拦截，dispetchServlet 注册 handlerMapping。
- 转发（调用自己的成员变量）给 handlerMapping 对象，它根据配置文件，初始化自己的值。其中有个自己维护的一张 map《url，bean》，根据拦截到的 url 的路径，可以找到对应的 controller 对象，封装为 HandlerExecutionChain（里面可能还有一些拦截器，面向切面那种）。返回给 dispetchServlet。
- dispetchServlet 转发 HandlerExecutionChain 给适配的 handlerAdapter（也是 dispetchServlet 自己注册维护的，默认有三种适配器）。
- handlerAdapter 根据接收到的 controller（也是 handler）的类名，利用反射，执行其 handle 方法。（如果有拦截器，会先执行 pre 拦截器，而后是 handler 的方法，之后是 post 拦截器。

这些拦截器好像都是动态代理前织入的）返回 modelandview 给 dispetchServilet。
- dispetchServilet 转发 modelandview 给 viewResolver；
- viewResolver 将 modelAndView 找出，这里其实也是一个 map 映射过程。modelAndView 只是一个包含 model 数据和页面名字字符串的对象。viewResolver 根据这个字符串去到配置的路径，如 resource 之类的地方，找到对应的 jsp 页面。然后返回具体的 view 对象给 dispetchServilet。
- dispetchServilet 调用对象 View 的方法 renter（）。该方法将 model 注入到 jsp 中（其实是装入了 response 的请求头里），然后返回。
- dispetchServilet 将视图返回给浏览器。

11. 分布式锁的实现原理：
答：三个：
- 利用 mysql
- 利用 redis 的 setnx 实现
- 利用 redis 集群算法 RedLock

12. 为什么用线程池不用 new？
答：1、每次 new Thread，新建对象性能差 2、缺乏统一管理，可能导致线程创建过多，死机等。3、缺乏更多功能，如：定时执行，定期执行，线程中断等。
13. 为什么线程池不允许使用 Executors 去创建？
答：推荐使用 threadPoolExecutor，因为 executors 其实就是对 threadPoolExecutor 的一个封装。而且封装的不好，它允许创建的最大线程是 maxint，很容易导致 oom，内存不足错误。所以，创建线程池，还是要我们自己手动通过 threadPoolExercutor 设置参数。
- newFixedThreadPool 和 newSingleThreadExecutor:是等待队列无限长
- newCachedThreadPool 和 newScheduledThreadPool:是最大线程数无限长
14. get 和 post 的区别，get 的最长数据段是？
答：
- get 主要是为了获取数据，而 post 是为了提交数据。两者虽然都有给服务器传输数据的功能，但是意义是不一样的。
- get 拼接在 url 后，post 的数据放入 body 的 param 内
- GET 的目的是是读取，所以，服务器对应的接口应该有幂等性。即：多次请求的数据，不会因为我的 get 改变。同时，因为幂等，所以就可以对 GET 请求的数据做缓存。
- get 在 url 上传递参数，默认是 ascii，不支持中文，要经过其他配置，而 post 内的编码可以是 unicode-8，支持中文。
- URL 的最大长度是 2083 个字符，path 的部分最长是 2048 个字符。不过其实是 ie8 规定的，http 协议没有这一点。
15. redis 的 5 大数据结构和其底层实现，还有应用场景
答：
16. url 的解析过程。
答：四个步骤；
- 如果本机刚刚联网，则会去通过 DHCP 协议，发送广播，目标地址是 67，源地址端口是 68 。获得即插即用的本机的 ip 地址和距离本机最近的 DNS 服务器的 ip，以及网关路由的 ip。
- 而后是解析浏览器的 url。由于我们只知道网关路由器的 ip，所以要通过 ARP 协议解析网关 ip，获得 mac 地址。

- 得到后，继续进行 url 的解析。通过发送 url 给网关路由，再由路由转发到对应的 DNS 服务器（ DNS 服务器的端口号是 53），DNS 服务器再通过域名解析，找到对应 url 的 ip 地址。
- 而后，浏览器可以根据 DNS 返回的 ip，进行 tcp/ip 协议，三次握手过程建立连接。（80 端口）
- 最后是双方通信，服务器返回 web 页面信息，浏览器渲染并且显示。

17. 浏览器会对 html 做什么？

答：渲染。

18. java 的启动过程。（类的加载） https://www.cnblogs.com/qiumingcheng/p/5398610.html

答; 1. 编译，2.类的加载（5 个步骤），3.执行程序输出结果。

19. mybatis 的启动过程： https://www.cnblogs.com/ZhuChangwu/p/11741125.html

答：

- 通过 sqlsessionFactory 类注册 xml 配置文件（数据库的 url 之类的）；
- sqlsessionFactory 调用方法，获得 sqlsession 类。这个类封装了和数据库的连接，还有一系列的事务管理功能；
- 而后，session 内，有方法可以通过 scan 扫描到配置文件，生成 mapperProxy 对象；该对象将读取对象的 xxxMapper.xml 文件和对应的 xxx 接口。采用动态代理方法，生成一个接口实现类。
- 执行 sql 语句时，通过反射的方式，调用实现类的方法，其中有两个参数，对应的方法名和 sql 语句。这由 mapperMethed 对象进行封装。
- mapperMethed 传入 sqlsession 内封装的 CRUD 方法，内有对应的执行器类 Executor；
- 执行器类将调用方法 query（），取出 mapperMethed 内的 sql 语句，生成 prepareStatement。
- 最后，像执行 jdbc 一样，进行数据库查询。

总结：mybatis，其实就是封装了 jdbc。只不过，采用了动态代理，将接口和方法实现拆分开来。

20. springboot 的启动过程。

答：

- new 了一个 SpringApplication 对象，使用 SPI 技术加载加载 ApplicationContextInitializer、ApplicationListener 接口实例
- 调用 SpringApplication.run() 方法
- 调用 createApplicationContext()方法创建上下文对象，创建上下文对象同时会注册 spring 的核心组件类（ConfigurationClassPostProcessor 、AutowiredAnnotationBeanPostProcessor 等）。
- 调用 refreshContext() 方法启动 Spring 容器和内置的 Servlet 容器

21. springboot 如何启动 mybatis： https://www.cnblogs.com/nxzblogs/p/10484281.html

答：

- 和上面启动 spring 一样，在 refreshContext（）方法调用后，会去将加载进来的各种配置进行处理，装入我们的 springboot 容器中，包括了 spring，和 mybatis。mybaitis 的具体过程，其实就是 19 的过程，只不过最后把 sqlsessionFactory 放入了容器中。

21. jva 的 gc 标记算法，以及为什么不用标记计数算法？

答; gc 标记算法有两个：一个是标记计数算法，有一个对象引用，就标记一次。但是，在两个对象出现循环引用的情况下，此时引用计数器永远不为 0，导致无法对它们进行回收。

第二个是可达性分析算法：即: 判断栈和方法区中是否有该对象的引用，没有，则回收。（标

记计数法中，存在的问题是，堆对象和堆对象之间互相计数+1 了，但是，栈中使用引用以及被释放，比如方法已经结束了，但是，没有进行-1 操作，所以存在无法回收）

22. 转发和重定向的区别：
答：转发就是服务器中，一个 servlet 对这个请求进行了处理，但是，没有返回，而是给了服务器另一个 sevrlet 再加工，而后返回。这个过程，请求头和响应头用的都是同一个。里面的协议，编码，数据都不改变。即：两个 servlet 间是实现了通信。
重定向就是客户端发送信息，到达了服务器，而后服务器接处理了请求后，返回消息给浏览器，并且告诉他，还要进行后续的操作，返回状态码为 301，浏览器立即进一步发送请求。这个过程是两次请求，而非 servlet 之间的通信。
表现是：浏览器上的 url 的改变，转发并不会改变。

23. 数据库中的乐观锁和悲观锁
答：我只是说了一下定义，面试官表示，不是要原理，而是实现。
定义就是：悲观锁是认为冲突会发生，所以加了锁，乐观锁是认为，冲突一般不会发生，会在完成业务逻辑后，在提交之前进行验证。
悲观锁我们数据库可以依靠自带的 XS 锁实现。
乐观锁，我们采用 CAS 和 vesion（或者时间戳）来实现。这就需要代码上进行一定的处理了。比如，我们可以多加一个字段 vesion，读取数据的时候，将 vesion 读取出来，保存前，进行比较 version 值，没有改变，则 version+1 后，再存入我们真正操作的数据。如果 version 改变了，那就代表，这个数据已经过期，则要重新来一遍了。