详细设计说明书

1. 引言

1.1编写目的

本项目为厦门大学信息学院软件工程专业软件工程系大三上学期《面向对象分析与设计》、《软件工程导论》和《JavaEE平台技术》课程的大作业。本文档的编写旨在详细介绍课程大作业中售后模块和服务模块的设计方案。

项目的整体目标是开发一个支持高并发、大负载的电子商城系统（OOMALL）的订单模块，用户可通过该系统完成与主流电商平台类似的购物流程，包括选购商品、发货、售后等功能。本小组负责售后和服务模块的开发工作。

项目将使用Java编程语言进行开发，由五人团队采用敏捷开发方式，并对软件的功能、性能和安全性进行全面测试。

OOMALL电子商城系统的售后与服务模块设计是项目开发的第三阶段。本阶段的主要任务是在概要设计说明书的基础上，详细阐述概要设计中各功能模块的实现过程，并进一步完善模块的内部设计与功能实现。

本文档针对本小组负责的售后与服务模块进行了系统架构的细化，详细描述了模块的实现过程，包括相关算法、逻辑流程、数据结构设计、物理设计等内容，全面展示了三个模块的具体实现过程。

1.2项目背景

本项目是厦门大学信息学院软件工程专业《软件工程》、《面向对象分析与设计》以及《JavaEE平台技术》三门课程的联合课程设计。该项目于2022年秋季学期由软件工程专业2020级学生完成了第一次迭代，于2023年秋季学期由软件工程专业2021级学生启动了第二次迭代，并于2024年秋季学期由软件工程专业2022级学生启动了第二次迭代。

1.3定义

DDoS 攻击（分布式拒绝服务攻击）：DDoS 是一种常见的网络攻击方式，它通过利用多台计算机组成攻击平台，向目标服务器或网络发送大量的请求，占用目标的资源，导致目标系统过载或瘫痪。攻击者通过控制大量的“僵尸”计算机（通常是被恶意软件感染的计算机）来实施攻击，使得单一来源的攻击难以被识别或阻止。

数据分片：数据分片是一种将大规模数据集拆分成较小的数据块的技术。这些数据块被分布存储在不同的节点或服务器上。这种技术广泛用于分布式数据库系统中，能够提高存储效率和数据处理的性能，尤其在大数据处理场景下，数据分片能够确保系统能够扩展并处理更高的负载。

分布式数据库：分布式数据库是一种将数据存储在不同的物理位置的数据库架构。数据可能存储在同一地点的多个计算机上，或分布在多个地理位置的计算机网络中。分布式数据库能够提供更高的可靠性和可扩展性，因为它能够处理更大的数据量并在多个节点间分担负载。

SSL（安全套接层）：SSL 是一种加密协议，最早用于通过不安全的互联网连接进行安全通信。SSL 协议为传输中的数据提供加密和完整性检查，确保通信内容不会被窃听或篡改。SSL 已被 TLS（传输层安全）协议所取代，后者提供了更强的加密标准和安全性。

TLS（传输层安全）：TLS 是一种加密协议，用于在两个通信应用程序之间提供机密性和数据完整性。TLS 协议是 SSL 的继任者，设计上提供了更强的安全性能，广泛应用于 HTTPS、电子邮件等服务中，确保数据在传输过程中的安全。

访问令牌：访问令牌是 Windows 操作系统中的一个概念，用于管理用户和计算机系统之间的安全通信。当用户登录时，操作系统会生成一个访问令牌，其中包含用户的安全标识符（SID）以及该用户被授权的访问权限。访问令牌用于在系统中验证用户的身份，并允许用户访问特定的资源或执行操作。

刷新令牌：刷新令牌是一种授权机制，通常用于OAuth等认证协议中。它允许用户在访问令牌过期后，获取新的访问令牌。通过刷新令牌，用户不需要重新登录就可以继续访问系统。刷新令牌通常具有较长的有效期，并且可以用来获取新令牌，从而保证系统的安全性和用户体验。

这些术语在网络安全、系统设计、数据库管理等多个领域中具有广泛应用，理解这些概念对于构建高效、可靠的电子商务平台系统至关重要。

1.4参考资料

软件工程术语（GB/T11475-2006）》

《需求规格说明书》

中国人民银行办公厅。 关于进一步加强无证经验支付业务整治工作的通知。 银办发[2017]217 号文

中国人民银行。 关于规范支付创新业务通知。 银办发[2017]281 号文

中国人民银行。 关于印发 < 条码支付业务规范（试行）>的通知。 银办发[2017]296 号文

郑志成。 京东到家支付平台的高可用性架构计

https://www.zhihu.com/question/527868488/answer/2438919186

微信支付。

https://pay.weixin.qq.com/wiki/doc/apiv3/apis/chapter8_1_1.shtm

支付宝互联网平台交易查询接口。

https://opendocs.alipay.com/open/02e7gm?ref=api

2. 总体设计

2.1需求概述

1.售后申请

顾客操作：顾客通过订单页面提交售后申请，可能包括退货、换货、维修等类型。顾客需要选择售后类型，并提供必要的证据（如照片、视频、描述等）以证明申请的合理性。

系统功能：系统应允许顾客填写申请表单，并上传相关凭证（如商品瑕疵照片、购买凭证、问题描述等）。还可以设置售后类型的选择（如退货、换货、维修等）以及申请的期限和条件。

审核提示：系统应显示申请的状态，例如“等待审核”，“审核通过”，“审核拒绝”等，并允许顾客查看审核结果。

2. 审核售后

商户操作：商户或售后客服对顾客提出的售后申请进行审核。商户根据申请的情况判断是否合理，并可以要求顾客提供更多信息或证据。

审核标准：商户需依据售后政策、产品质量、用户描述等信息判断申请是否符合条件。商户应按照规定的时间限制作出审核决定。

审核结果反馈：审核通过则进入后续处理流程（如退款、换货等）；审核未通过则拒绝申请，并反馈拒绝原因。

3. 取消售后

商户操作：如果商户决定拒绝顾客的售后申请，可以取消售后，并在系统中标记为“售后拒绝”状态。同时，商户需填写拒绝理由，并通过系统反馈给顾客。

系统提示：系统应向顾客展示被拒绝的理由，以避免产生争议。例如，拒绝理由可以是“产品未达到退货条件”、“超出退货时限”等。

顾客处理：顾客接到拒绝通知后，可以选择进一步沟通或提出申诉。

4. 售后收件

收件准备：对于退货或换货的情况，顾客需将原商品寄回。商户需要提供退货或换货的物流信息，并确保收件环节顺利进行。

商品检查：商户收到退货商品后，需进行检查。检查标准应提前明确，比如检查商品的完好程度、是否有使用痕迹等。

检查结果：

检查通过：若商品符合退货或换货条件，商户应完成退款或换货操作。

检查不通过：若商品不符合退货或换货条件，商户可以拒绝退款或换货，并将商品退回顾客。

5. 完成售后

服务型售后：对于服务类售后（如维修、安装等），售后应在服务完成后结束。如果服务未按约定完成，商户应及时与顾客沟通并提供解决方案。

退货型售后：对于退货类售后，顾客需确认退款金额无误，并等待商户处理。商户在确认退款操作完成后，售后流程结束。

换货型售后：对于换货型售后，一旦换货订单生成且商品送达顾客，售后流程应结束，商户可以确认换货完成。

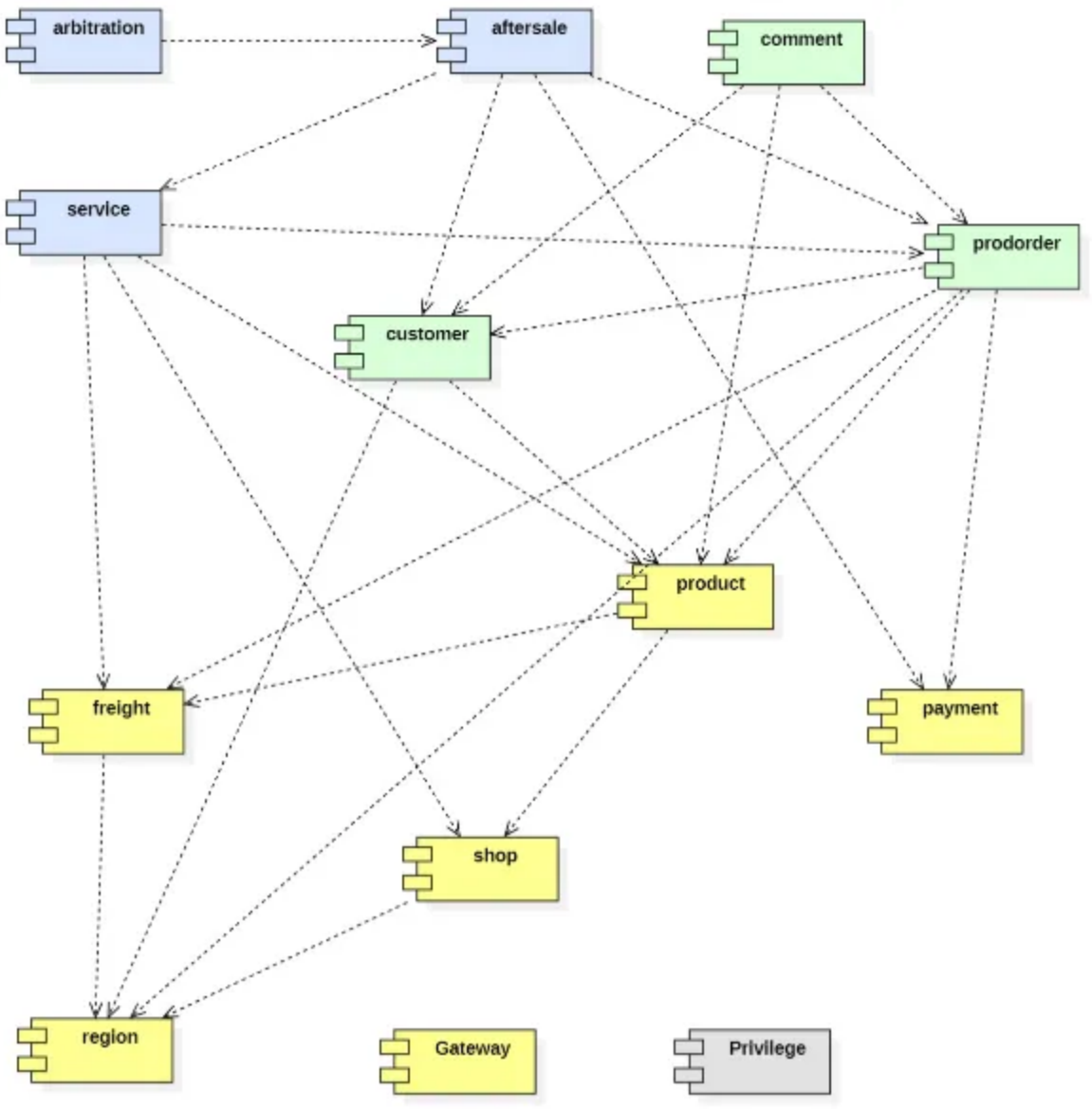
重要注意事项：

时间管理：整个售后流程中的每一步都应有明确的时间限制。例如，售后申请的审核时限、收件检查的时限等，以提高效率并保证顾客满意度。

系统通知与跟踪：系统应及时通知顾客每一步操作的状态，并允许顾客查询售后进度。

售后政策的透明度：商户应确保顾客了解其售后政策，包括退货期限、售后类型、审核标准等，以避免因信息不对称而产生不必要的争议。

2.2软件结构



3. 程序描述

【逐个模块给出以下的说明：】

3.1功能

3.1.1仲裁模块

1.获得仲裁单的所有状态：可以通过API获得仲裁单的所有状态。

2.顾客申请售后仲裁：顾客可以通过API申请售后仲裁，创建一个售后单等待审核。

3.仲裁员查询申请的仲裁单信息：仲裁员可以通过API查询正处于待审核状态的仲裁单信息，选择其中的仲裁单进行审核。

4.仲裁员查询自己负责的仲裁单信息：仲裁员可以通过API查询自己审核与仲裁过的仲裁单信息。

5.仲裁员根据仲裁id查询仲裁单详细信息：仲裁员可以通过API使用仲裁单的id查询特定仲裁单信息，该仲裁单需要时仲裁员负责的或未被审核的仲裁单。

6.顾客取消仲裁：顾客可以通过API取消未仲裁完毕的仲裁单。

7.仲裁员审核仲裁单：仲裁员可以通过API对受理的仲裁单进行审核。

8.仲裁员仲裁结案：仲裁员可以通过API对仲裁单进行判决，决定胜诉方。

3.1.2售后模块

1.获得售后单的所有状态：可以通过API获得售后单的所有状态。

2.顾客提交售后单：顾客可以通过API对已完成的订单中的商品提起售后单。

3.顾客查询所有的可申请售后的订单明细：顾客可以通过API查询所有的未申请过退换货的订单明细。

4.顾客查询所有的售后单信息：顾客可以通过API查询所有自己提起过的售后单信息。

5.顾客根据售后单id查询售后单信息：顾客可以通过API使用id查询自己提起过的特定售后单信息。

6.顾客修改售后单信息：顾客可以通过API修改售后单中的地址，售后商品的数量，申请售后的原因，联系人以及联系电话信息。

7.顾客取消售后单：顾客可以通过API取消处于申请和处理中状态的售后单。

8.平台管理员查看所有售后单：平台管理员可以通过API查看所有售后单。

9.管理员根据售后单id查询售后单信息：平台管理员可以通过API使用售后单ID查看特定售后单信息。

10.商户验收买家的退（换）货：商户可以通过API确认顾客的退（换）货是否符合描述。

11.平台管理员审核同意/不同意：商户可以通过API审核顾客提起的售后单。

12.通过售后单id查询售后轨迹：可以通过API通过特定售后单id查询售后轨迹。

13.店铺查询售后的服务单信息：商户可以通过API查询售后单对应的服务单信息

3.1.3服务模块

1.

3.2性能

服务和售后模块需要支持高负载和大并发，能够在高峰时段处理大量顾客频繁发起的售后单操作。系统需满足高并发要求，支持500个用户同时提交售后或服务单，同时确保售后单和服务单信息的一致性及正确的关联关系。同时，系统需具备高负载能力，确保在1秒内处理1000个下单操作的性能需求。

3.3输入项目

售后模块

功能	用户输入信息
顾客提交售后申请	订单 ID，售后服务信息
根据售后状态分类查询售后列表	售后状态，页码，每页数目
顾客查询所有的可申请的订单明细	页码，每页数目
顾客查询售后单信息	售后单 ID
顾客修改售后单信息	售后单 ID，可修改的信息：地址、售后商品的数量、申请售后的原因、联系人以及联系电话
顾客取消售后	售后单 ID
顾客申请售后仲裁	售后单 ID，售后信息
顾客申请二次仲裁	售后单 ID，仲裁详细信息
顾客取消仲裁	售后单 ID，仲裁单 ID

查询售后单对应的物流单号	售后单 ID
商铺审核售后	商铺 ID，售后单 ID，处理意见
商铺验收售后商品	商铺 ID，售后单 ID，处理意见
商铺取消售后	商铺 ID，售后单 ID
商铺应诉仲裁	商铺 ID，仲裁单 ID，仲裁详细信息
管理员受理仲裁单	商铺 ID（只能为 0），仲裁单 ID，仲裁员信息
管理员仲裁结案	商铺 ID（只能为 0），仲裁单 ID，仲裁结果信息
查看所有售后单	商铺 ID（管理员为 0），开始时间，结束时间，页码，每页数目，售后类型，售后状态
查询售后单信息	商铺 ID（管理员为 0），售后单 ID
查询顾客申请的仲裁单信息	商铺 ID（管理员为 0），页码，每页数目
查询仲裁单详情	商铺 ID（管理员为 0），仲裁单 ID

服务模块

功能	用户输入信息

商户选择商品服务	商铺 ID，商品 ID，服务商 ID
商户修改商品服务	商铺 ID，商品服务 ID
商户取消商品服务	商铺 ID，商品服务 ID
商户通过商品和地区查询服务商	商铺 ID，商品 ID，地区 ID
商户通过服务商 id 查询服务商	商铺 ID，服务商 ID
商户取消服务商的所有服务	商铺 ID，服务商 ID
获得服务单的所有状态	无
商户创建服务单	商铺 ID，商品 ID，地区 ID
获得服务单信息	商铺 ID
商户根据 id 查询服务单	商铺 ID，服务单 ID
取消服务单	商铺 ID，服务单 ID
服务商注册账号	服务商名称，密码，联系人，保证金等
服务商申请变更账号信息	服务商 ID，服务商信息
服务商注销账号	服务商 ID
服务商查看账户信息	服务商 ID
管理员查询服务商	管理员 ID，服务商类型，状态，名称
管理员审核服务商开户	管理员 ID，服务商 ID，审核结果
管理员审核服务商变更	管理员 ID，服务商 ID，审核结果

管理员暂停服务商	管理员 ID，服务商 ID
管理员恢复服务商	管理员 ID，服务商 ID
管理员审核服务	管理员 ID，服务 ID，审核结果
服务商查询服务单信息	服务商 ID
服务商根据 id 查询服务单信息	服务商 ID，服务单 ID
服务商接受服务单	服务商 ID，服务单 ID
服务商接受服务单	服务商 ID，服务单 ID
服务商收到寄出商品	服务商 ID，服务单 ID
服务商完成服务单	服务商 ID，服务单 ID
服务商取消服务单	服务商 ID，服务单 ID
服务商定义服务	服务商 ID，商品类别 ID，地区 ID
服务商取消服务	服务商 ID，服务 ID
服务商修改服务范围	服务商 ID， 服务 ID，地区 ID
服务商查询服务	服务商 ID，服务状态， 服务地区
服务商根据 id 查询服务	服务商 ID，服务 ID

3.4输出项目

售后模块

功能	正常输出	异常输出
顾客提交售后申请	系统提示操作成功并返回售后单号	

据售后状态分类查询售后列表	系统提示操作成功并按页返回售后信息列表	
顾客查询订单明细	系统提示操作成功并返回可售后的订单详细信息	
顾客查询售后单信息	系统提示操作成功并返回售后单详细信息	

顾客修改售后单信息	系统提示操作成功	
顾客取消售后	系统提示操作成功	
顾客申请售后仲裁	系统提示操作成功并返回售后仲裁信息	系统返回错误码 705（ 已经在仲裁中的售后不允许再仲裁）
顾客申请二次仲裁	系统提示操作成功并返回售后二次仲裁信息	系统返回错误码 705（已经在仲裁中的售后不允许再仲裁）
顾客取消仲裁	系统提示操作成功	
查询售后单对应的物流单号	系统提示操作成功并返回物流单号	
商铺审核售后	系统提示操作成功	
商铺验收售后商品	系统提示操作成功	
商铺取消售后	系统提示操作成功	
商铺应诉仲裁	系统提示操作成功并返回仲裁单详情	
管理员受理仲裁单	系统提示操作成功	
管理员仲裁结案	系统提示操作成功	
查看所有售后单	系统提示操作成功并按页返回售后单列表	

查询售后单信息	系统提示操作成功并返回售后单详细信息	
查询顾客申请的仲裁单信息	系统提示操作成功并按页返回仲裁单列表	
查询仲裁单详情	系统提示操作成功并返回仲裁单详细信息	

服务模块

功能	异常输出	异常输出
商户选择商品服务	系统返回错误码 801（商品服务已经存在，无法重复选择）	系统返回错误码 801（商品服务已经存在， 无法重复选择）
商户修改商品服务	系统提示操作成功	系统返回错误码 504（修改商品服务时发现该服务商被禁止或者服务商并不存在
商户取消商品服务	系统提示操作成功	

商户通过商品和地区查询服务商	系统提示操作成功并按页返回服务商信息	
商户通过服务商 id 查询服务商	系统提示操作成功并返回服务商信息	
商户取消服务商的所有服务	系统提示操作成功	
获得服务单的所有状态	系统提示操作成功并返回服务单的所有状态	

商户创建服务单	系统提示操作成功并返回服务单 ID	
获得服务单信息	系统提示操作成功并按页返回服务单信息列表	
商户根据 id 查询服务单	系统提示操作成功并返回服务单信息	
取消服务单	系统提示操作成功	
服务商注册账号	系统提示操作成功并返回服务商 ID	
服务商申请变更账号信息	系统提示操作成功	
服务商注销账号	系统提示操作成功	
服务商查看账户信息	系统提示操作成功并返回服务商信息	
管理员查询服务商	系统提示操作成功并按页返回服务商信息列表	
管理员审核服务商开户	系统提示操作成功	
管理员审核服务商变更	系统提示操作成功	
管理员暂停服务商	系统提示操作成功	
管理员恢复服务商	系统提示操作成功	
管理员审核服务	系统提示操作成功	
服务商查询服务单信息	系统提示操作成功并按页返回服务单信息列表	
服务商根据 id 查询服务单信息	系统提示操作成功并返回服务单信息	
服务商接受服务单	系统提示操作成功	
服务商接受服务单	系统提示操作成功	
服务商收到寄出商品	系统提示操作成功	

服务商完成服务单	系统提示操作成功	

服务商取消服务单	系统提示操作成功	
服务商定义服务	系统提示操作成功并返回服务 ID	
服务商取消服务	系统提示操作成功	
服务商修改服务范围	系统提示操作成功	
服务商查询服务	系统提示操作成功并按页返回服务信息列表	
服务商根据 id 查询服务	系统提示操作成功并返回服务信息	

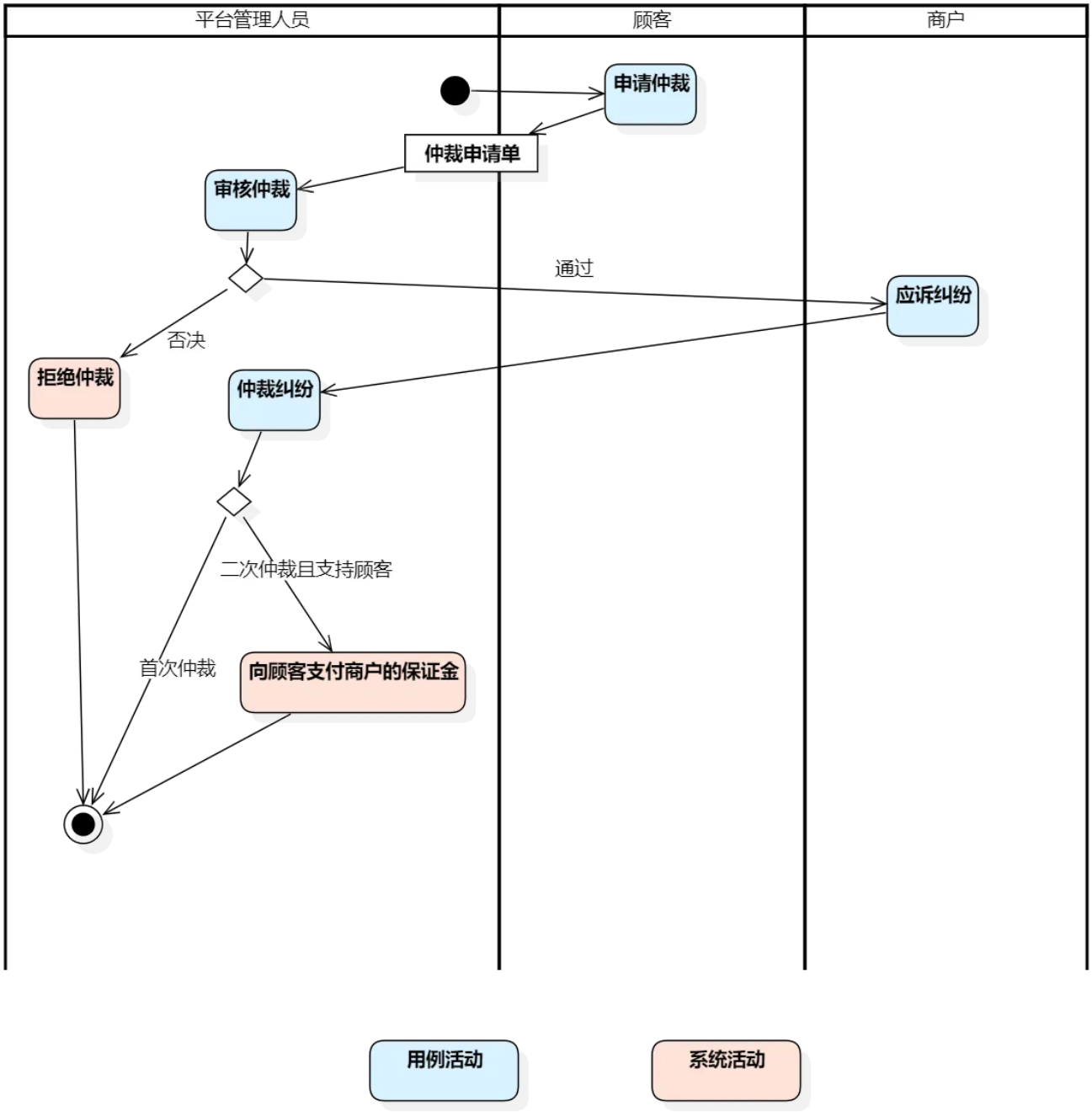
3.5算法

【模块所选用的算法。】

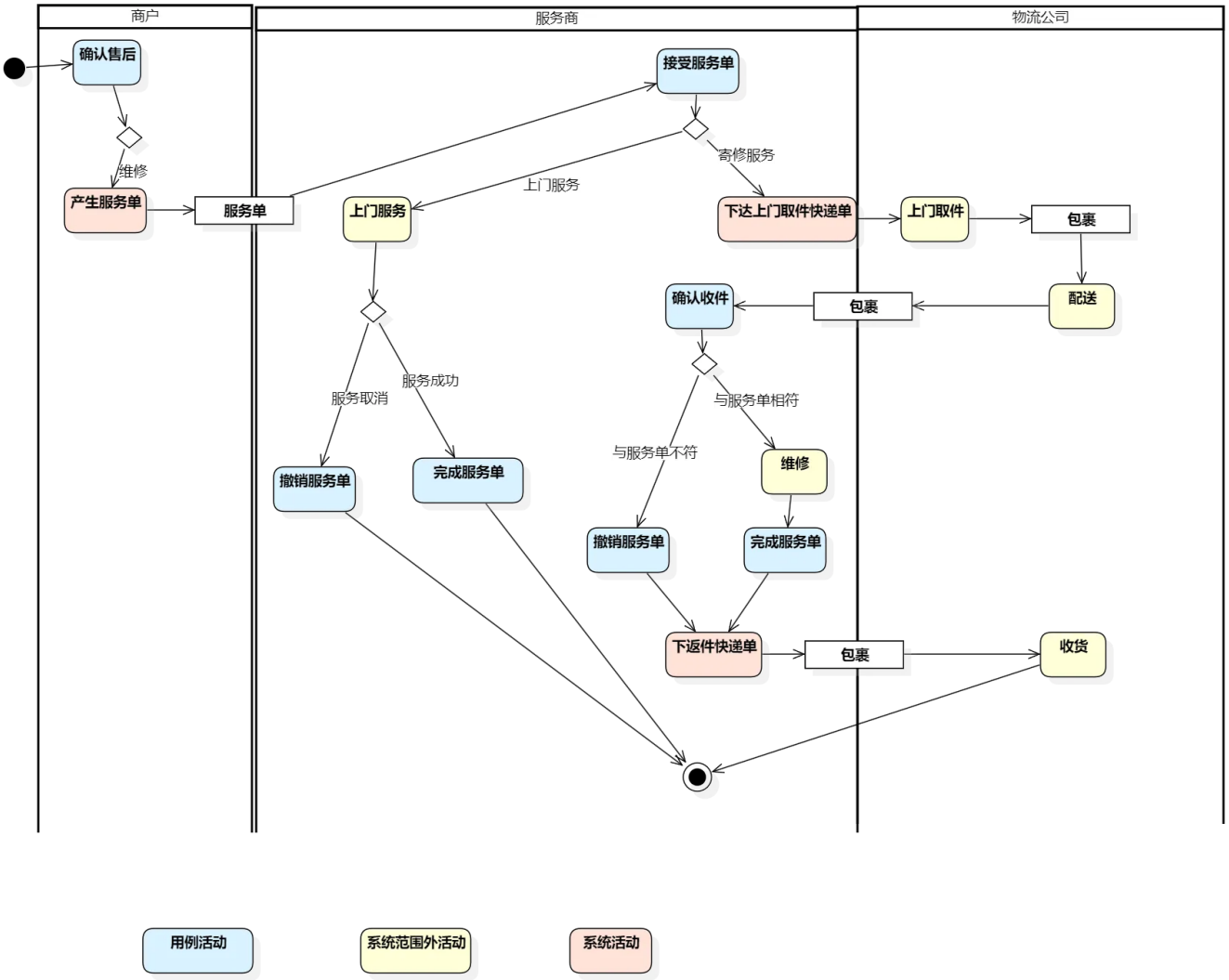
3.6程序逻辑

a. 业务流程图；

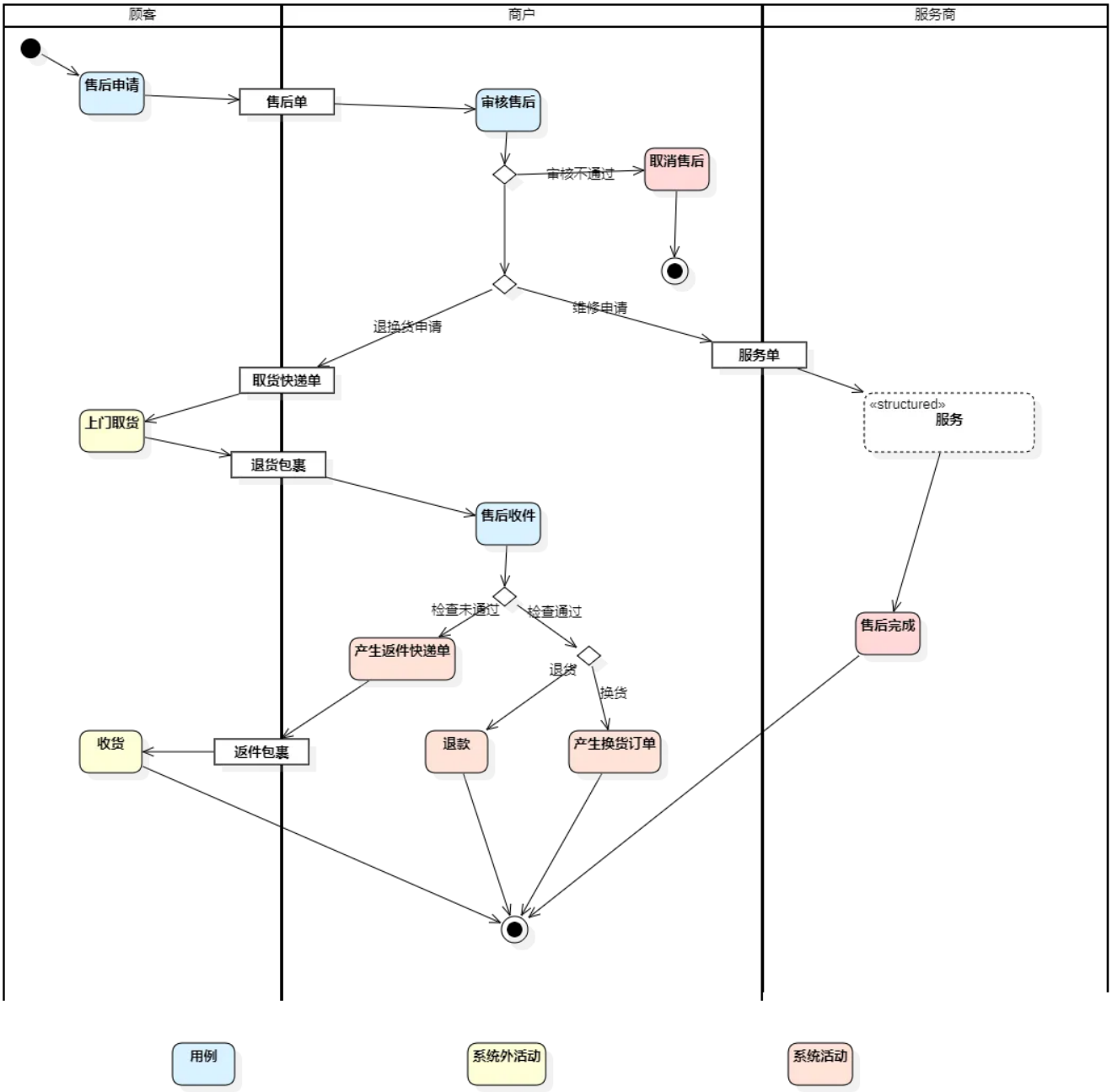
1.仲裁


平台管理人员
顾客
商户
申请仲裁
仲裁申请单
审核仲裁
通过
应诉纠纷
否决
拒绝仲裁
仲裁纠纷
二次仲裁且支持顾客
首次仲裁
向顾客支付商户的保证金
用例活动
系统活动

2.售后

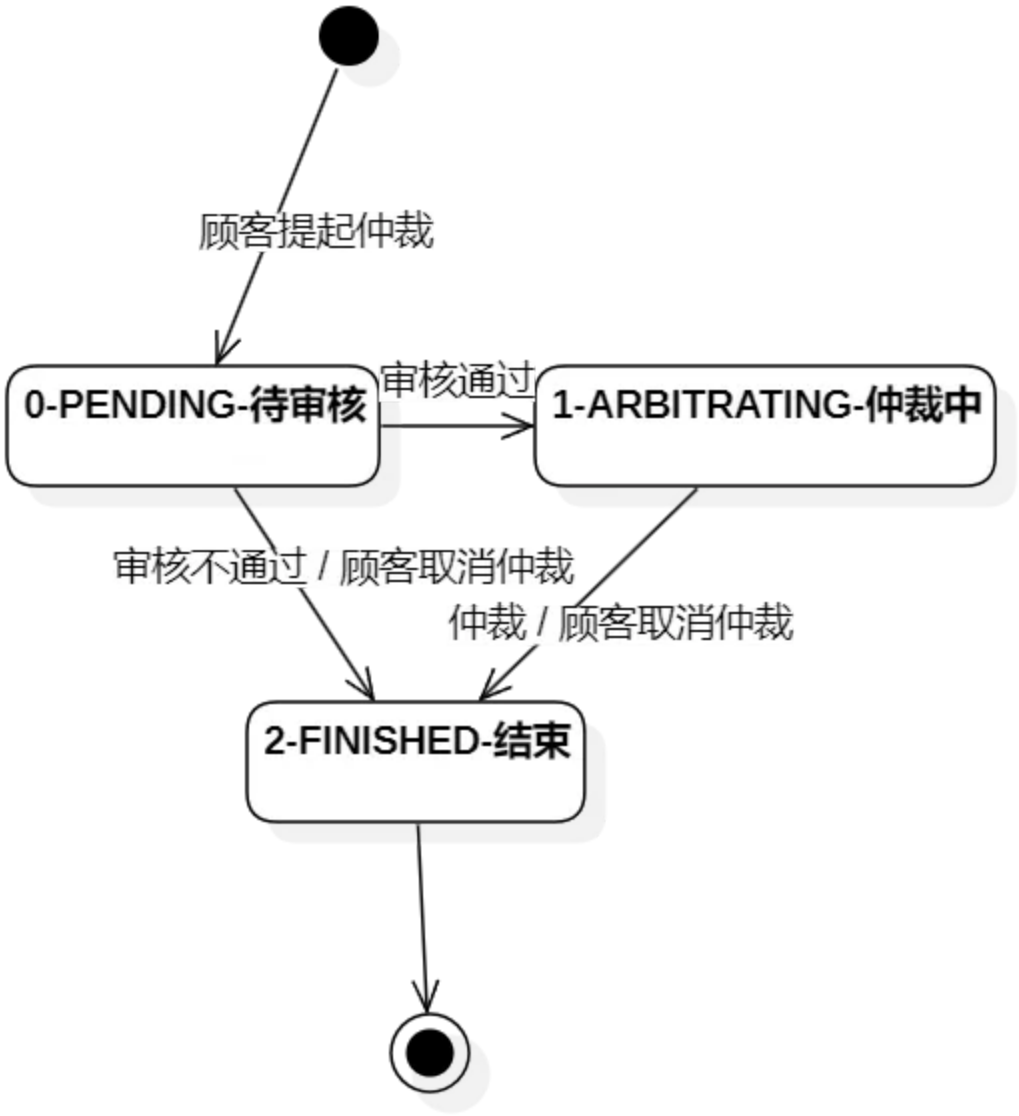

商户
服务商
物流公司
确认售后
维修
产生服务单
服务单
接受服务单
寄修服务
上门服务
上门服务
下达上门取件快递单
上门取件
包裹
配送
包裹
确认收件
服务取消
服务成功
与服务单相符
与服务单不符
维修
撤销服务单
完成服务单
撤销服务单
完成服务单
下返件快递单
包裹
收货
用例活动
系统范围外活动
系统活动

3.服务


顾客
商户
服务商
售后申请
售后单
审核售后
审核不通过
取消售后
退换货申请
维修申请
服务单
取货快递单
«structured»
服务
上门取货
退货包裹
售后收件
检查未通过
检查通过
售后完成
产生返件快递单
退货
换货
收货
返件包裹
退款
产生换货订单
用例
系统外活动
系统活动

b.判定表

1.仲裁单



(仲裁单状态取值：待审核 0，仲裁中 1，结束 2)

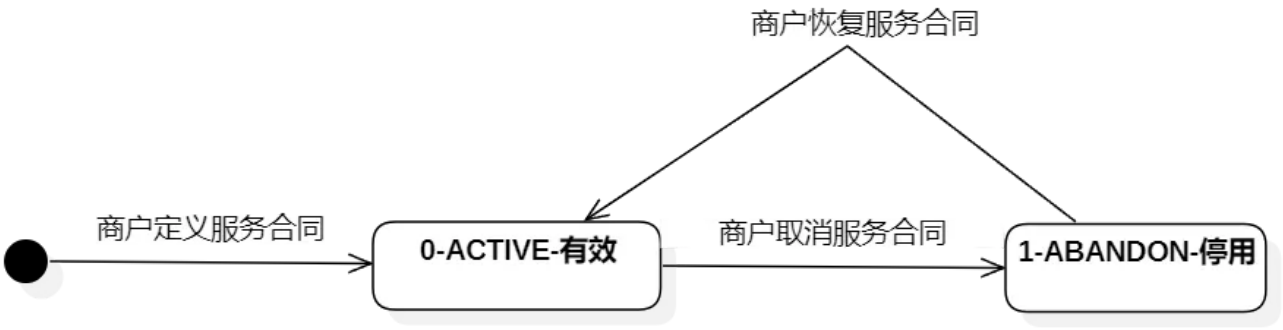
		1	2	3
条件	仲裁单状态	=0	=1	=2
操作	平台管理员审核仲裁单	√		
	顾客取消仲裁单	√	√	
	平台管理员执行仲裁		√	

2.服务单

（服务单状态取值：待接受 0，服务中 1，结束 2）

		1	2
条件	服务单状态	=0	=1
操作	服务商接单	√	
	服务商取消服务		√
	顾客结算服务单		√

3.服务合同

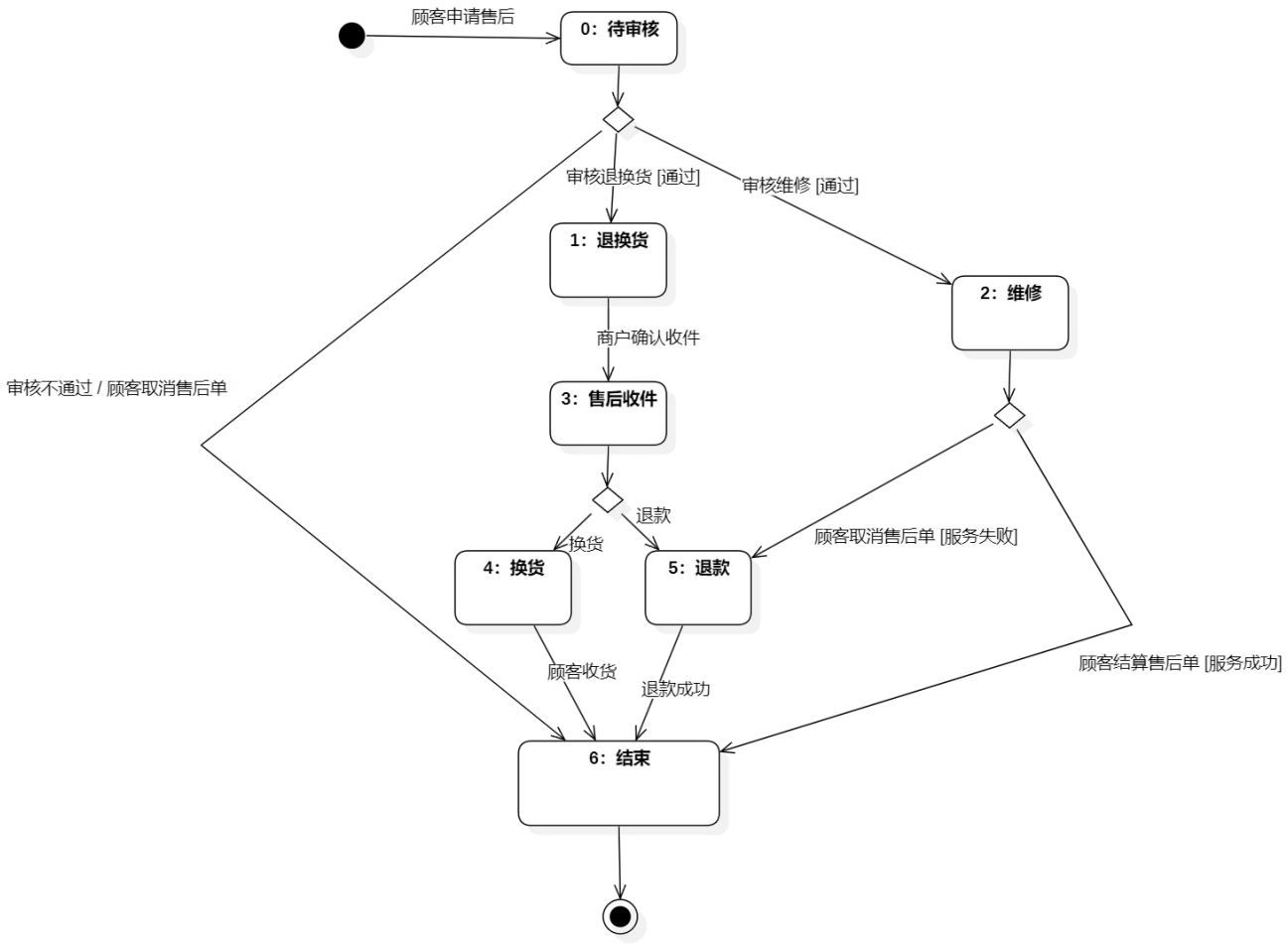


（服务合同状态取值：有效 0，停用 1）

		1	2
条件	服务单状态	=0	=1
操作	商户恢复服务合同	√	

	服务商取消服务合同		√

4.售后



（售后状态取值：待审核 0，退换货 1，维修 2，售后收件 3，换货 4，退款 5）

		1	2	3	4	5
条件	服务单状态	=0	=1	=2	=3	=4
操作	平台管理员审核售后单	√				
	商户验收顾客的退/换货		√			

	顾客修改售后单信息	√				
	顾客对失败的售后单维修退款			√		
	顾客结算售后单			√		
	顾客确认收货				√	

3.7接口

售后模块

功能	类型	API 接口	接口描述
顾客提交售后申请	POST	/orderitems/{id}/aftersales	用户 token 验证当前为顾客身份操作资源， 顾客进入售后申请界面提交售后单，设置售后单状态为未售后
顾客或者商铺根据售后状态分类查询售后列表	GET	/aftersales	默认所有售后单列表，选择按售后单状态作为条件查询。用户 token 验证身份，顾客和商铺只能查询到自己的售后单
顾客查询所有的可申请售后的订单明细	GET	/aftersales/orderitems	顾客只能查询到自己的订单明细，已经申请过退货或者换货的订单明细不会返回

顾客根据售后单 id查询售后单信息	GET	/aftersales/{id}	顾客通过这个 API 只能查询到自己的售后单

顾客修改售后单信息	PUT	/aftersales/{id}	顾客只能在处于申请态的售后单上进行修改，一旦售后服务进入其他状态（处理中、已完成等状态），顾客将无法再对售后单进行修改
顾客取消售后	DELE TE	/aftersales/{id}	顾客只可以在未售后和售后中的状态取消
顾客申请售后仲裁	POST	/aftersales/{id}/arbitrations	只允许顾客提出仲裁申请，需设置售后为仲裁中，如果已经在仲裁中的售后不允许再仲裁，则出现 705 错误
顾客申请二次仲裁	POST	/aftersales/{id}/second_arbitra tions	只允许顾客提出的仲裁申请，且有一次的仲裁记录，需设置售后为仲裁中，如果已经在仲裁中的售后不允许再仲裁，出 705 错误
顾客取消仲裁	DELE TE	/aftersales/{id}/arbitrations	顾客根据仲裁 ID 取消仲裁，顾客只可以在未仲裁和仲裁中状态取消，已仲裁的仲裁单不可取消

顾客、 商铺、管理员查询售后单对应的物流单号	GET	/aftersales/{id}/express	顾客、商铺、管理员通过 token 验证.根据售后 id 查询到对应的物流订单，已经申请过退货或者换货的订单明细不会返回
商铺审核售后	PUT	/shops/{shopid}/aftersales/{id}/confirm	商铺审核顾客提出的售后请求并设置售后状态为售后中或已售后
商铺验收售后商品	PUT	/shops/{shopid}/aftersales/{id }/receive	维修商品不可调用此 API
商铺取消售后	DELE TE	/shops/{shopid}/aftersales/{id }/cancel	商铺取消售后，需要设置相应售后为正常
商铺应诉仲裁	PUT	/shops/{shopid}/arbitrations/{id}/response	顾客提出仲裁，商铺应诉仲裁
管理员受理仲裁单	PUT	/shops/{shopid}/arbitrations/{id}/accept	仅申请中的仲裁单才能被受理
管理员仲裁结案	PUT	/shops/{shopid}/arbitrations/{i d}/close	仅仅受理态才可以结案，负责仲裁的仲裁员才可以结案，设置相应

售后为正常			
商铺或管理员查看所有售后单	GET	/shops/{shopid}/aftersales	商铺或管理员可通过售后单 ID、 售后类型和状态选择查看所有售后单

商铺或管理员查询售后单信息	GET	/shops/{shopid}/aftersales/{id }	商铺或管理员根据售后单 id 查询售后单信息
商铺或管理员查询顾客申请的仲裁单信息	GET	/shops/{shopid}/arbitrations	商铺或管理员查询顾客申请的仲裁单信息（根据仲裁单状态分类查询），按照申请时间从近到远排序
商铺或管理员查询仲裁单详情	GET	/shops/{shopid}/arbitrations/{i d}	商铺或管理员根据仲裁单 ID 查询仲裁单详情信息

服务模块

商户选择服务商提供的地区服务	POST	/shops/{did}/products/{id}/maintainers/{mid}/service/{sid}	商户选择服务商提供的地区服务，与具体商品关联。需要登录， 服务的状态默认为无效， priority默认为 1000
商户修改服务商在某个地区的商品服务	PUT	/shops/{did}/product_services/ {id}	商户修改服务商在某个地区的商品服务(暂停：从有效变为无效；恢复：从无效变为有效)
商户取消服务商在某个地区的商品服务	DELE TE	/shops/{did}/product_services/ {id}	商户取消商品服务；需要登录

商户通过商品和地区查询服务商	GET	/shops/{did}/products/{id}/reg ion/{rid}	商户通过商品和地区找到服务商， 可以用下级地区查找，返回结果按照优先级排序（已关闭的服务商不返回）
商户通过服务商 id查询服务商	GET	/shops/{did}/maintainers/{mid }	商户通过服务商 id找到服务商， 已经关闭的服务商不返回
商户取消服务商的所有服务	DELE TE	/shops/{did}/maintainers/{mid }	商户取消指定 ID 的服务商的所有服务；需要登录

获得服务单的所有状态	GET	/services/states	获得服务单的所有状态
商户创建服务单	POST	/internal/shops/{did}/products/{id}//region/{rid}/serviceOrde rs	商户创建服务单
商户查询服务单信息	GET	/shops/{did}/serviceOrders	商户通过这个 API 只能查询到自己的服务单，平台管理人员可以查询所有的服务单
商户根据服务单 id查询服务单信息	GET	/shops/{did}/serviceOrder/{id }	商户查询指定 ID 的服务单， 商户通过这个 API只能查询到自己的服务单，平台管理人员可以查询所有的服务单
管理员或商家取消服务单	DELE TE	/shops/{did}/serviceOrder/{id }	商家只能取消自己的服务单，管理员可以取消所有的服务单

服务商注册账号	POST	/maintainers	服务商注册账号时无需登录， 注册的账号在审核态
服务商申请变更账号信息	PUT	/maintainers/{mid}	服务商申请变更账号信息， 在除关闭态以外的状态均可以修改
服务商注销账号	DELE TE	/maintainers/{mid}	服务商注销账号，完成所有服务单后才可注销
服务商查看账户信息	GET	/maintainers/{mid}	服务商查看账户信息，需检查 maintainerId 是否与访问者的一致
管理员查询服务商	GET	/adminusers/{aid}/maintainers	管理员查询服务商，会返回所有符合状态的服务商
管理员审核服务商开户	PUT	/adminusers/{aid}/maintainers/{mid}/account	平台管理员修改服务商状态为开户状态的服务商
管理员审核服务商变更	PUT	/adminusers/{aid}/maintainers/{mid}/modify	平台管理员修改服务商状态为变更状态的服务商
管理员暂停服务商	PUT	/adminusers/{aid}/maintainers/{mid}/suspend	平台管理员暂停服务商，修改服务商状态为暂停
管理员恢复服务商	PUT	/adminusers/{aid}/maintainers/{mid}/resume	平台管理员恢复服务商，修改服务商状态为恢复服务

管理员审核服务	PUT	/adminusers/{aid}/services/{sid}/service	平台管理员审核服务
服务商查询服务单信息	GET	/maintainers/{mid}/serviceOrd ers	服务商通过这个 API 查询关联商家派发的服务单和已完成的服务单
服务商根据 id 查询服务单信息	GET	/maintainers/{mid}/serviceOrd ers/{id}	服务商查询指定 ID 的服务单信息， 通过这个 API查询关联商家派发的服务单和已完成的服务单
服务商接受服务单	PUT	/maintainers/{mid}/serviceOrders/{id}/accept	服务商接受服务单，当类型为寄件时，需要产生一个上门取件的物流单与之关联
服务商拒绝服务单	PUT	/maintainers/{mid}/serviceOrders/{id}/refuse	服务商拒绝服务单后，重新派单， 派给下一优先级的服务商
服务商收到寄出商品	PUT	/maintainers/{mid}/serviceOrders/{id}/receive	服务商验收一个商品，并反馈验收状态
服务商完成服务单	PUT	/maintainers/{mid}/serviceOrder/{id}/finish	服务商完成一个服务单，并反馈服务状态，当类型为寄件时，需要产生一个寄件的物流单与之关联
服务商取消服务单	PUT	/maintainers/{mid}/serviceOrders/{id}/cancelOrder	服务商取消一个服务单

服务商定义在某个或多个地区为某种商品的服务*	POST	/maintainers/{mid}/categories/{id}/service	服务商定义在某个或多个地区为某类商品的服务，需要登录；服务默认为无效，priority 默认为 1000
服务商取消服务	PUT	/maintainers/{mid}/services/{id}/cancel	服务商取消某一服务
服务商修改服务范围	PUT	/maintainers/{did}/services/{id}/changeRegion	服务商修改某服务的服务范围
服务商查询服务	GET	/maintainers/{mid}/services	服务商通过这个 API 查询自己注册的服务
服务商根据 id 查询服务	GET	/maintainers/{mid}/service/{id }	服务商根据服务 id，通过这个 API 查询

仲裁模块

仲裁申请	POST	/arbitrations/aftersales/{aftersalesId}/arbitrations	顾客选择并填写仲裁理由，提交申请，若重复申请则系统提示已提交
取消仲裁	DELETE	/arbitrations/shops/{shopId}/arbitrations/{id}	顾客选择需要取消的中菜单并取消
审核仲裁	PUT	/arbitrations/shops/{shopId}/arbitrations/{id}/accept	管理员查看仲裁单，审核相应仲裁申请，选择是否通过，若否决则需填写原因

仲裁纠纷	PUT	/arbitrations/shops/{shopId}/arbitrations/{id}/close	管理员查看纠纷详情，选择支持的对象
查询仲裁	GET	/arbitrations/shops/{shopId}/arbitrations	系统展示仲裁申请列表，管理员选择相应仲裁查看详情

3.8存储分配

3.8.1 物理结构设计

1. 定义数据库的物理结构：

数据库的物理结构主要包括存取方法和存储结构的设计。它们共同决定了数据库如何将数据存储在硬件介质中，并如何高效地检索和更新这些数据。

存取方法：

存取方法指的是数据库如何组织数据以及如何快速访问这些数据。在关系数据库中，常见的存取方法包括：

索引：索引是提高查询效率的常见技术。设计时需要选择合适的索引类型，如B+树索引、哈希索引等。索引的设计要考虑查询模式、数据的分布、插入/更新的频率等因素。例如，如果查询操作更多地依赖某些字段（如主键、外键等），就应该为这些字段建立索引。

数据访问路径：这包括顺序扫描、索引扫描、哈希扫描等。存取方法的选择要确保对不同查询类型（如全表扫描和范围查询）的高效支持。

查询优化：根据查询模式和数据访问特征，数据库可以采用不同的存取方法。查询优化器在设计过程中需要考虑多个访问路径并选择最优路径。

存储结构：

存储结构涉及如何将数据物理存储在磁盘上，并如何组织这些数据以提高I/O性能和空间利用率。常见的存储结构包括：

表的物理组织：在关系数据库中，表的存储通常是通过行存储（Row Store）或列存储（Column Store）来进行。对于频繁查询某些列的应用，列存储结构通常更为高效。而对于需要更新和插入数据频繁的应用，行存储结构更为常见。

数据分区和分布：大规模数据库常常会通过分区（Partitioning）技术将数据分布到多个物理存储设备上。分区可以基于不同的策略进行，比如范围分区、哈希分区或列表分区。通过分区，查询可以局部化，从而减少磁盘I/O操作的开销。

数据冗余和备份：考虑到容错性，数据可能会采取冗余存储方式，比如RAID（冗余磁盘阵列）技术或者通过分布式数据库实现数据的高可用性。

2. 物理结构的评价

在数据库物理结构设计的第二步，进行有效的评价是至关重要的。主要从时间效率和空间效率两个角度进行评估，确保数据库系统的高效运行。

时间效率：

时间效率是指数据库在处理查询、插入、更新等操作时的速度和响应时间。为了提高时间效率，通常需要考虑以下方面：

查询响应时间：选择合适的索引和查询优化策略，使得数据库能在最短的时间内找到所需数据。设计时要特别关注慢查询，采用合适的优化技巧，如预计算结果、物化视图等。

写入效率：更新、插入和删除操作的速度也非常关键。设计时需要考虑数据的写入模式，是否有高并发插入或更新需求。可以采用缓存技术、批量操作或事务合并等方法来提高写入效率。

并发控制：确保多个并发操作不会导致数据不一致，并尽量减少锁的竞争。设计合适的事务隔离级别，采用乐观锁或悲观锁技术。

空间效率：

空间效率关心的是数据库存储的利用率和成本。主要考虑以下几个方面：

数据压缩：数据压缩技术可以显著减少存储空间，尤其是在存储大量重复或冗余数据时。例如，列式存储通常支持数据压缩，可以有效减少数据存储空间。

冗余数据：需要谨慎处理冗余数据，避免不必要的存储浪费。数据库设计时应确保没有重复数据，并采用规范化来消除冗余。

存储设备选择：根据应用的特点选择合适的存储介质（如SSD和HDD）。SSD提供更高的读写速度，而HDD则在存储大数据时更具成本效益。

数据生命周期管理：数据的存储策略应根据数据的生命周期进行设计。例如，冷数据（不常访问的数据）可以采用低成本的存储方式，而热数据（频繁访问的数据）则需要高性能的存储设备。

3. 优化与持续评估

数据库的物理结构设计并不是一劳永逸的。随着数据量的增加、查询模式的变化以及硬件资源的变化，物理结构的优化和调整是持续的过程。例如：

定期审查索引的有效性：随着应用的不断变化，某些索引可能不再高效，定期检查并调整索引设计可以提升查询性能。

数据重组与压缩：随着数据的增加，表和索引可能会变得碎片化，影响查询性能。定期进行数据重组或压缩操作可以帮助维持良好的存储和访问效率。

硬件资源的调整：随着业务增长，可能需要对硬件资源进行扩展，如增加存储设备、升级处理器或扩展内存等。

总结：数据库的物理结构设计是一个系统的工程，涉及存取方法、存储结构的选择、时间与空间效率的评估等多个方面。设计时需要综合考虑硬件资源、数据库的工作负载、查询和写入模式以及未来的扩展需求。通过优化这些物理设计，数据库能够在保证高效性能的同时，最大限度地利用存储资源，确保系统的稳定性和可维护性。

3.8.2 存取方法选择

3.8.2.1 B+树索引存取方法的选择

1. 为每个表的主键添加 B+树索引

主键是每个表的唯一标识符，它保证了数据的唯一性和完整性。由于主键通常会作为查询条件，特别是在需要快速检索特定行时，通过为主键添加 B+树索引可以显著提高查询效率。

2. 为售后表的售后类型字段添加 B+树索引

售后表的售后类型字段可能用于各种查询操作，尤其是在需要按售后类型进行过滤和范围查询时（如查询某一类型的所有售后记录）。通过为该字段添加 B+树索引，可以显著提升查询性能。

3. 为服务单表的服务状态字段添加 B+树索引

服务单表中的服务状态字段经常用于查询和过滤操作，如查询某些特定状态（如“未完成”、“待处理”等）的服务单。为服务状态字段添加 B+树索引将大大提升这些查询的效率，尤其是需要频繁筛选和排序服务单状态时。

4. 综合考虑索引的设计

在实际设计中，需要综合考虑各个索引的影响，以下几点是优化设计时需要注意的：

索引创建的开销：每个索引都会占用额外的存储空间，并且会在插入、更新和删除操作时产生额外的开销。因此，除了主键索引外，其他索引应当根据实际的查询需求进行设计，避免不必要的索引。

查询模式分析：在设计索引时，应分析查询模式，特别是常见的查询类型和条件。例如，如果某字段经常作为查询条件，则可以考虑为其添加索引。如果字段较少参与查询或更新操作，可能就不需要创建索引。

索引的维护成本：索引的维护会随着数据量的增加而变得更为复杂，特别是在写操作频繁的情况下，插入和更新操作的性能可能会受到影响。因此，设计索引时需要权衡读写负载。

3.8.2.2 Hash 索引存取方法的选择

为商品明细表的串号添加 Hash 索引：串号属性在查询中频繁使用，为了提高查询性能，采用 Hash 索引是一种有效的选择。Hash 索引能够高效地支持等值查询，特别适合用于像串号这样的经常作为检索条件的属性。通过 Hash 索引，可以显著加快基于串号的查询速度，尤其是在处理大规模数据时。

3.8.2.3 聚簇存取方法的选择

为数据表中涉及的相关状态和类型属性添加聚簇索引：聚簇索引通过将相同值的元组存储在物理上相邻的区域，能够优化基于这些属性的查询性能。对于状态和类型等经常进行分类展示和处理的属性，聚簇索引能够有效地将相同状态或类型的数据紧密地存放在一起，从而减少查询时的数据扫描范围，显著提高查询效率。此外，聚簇索引还可减少磁盘I/O操作，进一步提升数据访问速度。

3.8.3 存放位置和存储结构

3.8.3.1 确定数据的存放位置

为了提升系统性能，我们需要根据应用需求，将数据分为易变部分、稳定部分、经常访问部分和低频访问部分，并进行分层存储。关系型数据可以使用 MySQL 存储，而对于非关系型数据或关系复杂、难以

用传统关系数据库存储的对象，可以使用 MongoDB 来存储。

3.8.3.2 确定数据的存储结构

在确定数据存储结构时，我们需要综合考虑数据的访问模式和使用需求，选择最合适的存储结构。常见的存储结构包括顺序存储、链式存储、哈希、树和图等，每种结构有其特定的应用场景和优势。以下是对这些存储结构的详细分析：

顺序存储：

顺序存储适用于数据是有序的情况，能够提供高效的顺序遍历和检索。它特别适合用于对数据进行顺序分析或按特定顺序检索的场景。例如，排序后的数据列表或日志文件存储。顺序存储结构对于顺序操作非常高效，但在插入和删除操作时可能较为低效，尤其是需要频繁更新的数据集。

链式存储：

链式存储采用链表结构，适用于频繁插入和删除的场景。链表通过指针连接数据元素，使得插入和删除操作能够在常数时间内完成。然而，链表的随机访问性能较差，特别是在需要按索引快速查找数据时，效率较低。因此，链式存储适合动态数据集，但不适用于需要频繁查找的场景。

哈希存储：

哈希存储适用于等值查询的场景，通过哈希函数将关键字映射到存储位置，从而实现快速的查找操作。哈希存储结构对于精确查找非常高效，但在处理冲突时（即多个元素映射到同一位置）需要采用冲突解决策略，如开放地址法或链式解决。此外，哈希表通常不适合范围查询或有序遍历，因此在需要这些操作的场景中，哈希表并不理想。

树形存储：

树形存储结构，特别是 B+ 树，适用于范围查询和有序数据的存储。B+树是一种自平衡的树形结构，支持高效的插入、删除和范围查询操作。由于其平衡性，B+树能够在保证查询效率的同时支持有序遍历，因此广泛应用于关系数据库系统中，如索引结构。此外，树形结构在多层次数据组织（如文件系统和目录结构）中也非常有效。

图形存储：

图形存储结构适用于表示复杂的关系，特别是在社交网络、推荐系统和网络拓扑等应用中。图数据库通过节点和边的关系来表达实体之间的关联，能够高效地处理复杂的查询，如路径查找、社交网络分析等。图结构在表示一对多或多对多关系时具有很大的优势，适合用来处理高度互联的数据。

选择存储结构时的考量因素：

数据访问模式：不同的存储结构适用于不同类型的查询和操作。如果数据主要是顺序访问，顺序存储会非常高效；如果需要频繁插入和删除，链式存储则是更好的选择；如果应用场景主要是精确查找，哈希存储可能最为高效。

数据的复杂性：如果数据具有复杂的关系或层级结构（如社交网络或组织架构），图结构或树形结构将更合适；而对于平坦的数据或结构简单的数据集，顺序存储或哈希表更为有效。

查询类型：如果查询需求主要是范围查询或排序，B+树等树形结构更为合适；如果是需要快速的精确匹配，哈希表则是理想选择。

性能需求：不同存储结构对性能的影响不同。例如，哈希表在查找操作上具有较高的时间效率，但在进行范围查询时，树形结构则更优。

3.8.4评价物理结构

存储空间利用率：采用分布式数据库（如 MongoDB），以提升存储空间的灵活性和利用率。分布式数据库通过将数据分布在多个节点上，能够有效减轻单一节点的存储压力，并优化整体存储资源的利用，从而提高系统的存储效率。

存取时间：单一数据库服务器在高峰期可能会出现性能瓶颈，影响系统响应速度。为了解决这个问题，可以引入 Redis 作为缓存层，结合数据分片和负载均衡技术，缓解数据库服务器的压力。Redis可以将热数据存储在内存中，快速响应用户请求，减少数据库查询负担，同时通过分片和负载均衡分散访问压力，进一步提高存取效率。

维护代价：为降低维护成本，可以选择开源数据库 Redis，并利用华为 CodeArts 优惠券。Redis 作为高性能的开源缓存数据库，能提供快速的数据访问，降低硬件和运维成本。而华为 CodeArts 优惠券则能够进一步降低相关云服务的费用，从而在整体维护成本上获得显著的经济优势。

3.9限制条件

1. 服务器必须正常启动，且各服务之间能够正常访问

服务器状态监控：确保服务器启动正常并持续运行，可以通过定期的健康检查和状态监控工具（如 Prometheus, Zabbix）来监控服务器的健康状况。

服务间通信：各服务之间需要能够无障碍地访问对方。这要求服务架构设计合理，网络连接稳定，且确保负载均衡和容错机制的有效性。

故障恢复：在服务出现故障时，能够迅速定位并恢复服务。可能需要配置自动重启策略，或者使用容器编排工具（如 Kubernetes）来实现高可用和自动恢复。

2. 网络状况必须良好

带宽和延迟：保证网络带宽满足系统的高并发需求，并尽量减少网络延迟。在高负载情况下，网络瓶颈可能会影响性能。

负载均衡：通过设置负载均衡器，确保流量在多个服务器之间均匀分配，减少某一节点的压力，避免单点故障。

网络监控与优化：实时监控网络状况，确保无大规模的网络丢包或延迟，必要时对网络进行优化，例如通过CDN加速，或者选择更稳定的网络链路。

3. 对大部分 API，需要用户登录并进行身份检查

身份认证：实现基于 Token（如 JWT）或基于 OAuth 2.0 的认证机制，确保只有经过授权的用户才能访问受限资源。

权限管理：除了身份验证，还需要设计权限控制，确保不同角色的用户只能访问其有权限操作的 API。

加密传输：为确保敏感数据的安全传输，所有 API 请求和响应应通过 HTTPS 加密，防止数据泄露。

API Rate Limiting：为了防止恶意攻击，限制每个用户或每个 IP 地址的访问频率，有效防止 DDoS 等攻击。

3.10测试要点

1. 测试要求：

（1）要求所有 API 返回的数据正确

（2）要求 API 能够正确的处理异常

（3）要求 API 能够较快的响应请求

2. 测试用例：等价类划分法：

有效输入的等价类：

正常值：API 接受合法的参数，进行正常流程测试。

边界值：针对输入的边界进行测试，比如最小值、最大值、临界值等。

无效输入的等价类：

空值：传递空值、空字符串等。

非法字符：传递特殊字符或非预期格式的数据。

超出范围：测试输入值超出合理范围的情况（如数值过大、日期无效等）。

异常输入：

数据类型不匹配：如将字符串传递给期望是整数的字段。

错误的请求格式：测试未遵循 API 文档的请求格式（如不正确的 HTTP 方法、缺少必填字段等）。

3. 测试方法：

白盒测试方法：

白盒测试侧重于了解程序内部实现，并基于程序代码进行测试。

基本路径测试方法：

基本路径测试是一种通过识别程序中的基本控制流路径并设计测试用例来验证每条路径的正确性的方法。