

秒杀系统与订票系统设计

Flash Sale & Booking System Design

主讲人：南帝老师



系统架构设计2021答疑群、
课程信息、优惠福利等等
欢迎来撩班主任小佳娃

为什么要学习 系统设计？

1. 为系统设计面试做准备

在很多科技公司中，除了代码面试，系统设计也是技术面试过程中的一个必要环节。

2. 后端架构师必备技能

要想成为年入百万的后端架构师，系统设计是必不可少的技能。

从课程中我们能 学到什么？

1. 如何在面试中回答

在本课程中，介绍了4S分析法，并对大量常见系统进行了设计与讲解。

2. 如何设计与搭建系统

学习完本课程后，对于大部分系统我们都能有一个相对完整的方案，让我们在从零搭建一个陌生的项目时，不再毫无头绪。

随时可看	<ul style="list-style-type: none">• 【互动】走进系统设计 & 新鲜事系统设计• 【直播】秒杀系统与订票系统设计
Week1	<ul style="list-style-type: none">• 【互动】从用户系统中学习数据库与缓存• 【互动】网站系统、API设计与短网址• 【直播】优惠券系统设计
Week2	<ul style="list-style-type: none">• 【互动】数据库拓展与一致性哈希算法• 【互动】分布式文件系统 GFS• 【直播】文档协同编辑系统设计
Week3	<ul style="list-style-type: none">• 【互动】分布式数据库 Big Table• 【互动】聊天系统 IM System• 【直播】视频流系统设计
Week4	<ul style="list-style-type: none">• 【互动】基于地理位置的信息系统• 【互动】分布式计算 MapReduce• 【直播】推特搜索系统设计 Twitter Search
Week5	<ul style="list-style-type: none">• 【互动】爬虫系统与搜索建议系统• 【互动】系统的核心必考知识点：数据库索引与事务





Scenario 场景

- 说人话：需要设计哪些功能，设计得多牛
- Ask / Features / QPS / DAU / Interfaces

Service 服务

- 说人话：将大系统拆分为小服务
- Split / Application / Module

Storage 存储

- 说人话：数据如何存储与访问
- Schema / Data / SQL / NoSQL / File System

Scale 升级

- 说人话：解决缺陷，处理可能遇到的问题
- Sharding / Optimize / Special Case



Work Solution
Not Perfect Solution

Scenario 场景

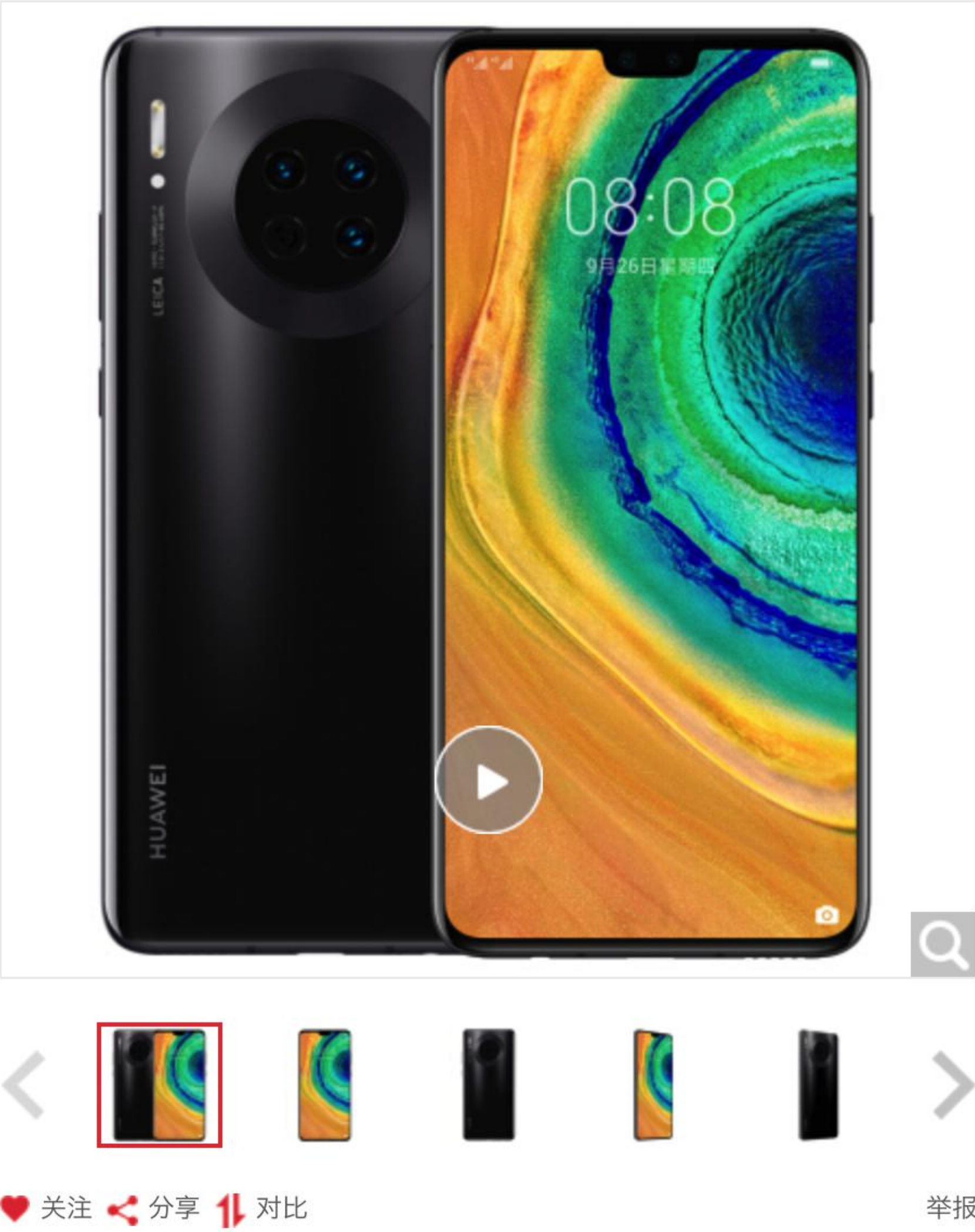
- 具体场景有哪些?
- 实际需求有什么?
- 详细流程怎么样?

2020 年6 月18 日 0 点开始，京东自营限量 100 台，以 4000 元的价格，抢购 iPhone 11 64G 版本，先到先得，一人限购一台，售完即止。

微信抢红包

抢春运火车票

抢购小米手机



华为 HUAWEI Mate 30 麒麟990旗舰芯片4000万超感光徕卡影像屏内指纹8G+128GB亮黑色4G全网通版
游戏手机

【直降400，到手价3599】买5G手机，4000万超感光徕卡影像屏内指纹；[爆款至高立省800](#)》

秒杀预告 7月4日00:00 秒杀价 ¥ 3369

距离开始还剩 07 : 56 : 26

京东价 ¥ 3599.00 降价通知

累计评价
27万+

增值业务 高价回收，极速到账 套 3元1G

配送至 浙江杭州市滨江区西兴街道 有货 支持 京尊达 京准达 211限时达 免举证退换货

由 京东 发货，并提供售后服务。23:54前下单，预计明天(07月04日)送达

重量 0.54kg

选择颜色 星河银 亮黑色 翡冷翠 罗兰紫

选择版本 8GB+128GB 6GB+128GB

购买方式 官方标配 联通不换号 最高直降2560元 选联通靓号 最高直降2560元

套 装 优惠套装1 优惠套装2 优惠套装3 优惠套装4 优惠套装5 优惠套装6

增值保障 2年全保修 ¥199 原厂碎屏保修 ¥299 2年电池换新 ¥39

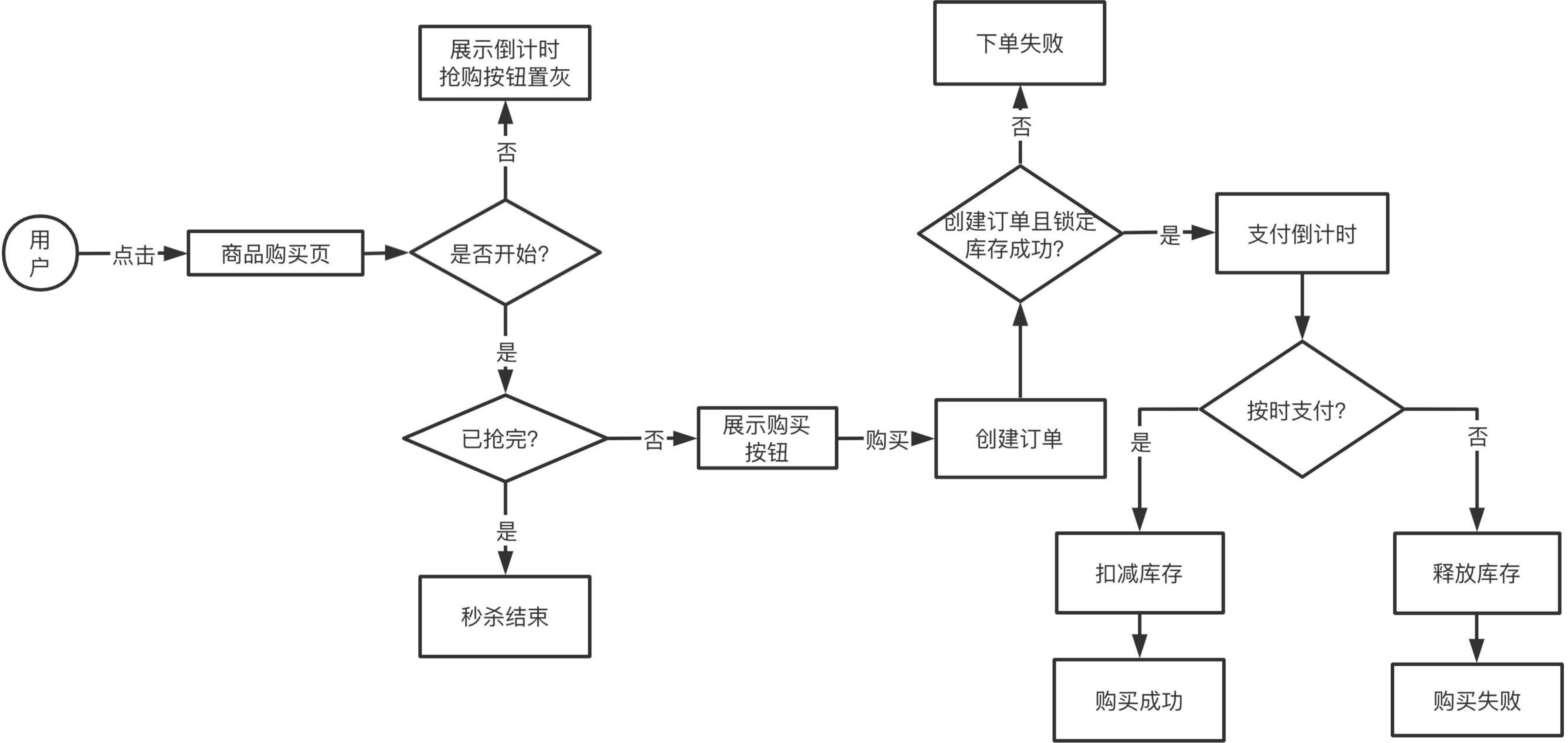
京东服务 DIY手机壳 ¥38 数据恢复 ¥39

秒杀系统场景 - QPS 分析

平日每秒 1000 人访问该页面。

秒杀时每秒数10万人访问该页面。

QPS 增加 100 倍以上。



Large flow and high concurrency

瞬时大流量高并发

服务器、数据库等能承载的 QPS 有限，如数据库一般是单机 1000 QPS。需要根据业务预估并发量。

Over Sale

有限库存，不能超卖

库存是有限的，需要精准地保证，就是卖掉了 N 个商品。不能超卖，当然也不能少卖了。

Malicious ticket Grab

黄牛恶意请求

使用脚本模拟用户购买，模拟出十几万个请求去抢购。

固定时间开启

时间到了才能购买，提前一秒都不可以（以商家「京东」「淘宝」的时间为准）。

Purchase limit

严格限购

一个用户，只能购买 1 个或 N 个。

商家侧（京东自营、淘宝天猫店家）

新建秒杀活动

配置秒杀活动

用户侧

商品秒杀页面（前端或客户端）

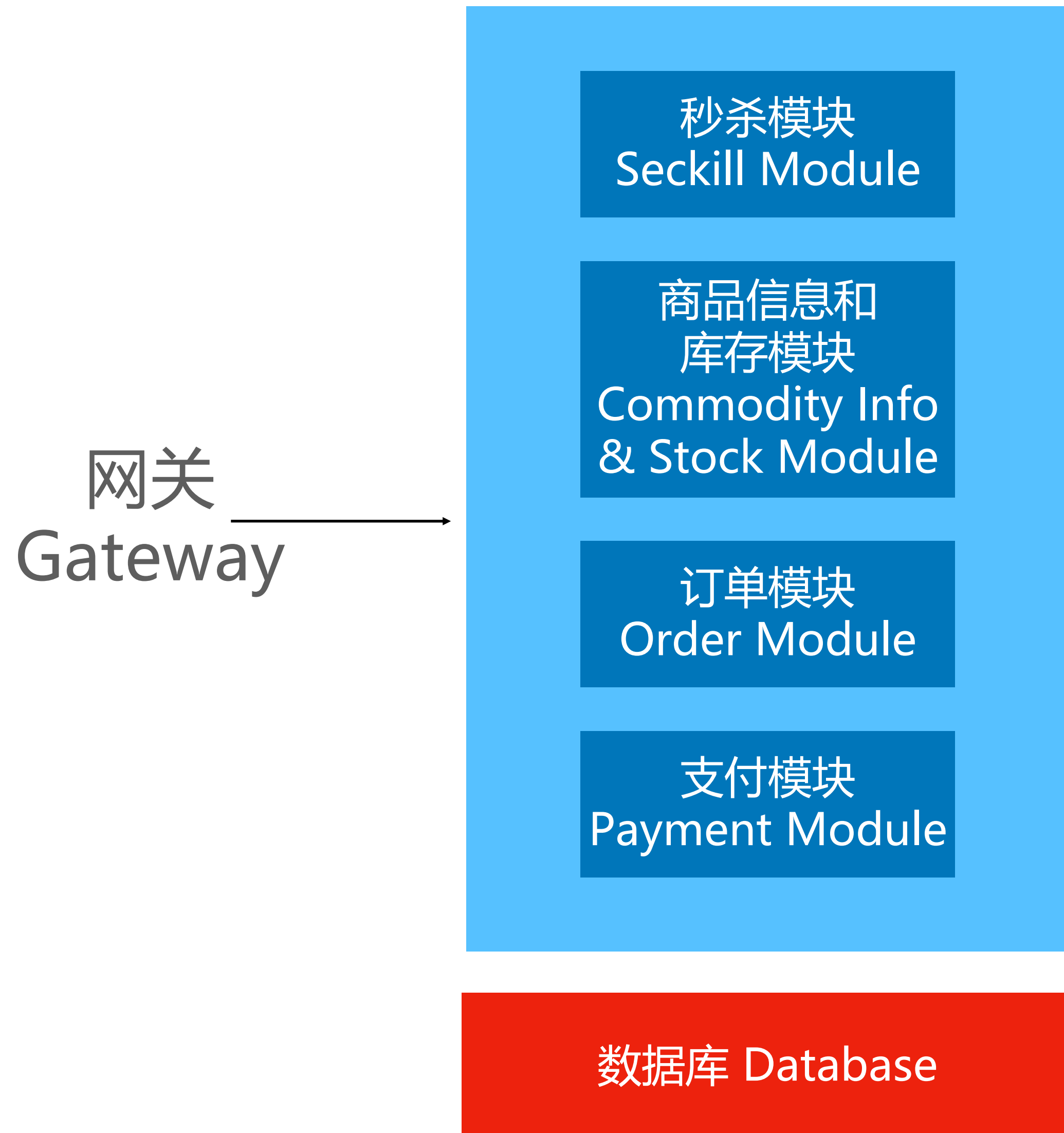
购买

下单

付款

Service 服务

单体架构 or 微服务?



Coupling

前后端**耦合**，服务压力较大。

各功能模块耦合严重。

系统复杂，一个模块的升级需要导致整个服务都升级。

Expandability

扩展性差，难以针对某个模块单独扩展。

Cooperation

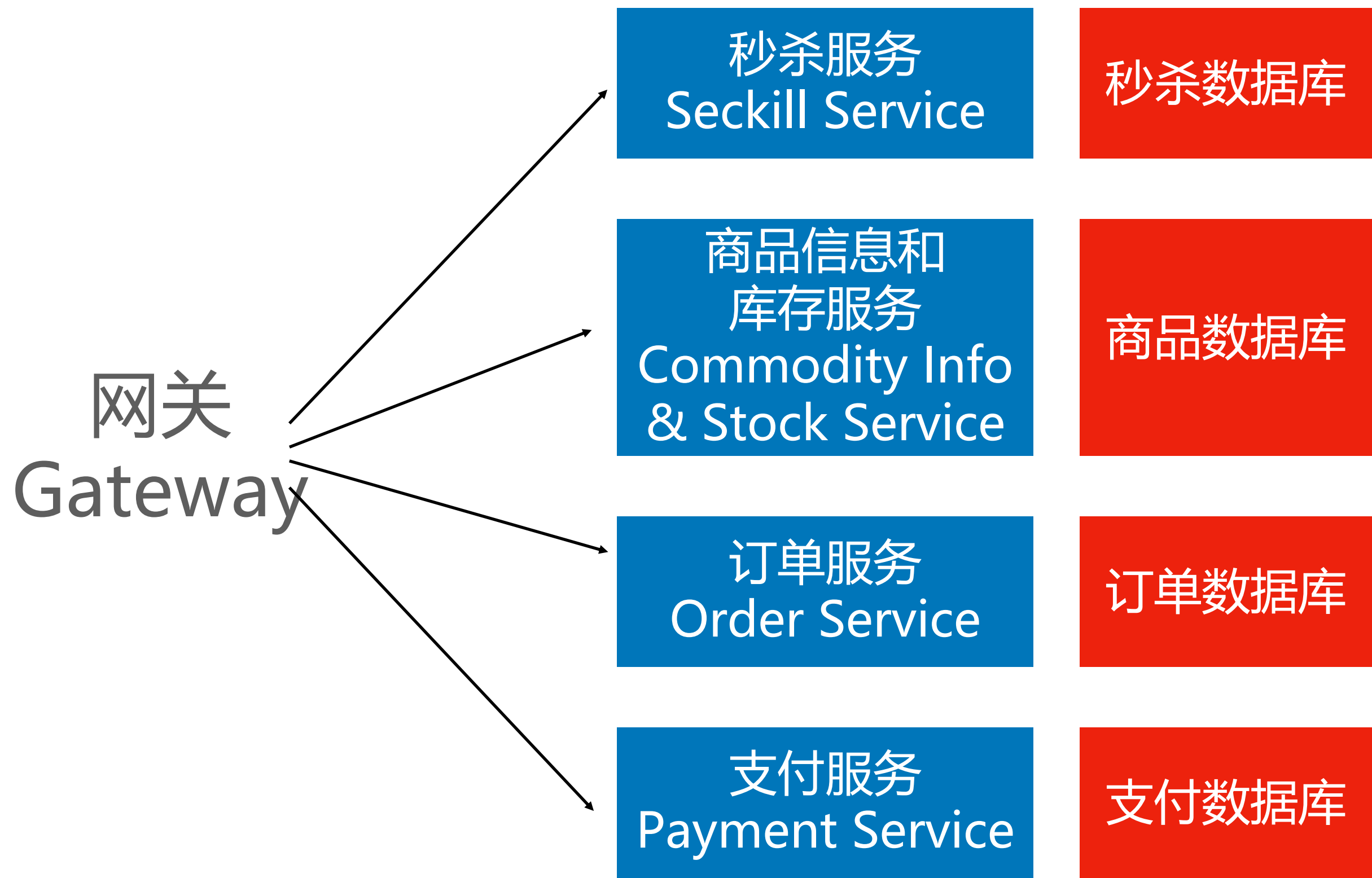
开发协作困难，各个部门的人都在开发同一个代码仓库。

Cascading Failure

级联故障，一个模块的故障导致整个服务不可用。

陷入某种单一技术和语言中。

数据库崩溃导致整个服务崩溃。



Decoupling Single Responsibility

各功能模块解耦，保证单一职责。

系统简单，升级某个服务不影响其他服务。

Expandability Expansion Shrink
扩展性强。可对某个服务进行单独扩容或缩容。

Cooperation

各个部门协作明晰。

Isolation

故障隔离。某个服务出现故障不完全影响其他服务。

可对不同的服务选用更合适的技术架构或语言。

数据库独立，互不干扰。

Service 存储

数据如何存储与访问

1. Select 为每个 Service 选择存储结构
2. Schema 细化表结构

商品信息表

commodity_info

商品id id	商品名称 name	商品描述 desc	价格 price
189	iPhone 11 64G	xxxxxxxx	5999

库存信息表

stock_info

库存id id	商品id commodity_id	活动id seckill_id	库存 stock	锁定 lock
1	189	0	1000000	0
2	189	28	100	5

秒杀活动表

seckill_info

秒杀id id	秒杀名称 name	商品id commodity_id	价格 price	数量 number
28	618 iPhone 11 64G 秒杀	189	4000	100

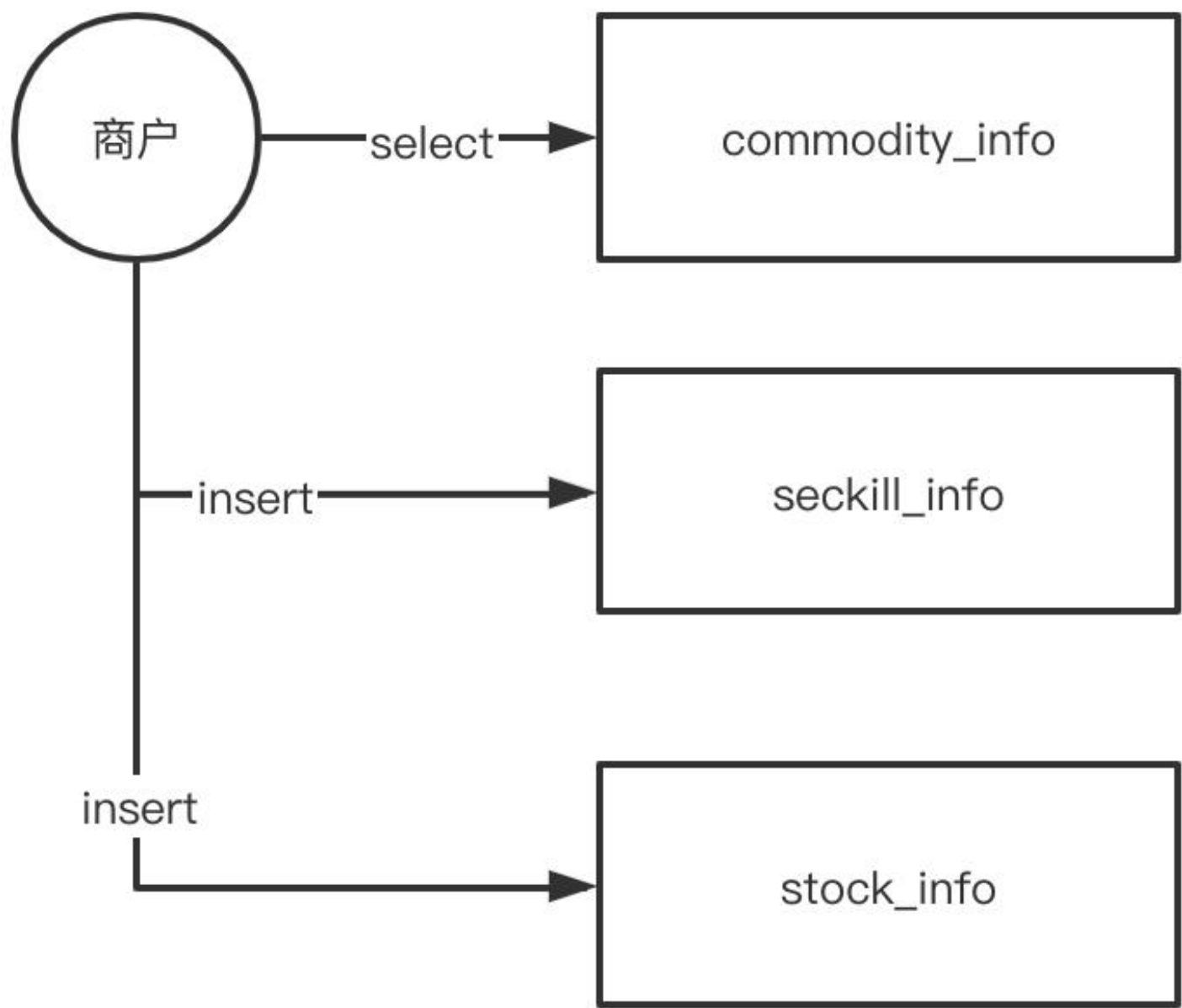
订单信息表

order_info

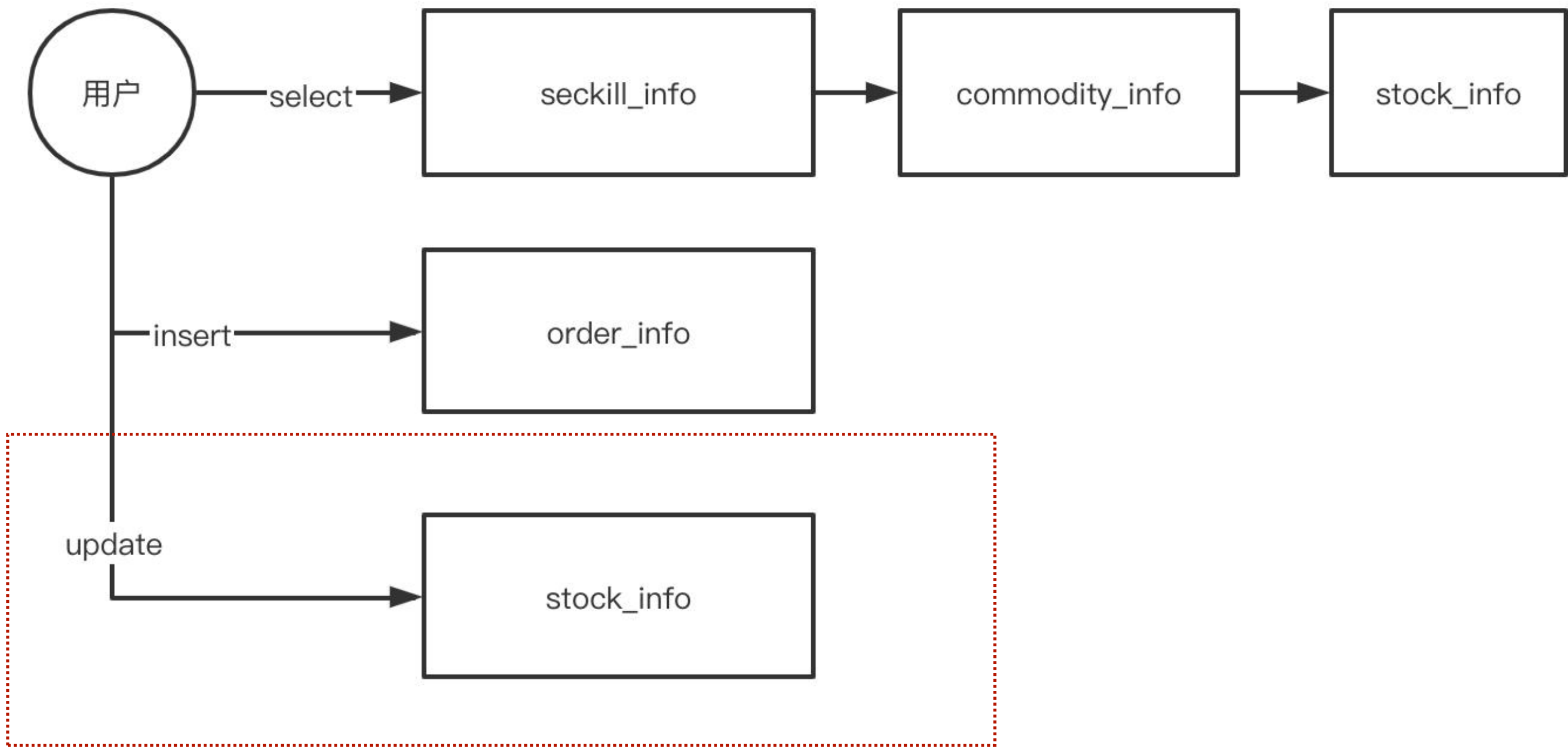
订单id id	商品id commodity_id	活动id seckill_id	用户id user_id	是否付款 paid
1	189	28	Jack	1

如何添加索引(Index)?

商家侧（京东自营、淘宝天猫店家）



用户侧



秒杀操作 - 扣减库存

读取判断库存，然后扣减库存

1. 查询库存余量

```
SELECT stock FROM `stock_info`  
WHERE commodity_id = 189 AND seckill_id = 28;
```

2. 扣减库存

```
UPDATE `stock_info` SET stock = stock - 1  
WHERE commodity_id = 189 AND seckill_id = 28;
```

并发导致超卖的问题如何解决？

秒杀操作 - 扣减库存

读取和判断过程中加上事务

Database Transaction

第十六章 系统设计的核心必考知识
识点：数据库索引与事务

1. 事务开始

```
START TRANSACTION;
```

2. 查询库存余量，并锁住数据

```
SELECT stock FROM `stock_info`  
WHERE commodity_id = 189 AND seckill_id = 28 FOR UPDATE;
```

3. 扣减库存

```
UPDATE `stock_info` SET stock = stock - 1  
WHERE commodity_id = 189 AND seckill_id = 28;
```

4. 事务提交

秒杀操作 - 扣减库存

Database Row Lock

使用 UPDATE 语句自带的行锁

1. 查询库存余量

```
SELECT stock FROM `stock_info`  
WHERE commodity_id = 189 AND seckill_id = 28;
```

2. 扣减库存

```
UPDATE `stock_info` SET stock = stock - 1  
WHERE commodity_id = 189 AND seckill_id = 28 AND stock > 0;
```

超卖问题解决了，其他问题呢？

1. 大量请求都访问 MySQL，导致 MySQL 崩溃。

对于抢购活动来说，可能几十万人抢 100 台 iPhone，实际大部分请求都是无效的，不需要下沉到 MySQL。

秒杀操作 - 库存预热

秒杀的本质，就是对库存的抢夺。

每个秒杀的用户来都去数据库查询库存校验库存，然后扣减库存，导致数据库崩溃。
Database Crash

MySQL 数据库单点能支撑 1000 QPS，但是 Redis 单点能支撑 10万 QPS，可以考虑将库存信息加载到 Redis 中。

直接通过 Redis 来判断并扣减库存。

一种主要将数据**存储于内存中**的非关系型的键值对数据库 (NoSQL 的一种), 但也可以将数据持久化 (Data Persistence) 到磁盘中。

Non-relational

支持多种数据**非关系型**的数据结构。

1. 字符串/数字 (STRING)
2. 哈希表 (HASH)
3. 链表 (LIST)
4. 集合 (SET)
5. 有序集合 (ZSET)

单线程的数据库。通过IO多路复用实现并发。

支持数据的主备容灾 (Disaster Tolerance) 存储。

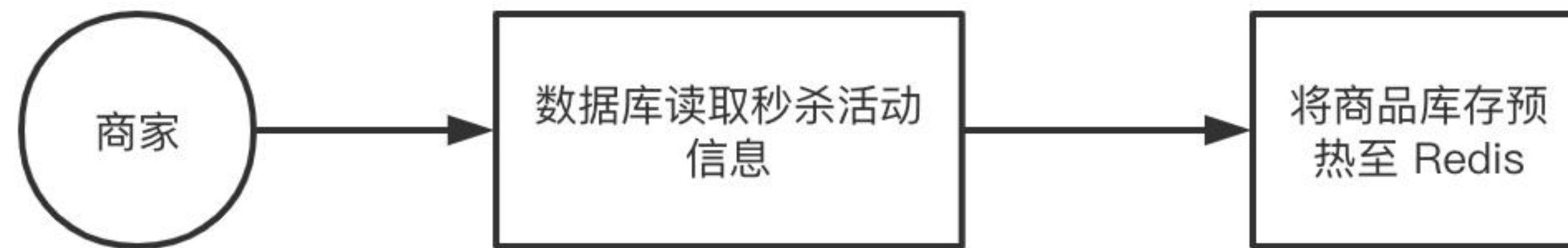
所有单个指令操作都是原子的, 即要么完全执行成功, 要么完全执行失败。多个指令也可以通过 Lua 脚本事务操作实现原子性。

因为都在内存中操作, 性能极高, 单机一般可支撑 10万数量级的 QPS。

可用作**数据缓存 (Cache)**、数据持久存储和消息队列 (Message Queue)。

什么时候进行预热 (Warm-up)?

活动开始前



语法: SET **KEY** **VALUE**

作用: 设置指定 key 的值

SET `seckill:28:commodity:189:stock` `100`

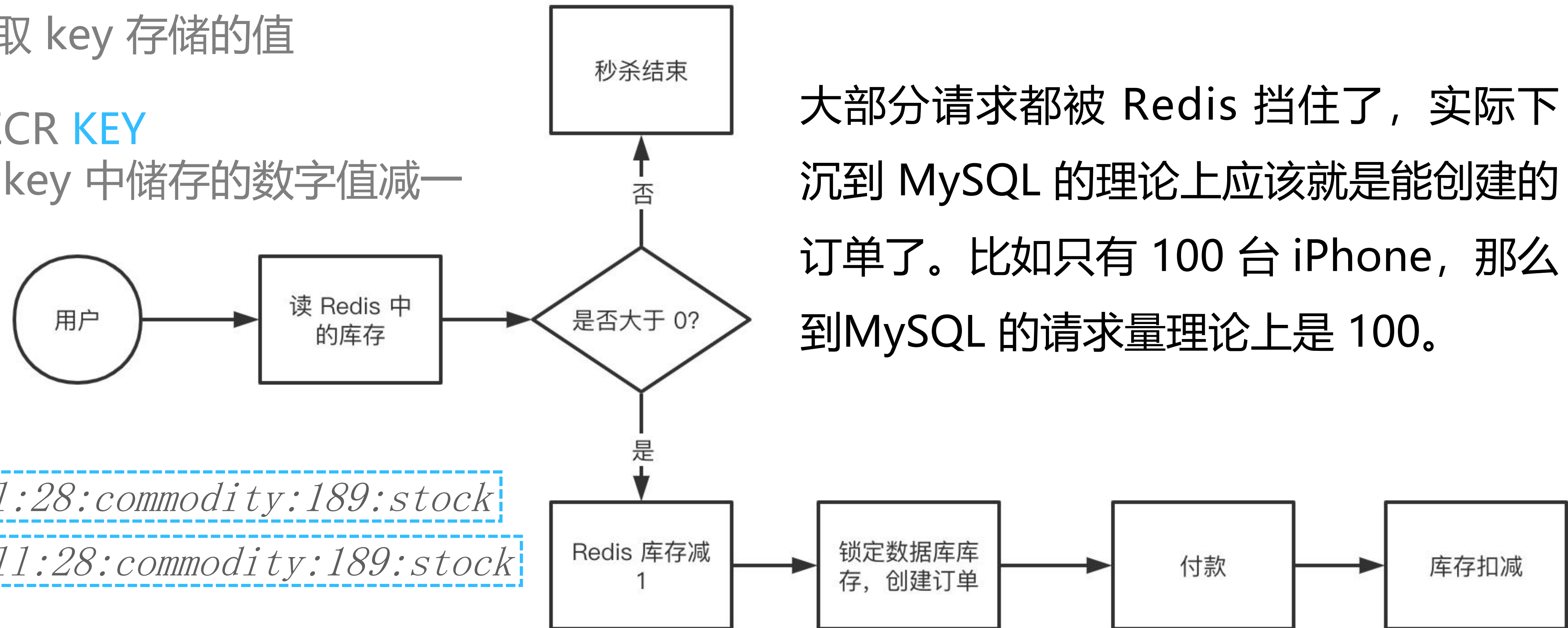
秒杀操作 - 通过 Redis 扣减库存

语法: GET **KEY**

作用: 获取 key 存储的值

语法: DECR **KEY**

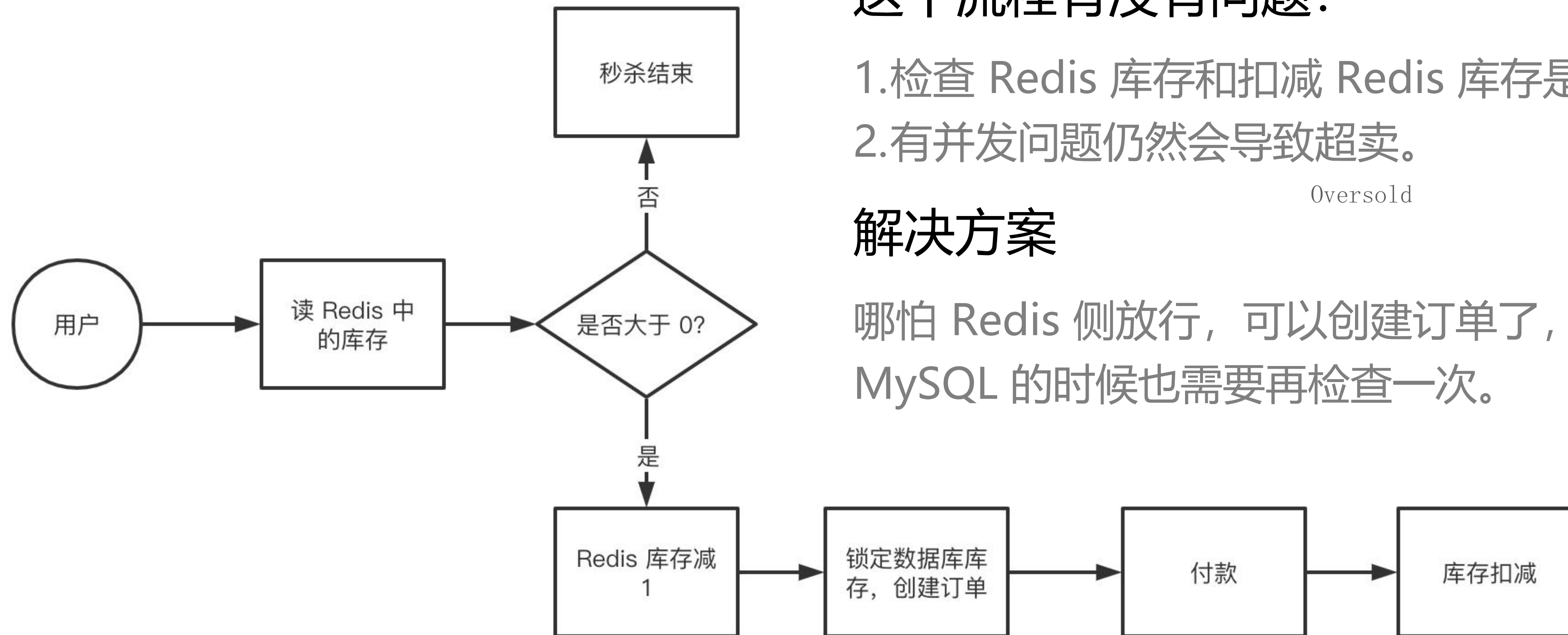
作用: 将 key 中储存的数字值减一



`GET seckill:28:commodity:189:stock`

`DECR seckill:28:commodity:189:stock`

秒杀操作 - 通过 Redis 扣减库存



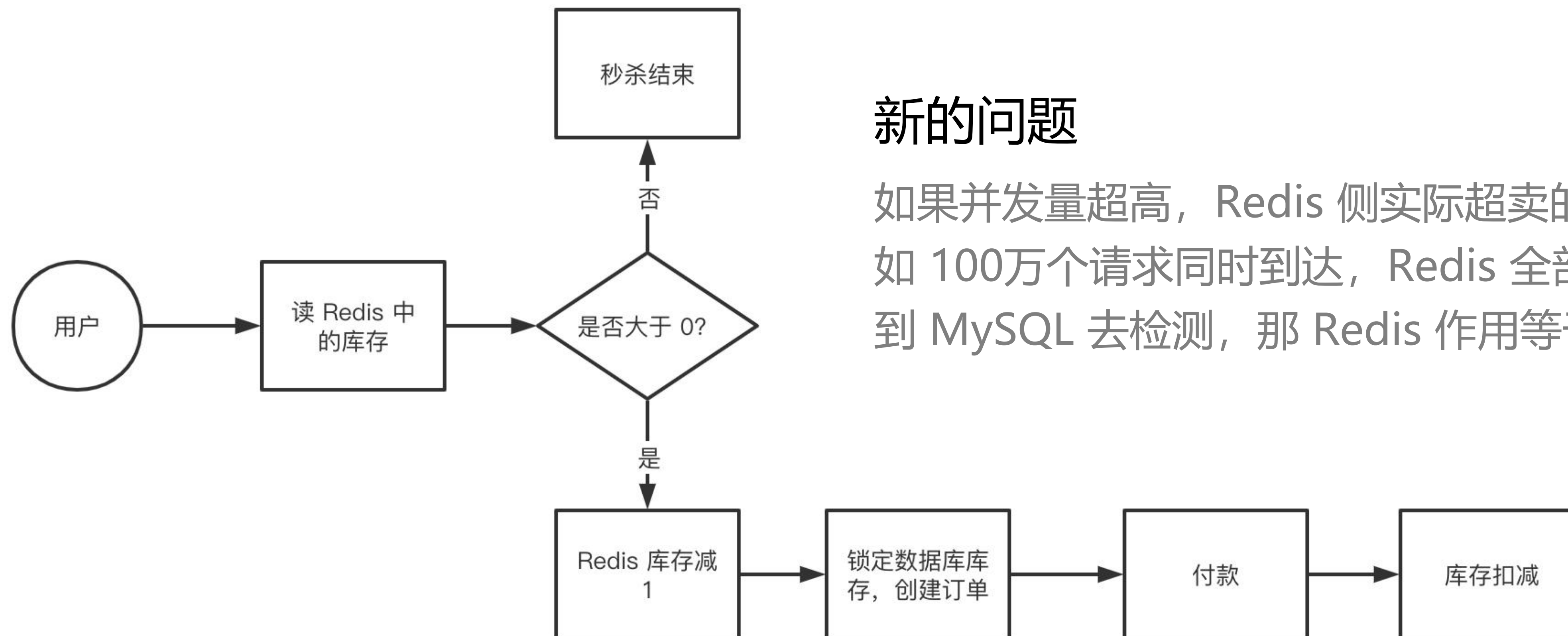
这个流程有没有问题?

- 1.检查 Redis 库存和扣减 Redis 库存是两步操作。
- 2.有并发问题仍然会导致超卖。

Oversold

解决方案

哪怕 Redis 侧放行, 可以创建订单了, 到 MySQL 的时候也需要再检查一次。



新的问题

如果并发量超高，Redis 侧实际超卖的量过大，如 100万个请求同时到达，Redis 全部放行。再到 MySQL 去检测，那 Redis 作用等于没有。

Atomic Operation

通过 Lua 脚本执行原子操作

Lua 脚本功能是 Redis 在 2.6 版本中推出，通过内嵌对 Lua 环境的支持，Redis 解决了长久以来不能高效地处理 CAS（check-and-set）命令的缺点，并且可以通过组合使用多个命令，轻松实现以前很难实现或者不能高效实现的模式。

Lua 脚本是类似 Redis 事务，有一定的原子性，不会被其他命令插队，可以完成一些 Redis 事务性的操作。

```
1  if (redis.call('exists', KEYS[1])>0) then
2      local stock = tonumber(redis.call('get', KEYS[1]));
3      if (stock <= 0) then
4          return -1
5      end;
6      redis.call('decr', KEYS[1]);
7      return stock - 1;
8  end;
9  return -1;
```


秒杀操作 - 通过 Redis 扣减库存

如果秒杀数量是1万台，或者10万台呢？

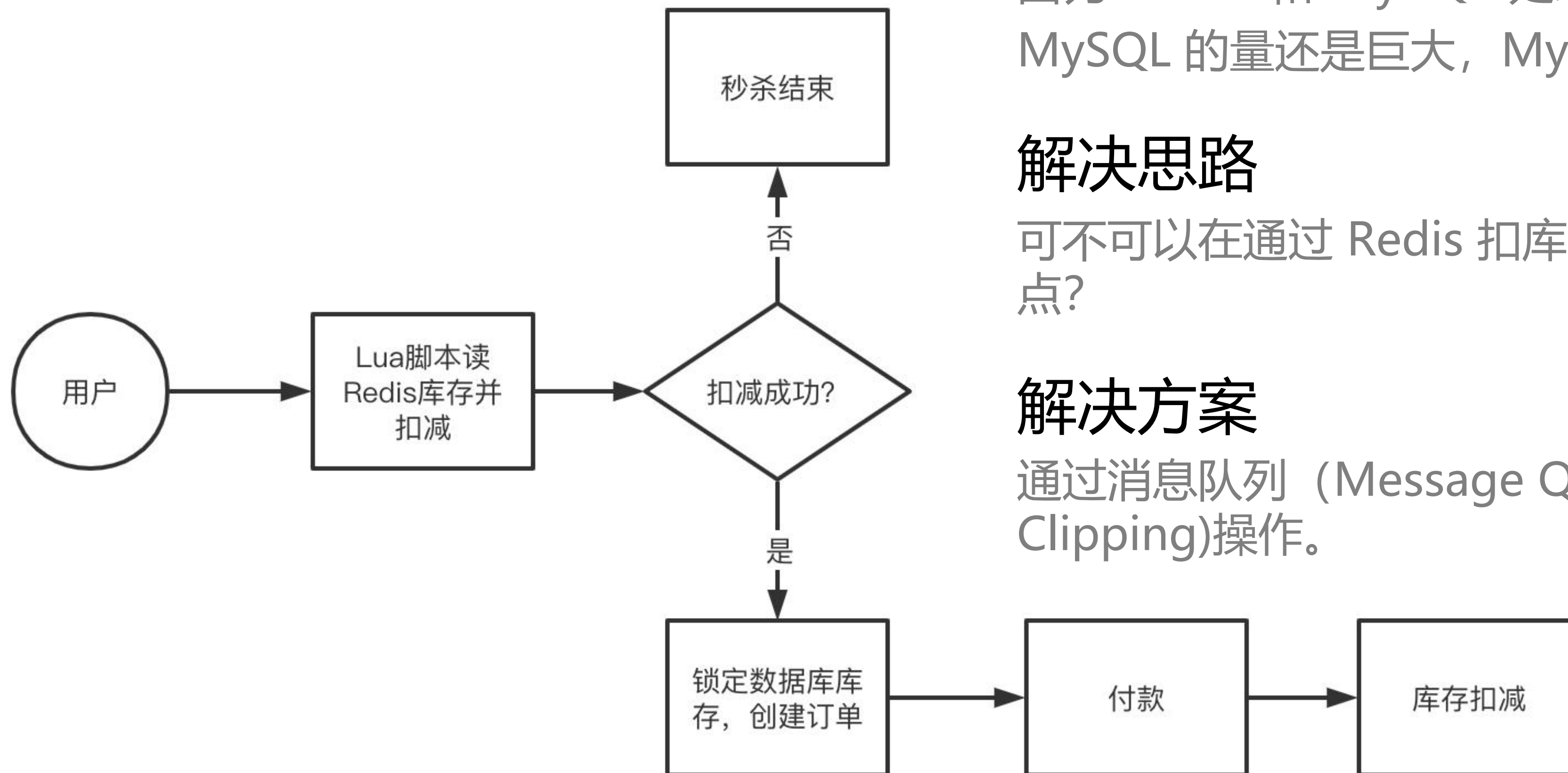
因为 Redis 和 MySQL 处理能力的巨大差异。实际下沉到 MySQL 的量还是巨大，MySQL 无法承受。

解决思路

可不可以在通过 Redis 扣库存后，到 MySQL 的请求慢一点？

解决方案

通过消息队列（Message Queue, MQ）进行削峰(Peak Clipping)操作。



消息队列简介

Producer/Consumer Model

一类基于**生产者/消费者模型**的组件。

Decoupling Asynchronous

用于实现两个不同的系统之间的**解耦**和**异步操作**。

Producer

Post Message

生产者可以高速地向消息队列中投递（生产）消息。

Consumer

Consume

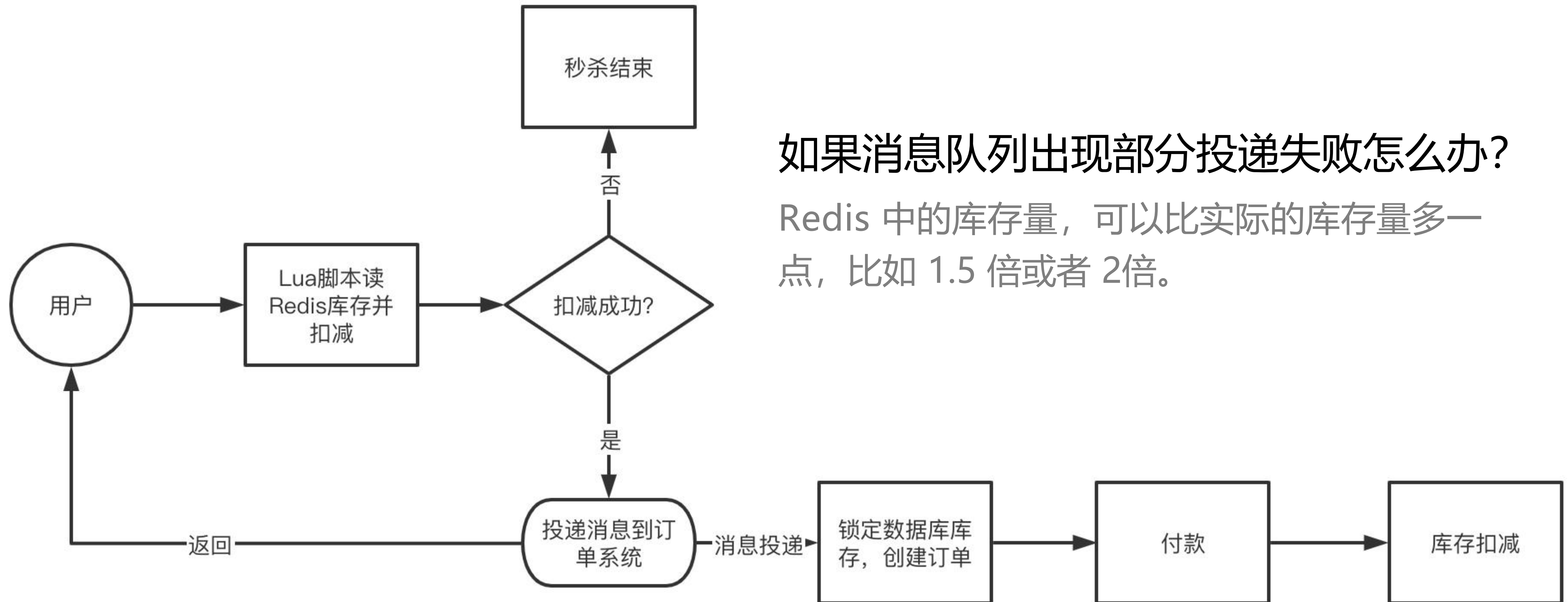
消费者可以按照自己的节奏去消费生产者投递的消息。

Retry

消息队列一般带有重试的能力。可以持续投递，直到消费者消费成功。



秒杀操作 - 通过消息队列异步地创建订单



如果消息队列出现部分投递失败怎么办?

Redis 中的库存量, 可以比实际的库存量多一点, 比如 1.5 倍或者 2倍。

下单时立即减库存。

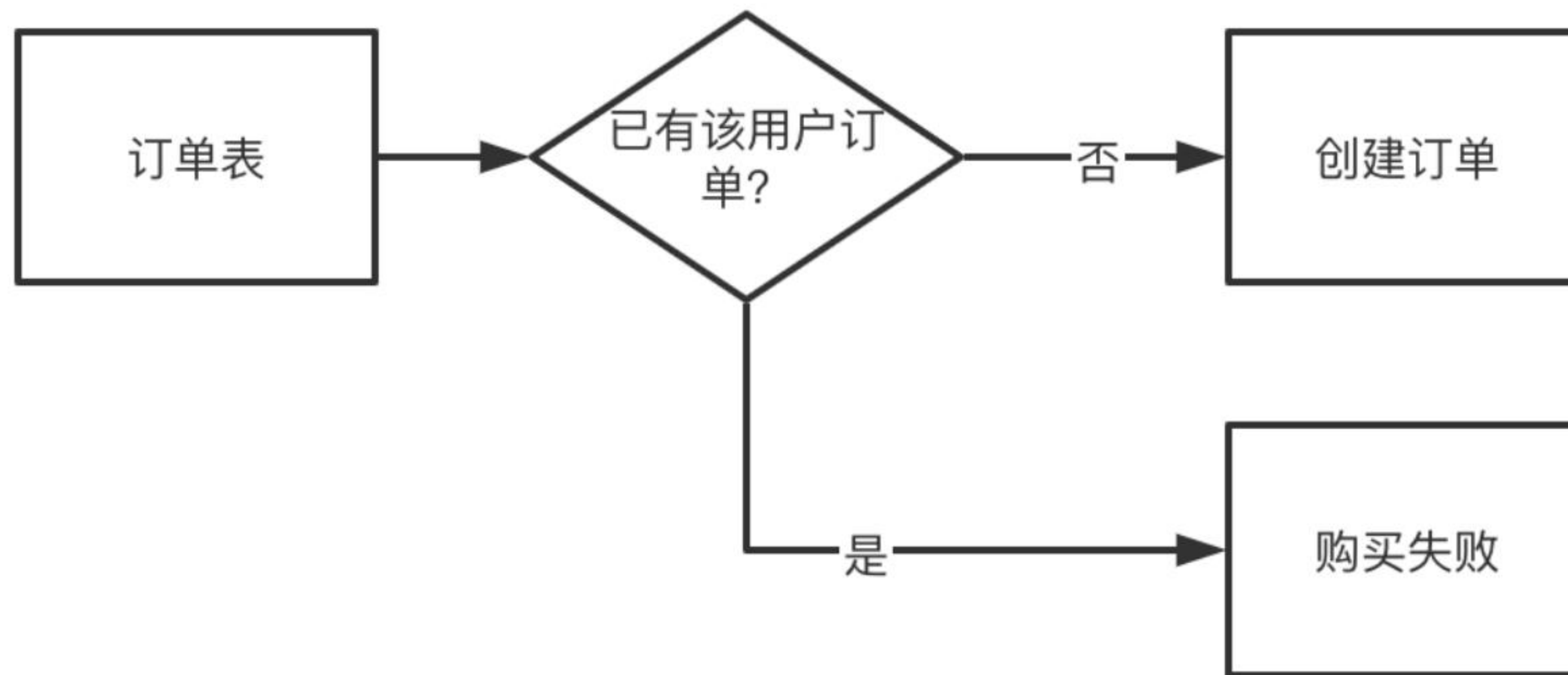
用户体验最好，控制最精准，只要下单成功，利用数据库锁机制，用户一定能成功付款。
可能被恶意下单。下单后不付款，别人也买不了了。

先下单，不减库存。实际支付成功后减库存。

可以有效避免恶意下单。
对用户体验极差，因为下单时没有减库存，可能造成用户下单成功但无法付款。

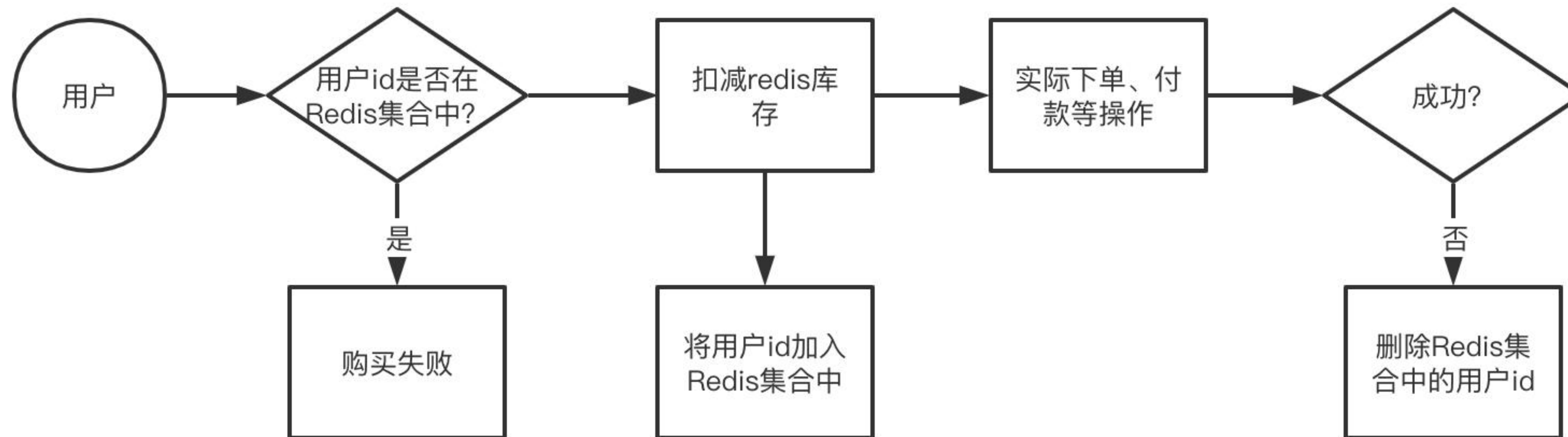
下单后锁定库存，支付成功后，减库存。

MySQL data verification 数据校验



Redis 数据校验^{data verification}

使用 Redis 提供的集合数据结构，将扣减 Redis 库存的用户 ID 写入。



语法: `SADD KEY VALUE1.....VALUEN`

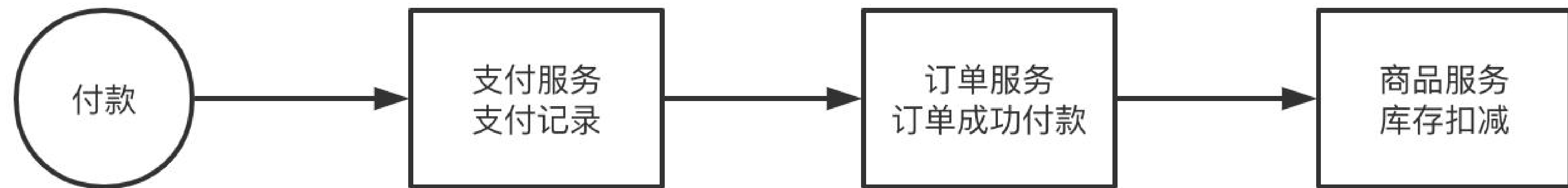
作用: 将一个或多个成员元素加入到集合中，已经存在于集合的成员元素将被忽略。

`SADD seckill:28:commodity:189:user uid1`

语法: `SISMEMBER KEY VALUE`

作用: 判断成员元素是否是集合的成员。

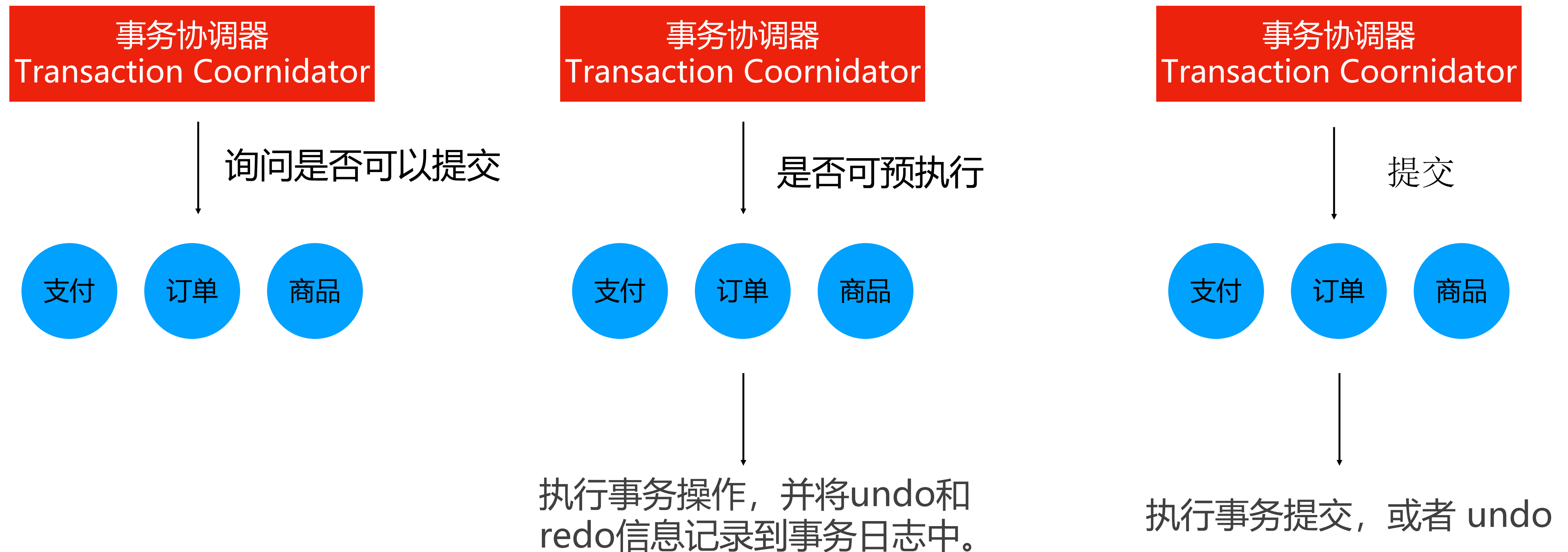
`SISMEMBER seckill:28:commodity:189:user uid1`



付款和减库存的数据一致性 - 分布式事务

保证多个存在于不同数据库的数据操作，要么同时成功，要么同时失败。主要用于强一致性的保证。

三阶段提交，有超时机制



Scale 拓展

如何优化系统
加分项

可能十万人抢购 100 台 iPhone，大部分请求是无效的。

Redis 能力高过 MySQL，但能力还是有限。

Redis 库存扣减完毕后，是否后面的请求可以直接拒绝了？

防止刷爆商品页面

前端资源静态化

CDN

前端静态资源
Front-end Static
Resources

网关

秒杀服务

秒杀数据库

商品信息和
库存服务

商品数据库

订单服务

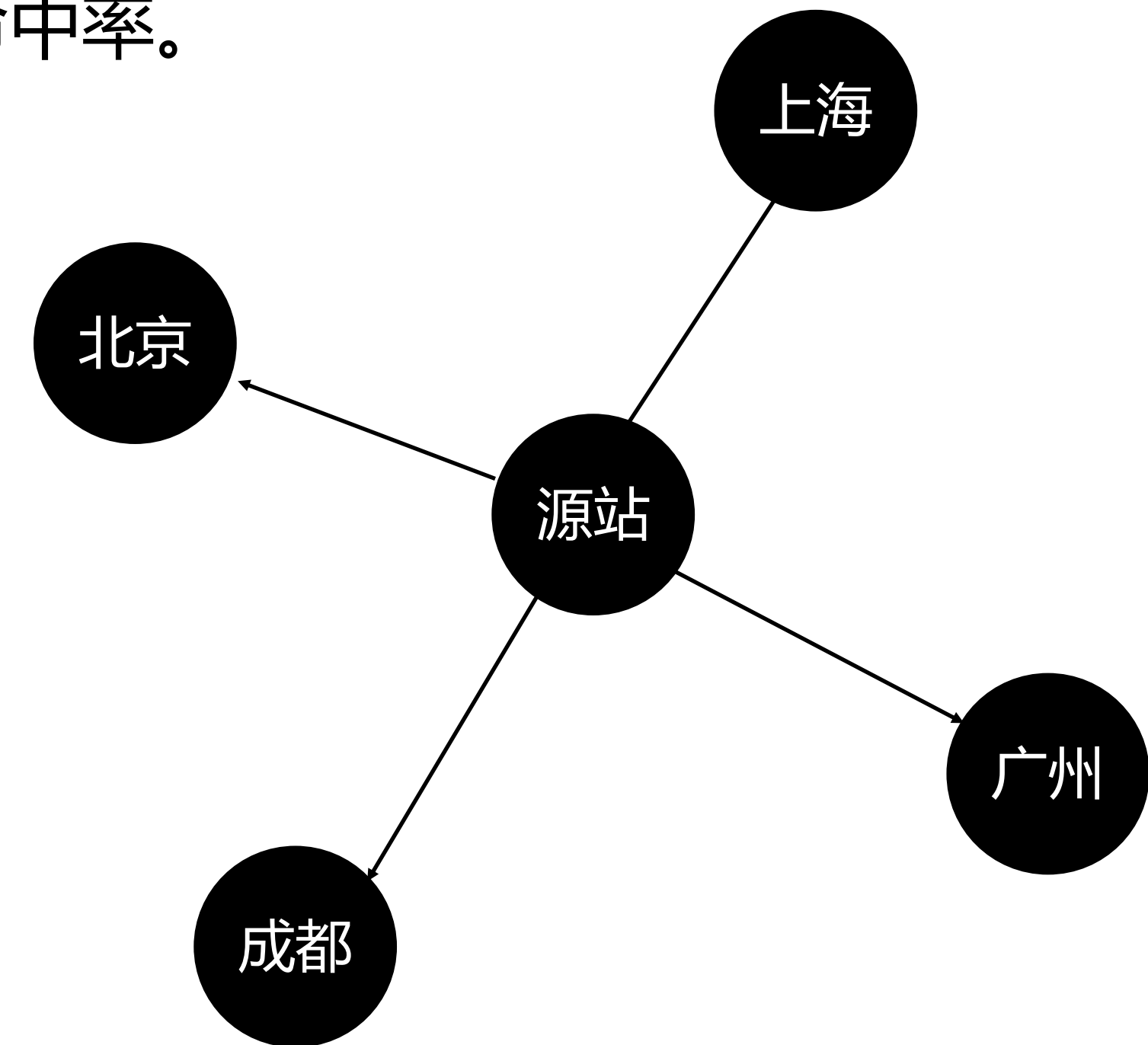
订单数据库

支付服务

支付数据库

CDN 的全称是 Content Delivery Network, 即内容分发网络。

CDN 是依靠部署在各地的边缘服务器, 通过中心平台的负载均衡、内容分发、调度等功能模块, 使用户就近获取所需内容, 降低网络拥塞, 提高用户访问响应速度和命中率。



front-end current-limiting

前端限流

点击一次后，按钮
短时间内置灰

购买

购买

部分请求直接
跳转到「繁忙页」

防止刷爆商品页面

未开始抢购时，禁用抢购按钮。

如何计算倒计时？

1. 打开页面获取活动开始时间，然后前端页面开始倒计时
2. 打开页面获取距离活动开始的时间差，然后前端页面开始倒计时
3. 前端轮询 (Poll) 服务器的时间，并获取距离活动开始的时间差

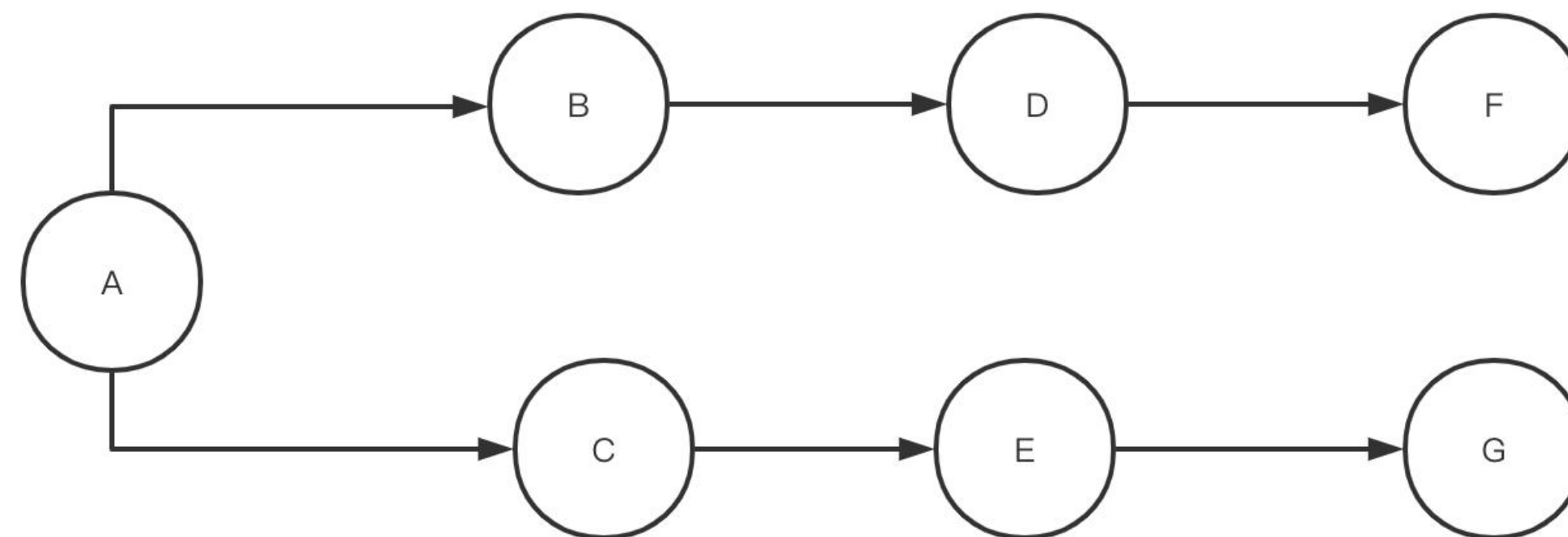


尽量不要影响其他服务，尤其是非秒杀商品的正常购买。

服务雪崩 (Avalanche)

Microservice
多个微服务之间调用的时候，假设 微服务A 调用 微服务B 和 微服务C，微服务B 和微服务C 又调用其他的微服务，这就是所谓的“扇出 (Fan-out)”，如扇出的链路上某个微服务的调用响应式过长或者不可用，对 微服务A 的调用就会占用越来越多的系统资源，进而引起系统雪崩，所谓的“雪崩效应”。

服务雪崩效应是一种因“服务提供者”的不可用导致“服务消费着”的不可用并将这种不可用逐渐放大的过程。



尽量不要影响其他服务，尤其是非秒杀商品的正常购买。

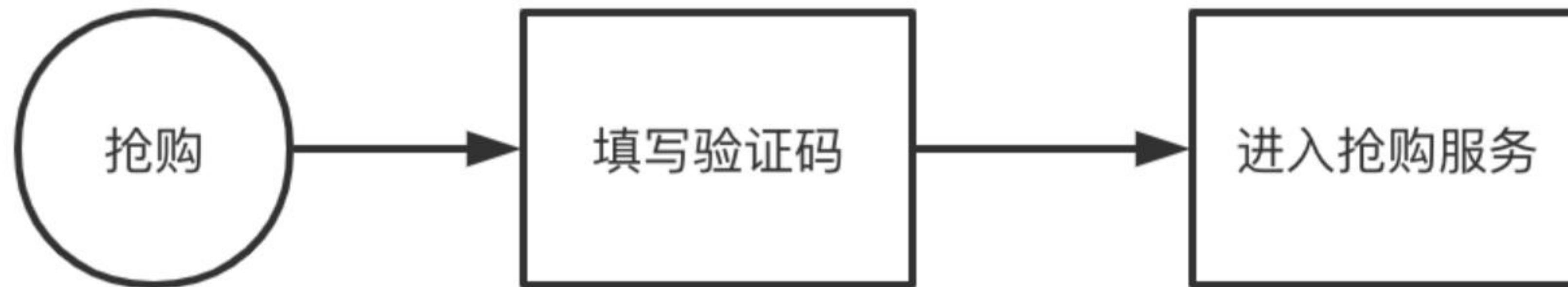
服务熔断 (Fuse or Circuit-breaker)

熔断机制是应对雪崩效应的一种微服务链路保护机制，当扇出链路的某个微服务不可用或者响应时间太长时，熔断该节点微服务的调用，快速返回“错误”的响应信息。当检测到该节点微服务响应正常后恢复调用链路。

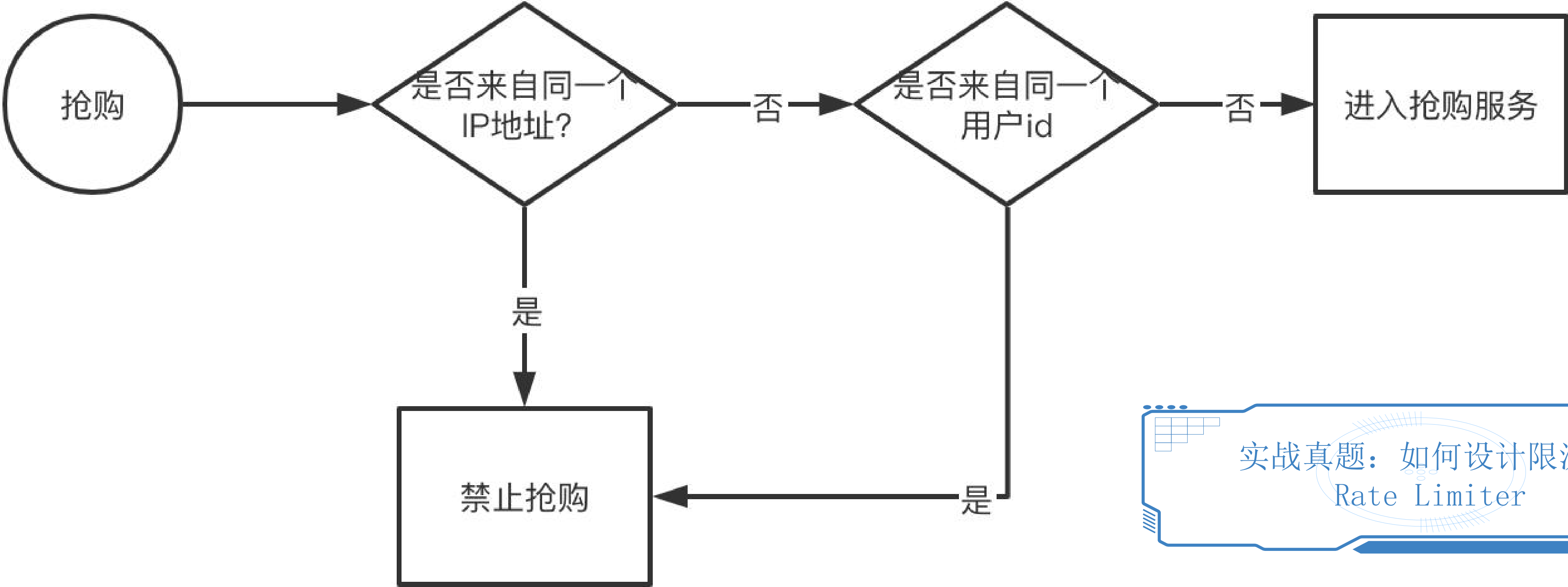
Netflix Hystrix

Alibaba Sentinel

验证码机制 Verification Code Mechanism



限流机制
Ratelimit Mechanism



实战真题：如何设计限流器
Rate Limiter

黑名单机制 Blacklist Mechanism

1. 黑名单 IP 地址
2. 黑名单用户ID

秒杀系统 vs 订票系统

在业务上，他们有哪些差异？

100 台 iPhone 没有区别

但是 100 张同一车次的火车票，有座位的区别（暂时忽略一等座二等座等）

车次信息表

train_number_info

车次id id	车次 number	车次描述 desc	价格 price
189	G1100	xxxxxxxxx	100

库存信息表

stock_info

库存id id	商品id commodity_id	座位号 seat	库存 stock	锁定 lock
1	189	1A	1	0
2	189	1B	1	1