# 场景题

**订单超时未支付如何关单，恢复库存**

**定时任务**

@Scheduled 定时扫描订单表，检查是否超时；容易实现；存在间隔导致关单不及时；表中数据越多查询时间越长，导致关单不及时

**JDK 延迟队列 DelayQueue**

DelayQueue :JDK 提供的无界队列，添加任务的延迟时间过后，才会真正放入队列中。while (true) 循环查询队列，存在任务则进行处理关单。容易实现；订单过多会导致OOM; 数据存在内存，服务器重启/宕机会丢失数据，适合数据量小的非主要场景

**Redisson 分布式延迟队列**

用 zset 实现，当前时间+延迟时间排序，时间到后放到任务list中执行，适合分布式场景

**RocketMQ 延迟消息**

延迟消息 ：消息写到Broker 后，等待指定时长后才被消费。系统解耦；mq 是重量级组件，存在消息丢失、幂等性问题

**RabbitMQ 死信队列（延迟消息）**

消息TTL 过期、超过队列长度，被消费者拒绝，无法被消费时，成为死信消息，进入死信队列。给消息设置 ttl，不消费，进入死信队列后再消费；队头过期时间长，会导致后面过期消息无法及时消费 ，造成消息阻塞

**分布式锁过期时间的作用**

避免节点宕机导致锁无法释放；存在任务未执行完，但是锁被释放的问题

Redisson看门狗：10秒检查一次， 线程还在运行则延长过期时间

心跳检测：持有锁的节点定期向系统发送心跳，规定时间内未收到则释放锁

**分布式锁的实现**

**setnx +expire:**抢锁后设置过期时间；非原子操作，抢锁后，系统宕机导致锁无法释放

**Lua脚本:** 执行SETNX + EXPIRE，保证原子性

**SET EX PX NX:** 原子操作

**redisson:**看门狗机制解决了任务未执行完但是释放锁的问题

**Redlock+Redisson**：在redis集群中，主节点持有锁宕机后，从节点升为主节点，第二个线程又能获得锁。使用redisson中的RedissonRedLock解决

**Etcd分布式锁**：键值存储，支持原子操作和分布式事务。强一致性,数据写入后，所有副本立即更新，确保数据的一致性和可用性

**数据库分布式锁**：加锁时插入数据(key,ttl)，解锁时删除数据,定时处理过期的锁

zookeeper分布式锁：唯一节点特性，多个请求同时创建节点，只有一个能创建成功(加锁)

**redis分布式锁解决集群服务器定时任务重复执行问题**

定时任务开始时，setnx加锁，当key不存在，放入缓存，返回true,并设置超时时间，结束后解锁；第二个任务setnx加锁，发现已存在key，返回false,不执行定时任务

**数据库事务最小化**，防止长事务，行级锁长时间无法释放，导致其他事务等待。避免非内部逻辑导致不必要的事务回滚

**长事务存在的问题**：

降低数据库并发性能：长事务会锁主资源，其他事务访问

耗尽数据库连接池：长事务会导致客户端占用数据库回话时间过长，占用连接数量，导致数据库拒绝服务

耗尽线程池：长事务会中线程一直等待事务提交，严重情况下会导致线程池耗尽

**导致长事务的情况(需要设置超时处理)**：

事务中调用其他微服务，但一直没响应

事务中调用基础设施：Redis,MQ,etcd,ES等,由于故障没有响应

事务中进行大量循环

**数据导出成excel**

使用easyExcel将数据导出成excel，调接口保存到文件服务器上，并返回文件Id给前端，前端调文件获取接口进行下载(可以下载历史导出的文件)； 文件服务器做了处理，如果同名不会覆盖，两个文件的文件Id不同

**百万数据如何导出成excel**

用easyExcel

分批进行分页查询，每次查询1000条，拼起来

使用线程池，多线程查询

分多个sheet页，一个sheet只能存100多万数据, 每个sheet50W数据，数据越多打开越慢

**百万数据如何通过excel导入**

分批读取数据

用批量插入语句(insert into values)，不要一条一条插入

**@Schedule和xxl-job执行定时任务的区别**

@Schedule:不需要依赖框架，无法管理任务(暂停/手动触发，任务是否执行成功)，集群中如果不加分布式锁，每个节点的定时任务会重复执行

Xxl-job:分布式任务调度

**拦截器AOP面相切面编程**

如在所有请求的header中增加参数

**订单超时未支付如何解决**

1.在生成订单时(表中记录订单的创建时间)，加入队列

2.根据订单的创建时间到30分钟为准,如果订单没有支付

3.将订单生成消息放到死信队列

4.消费死信队列里面的消息: 删除数据库中未付款的订单, 需要去修改商品的库存

**代码重构手法**

将重复代码提取成方法，进行复用

将公共方法提取到父类中

**在事务中又查询又调接口会有什么问题**

如果事务回滚，调用的外部接口无法回滚

要做到事务最小化

**单据序列号生成器(固定前缀+年月日+序列号)**

1. 将前缀，年月日，序列号存到数据库中，定时任务更新年月日和序列号每日刷新，每个请求读取序列号时上分布式锁
2. 使用redis原子操作存储和更新序列号，定时任务更新序列号， 计数器自增都是原子操作

**如何设计字段的数据权限**

设计角色-字段表，通过拦截器，查询时，修改sql, 只查询有权限查看的字段

**Cpu狂飙，如何处理**

使用top命令，查看哪些进程占用cpu比较多

对服务器进行临时扩容，或者进行版本回退

查看数据库使用情况，是否是慢sql导致，适当增加索引，优化sql逻辑

查看程序日志和代码，检查原因，后续使用多线程异步执行复杂任务

**高并发如何处理**

线程池-多线程处理-阻塞队列

使用消息队列-排队处理

使用redis缓存-避免频繁读取数据库

数据库sql优化，读写分离

使用锁-保证线程安全

减少锁的颗粒度

将高并发服务和一般服务拆分，避免整个系统崩溃

增加服务器节点

**如何解决死锁**

给锁增加过期时间

避免嵌套锁

加/释放锁都记录日志，方便排查