**咚宝商城-购物车-模块开发流程文档**

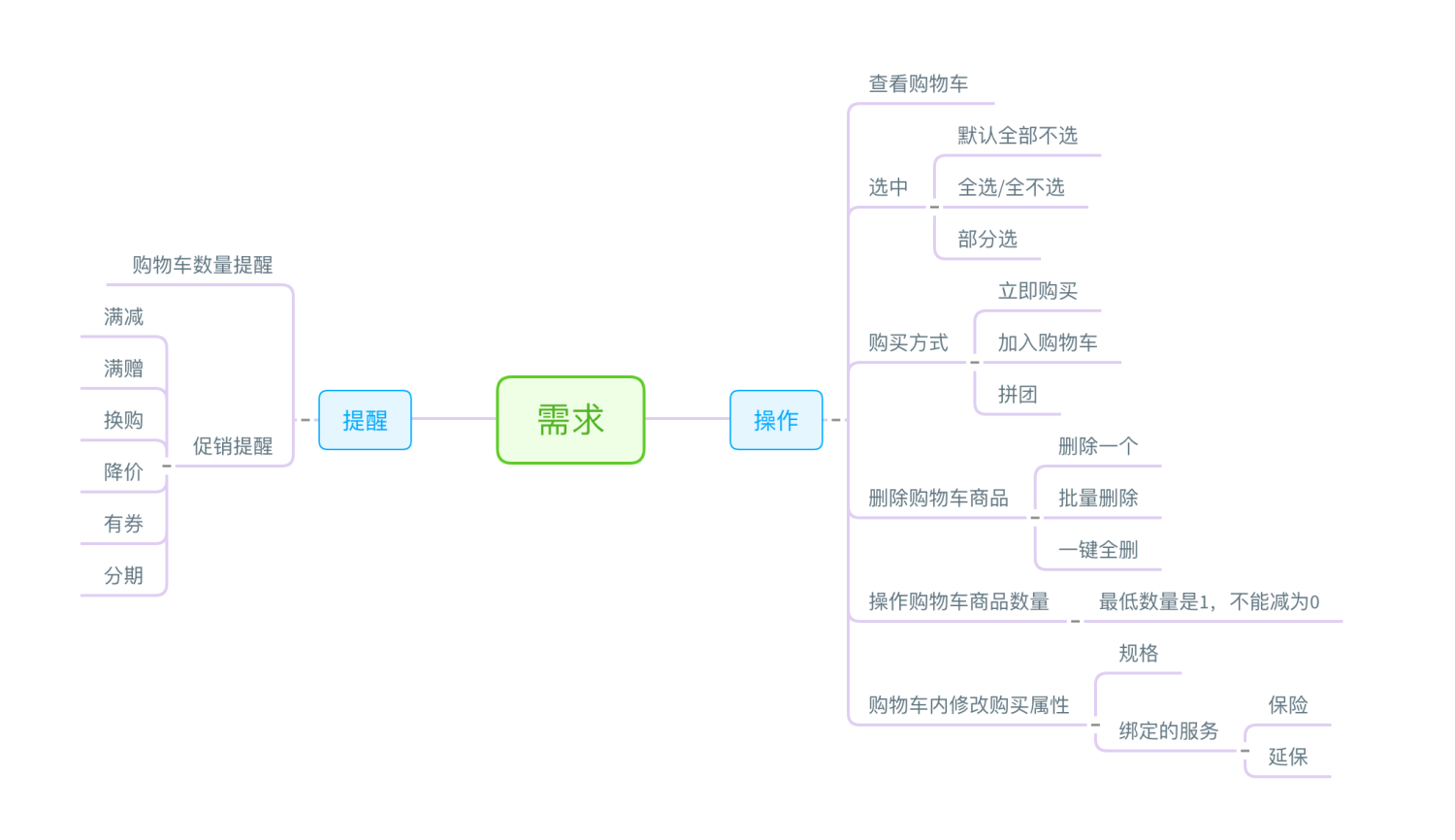
**1. 开发模块描述**

对于一个电商来讲，购物车是整个购买流程最重要的一步。因为电商发展到今天购物车不仅仅只是为了完成打包下单的功能；也是收藏、对比、促销提醒、相关推荐的重要展示窗口。如此多的能力我们该如何设计保证购物车的高性能、以及良好的扩展能力来满足未来的发展呢？

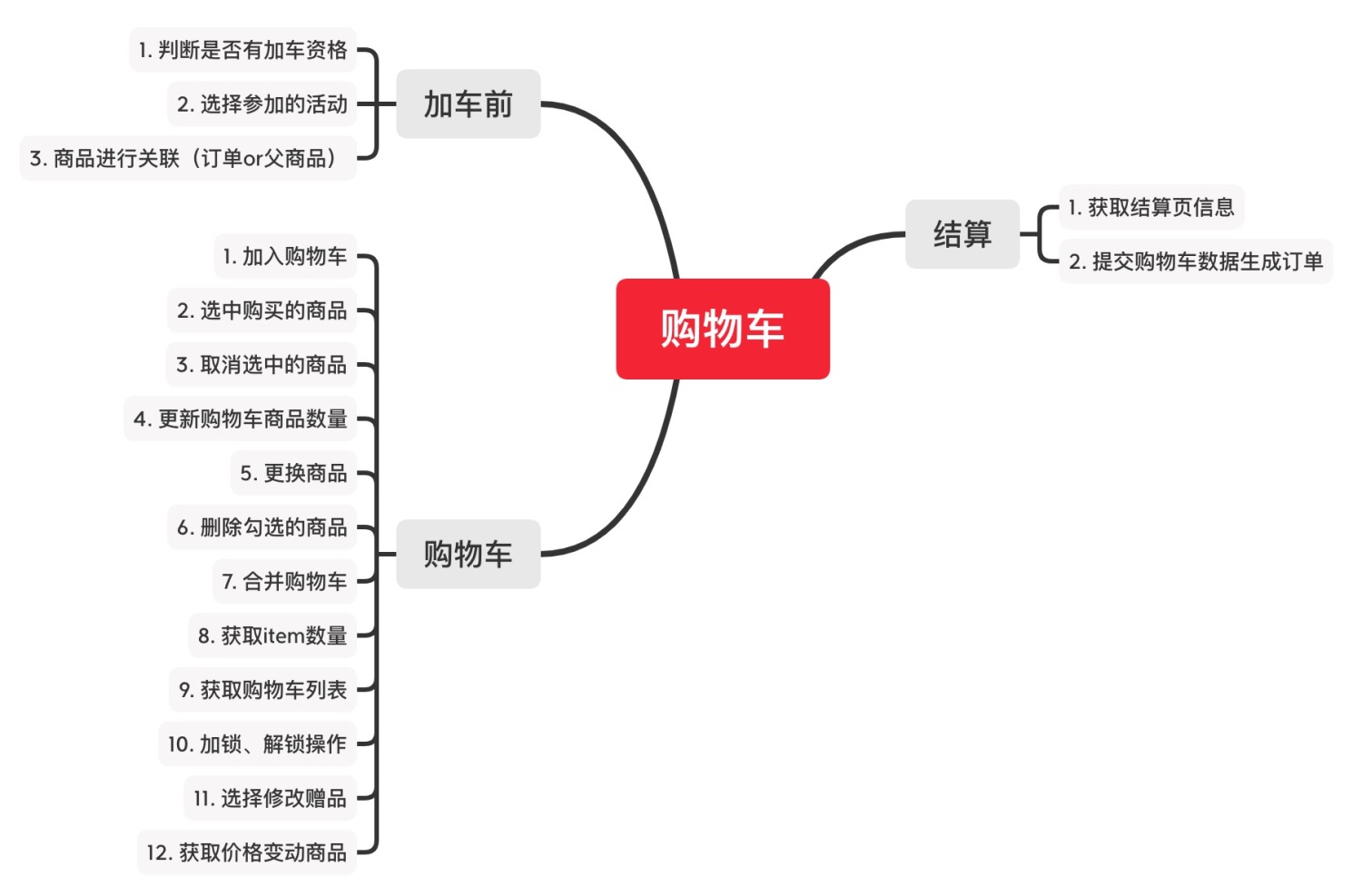
今天开始我们就以一个假定的场景来输出一个购物车设计：某某电商平台，是一个多租户模式（我们前面的诸多设计都是多租户模式），用户可以把商品加入到购物车，并切按照商户纬度来展示、排序。当然购物车也支持常规的各种操作：选择、删除、清空、商品失效等。并且有相关的促销能够提醒用户。同时为了监控、运营，要支撑购物车数据同步到监控、数仓等能力。

1. 未登录，添加购物车
2. 登录后将cookie中的购物车数据合并到当前登录用户的购物车中
3. 购物车数据的维护
4. 对接下单流程

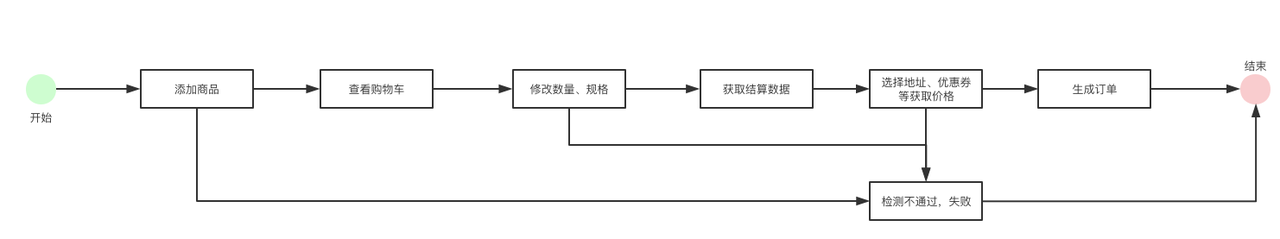
**用户维度**



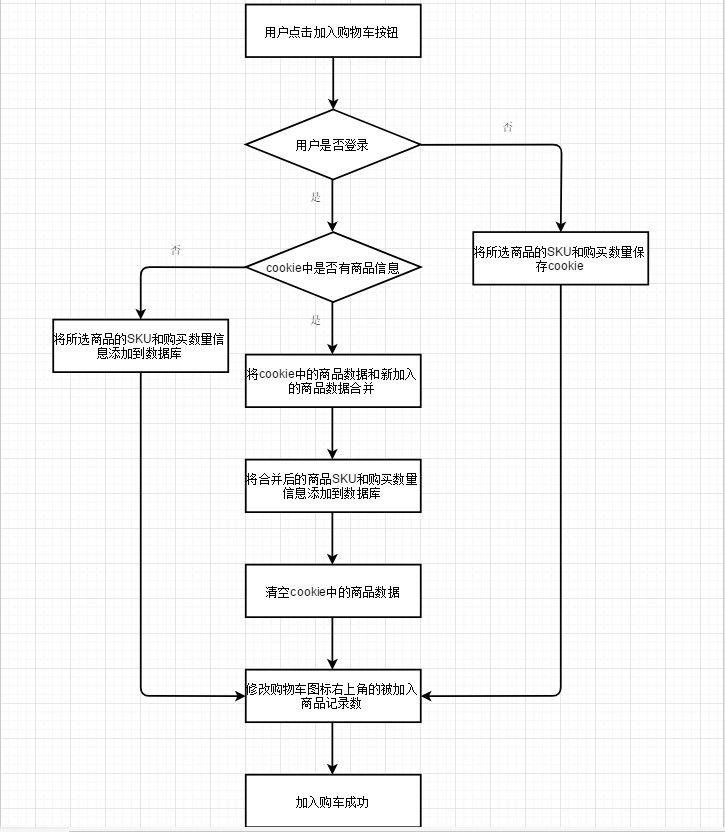
**研发维度**



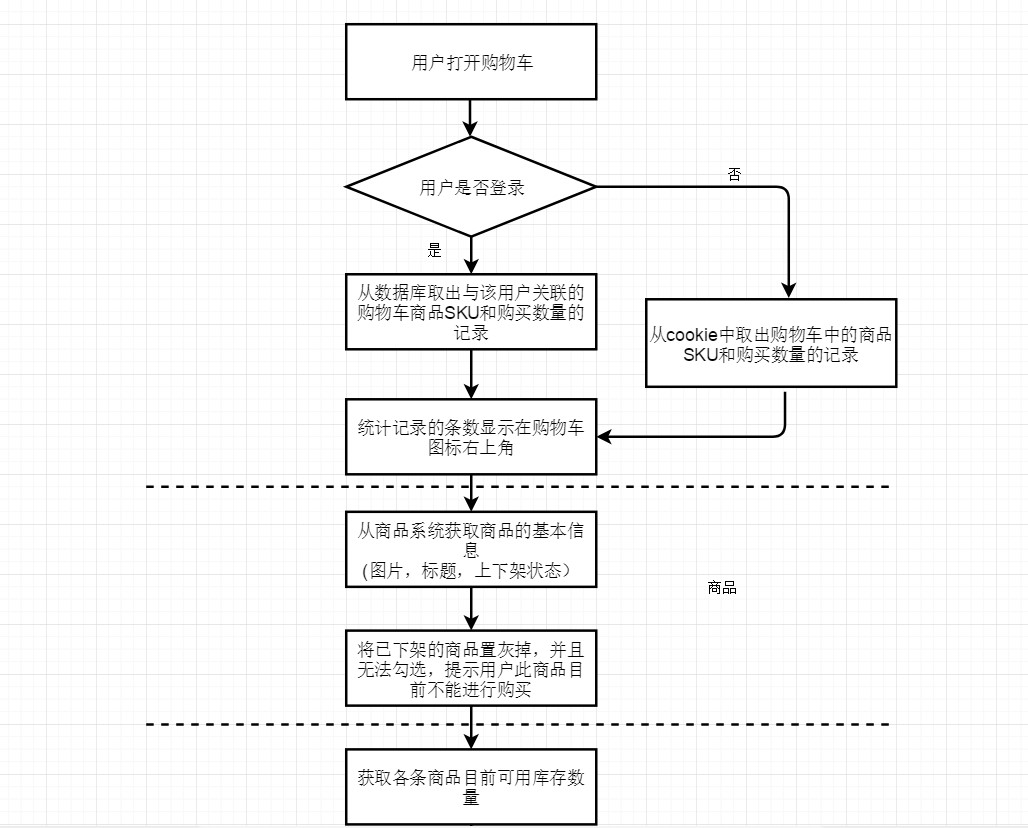
**2. 业务时序流程图**



**添加购物车流程**



**用户打开购物车流程**



**3. 业务实现细节考虑点**

**1. 操作**

我们从用户的角度来看，购物车对于用户来说可以添加商品到购物车（加购物车、立即购买都属于一种添加方式）；加入进购物车后，不想要了可以删除该商品（删一个、删多个、清空）；想多买可以修改购买数量，发现钱不够可以减少购买数量；或者发现红色的比白色更漂亮，可以在购物车方便的进行更换规格；对于一些价格很贵的商品，能够在购物车添加一些保障服务（其实是绑定的虚拟商品）；在要去结算的时候，还会提供选择能力让用户决定哪些商品真的本次要购买。

通过上面的描述我们可以看到这个过程是有其内在联系的。这里说一下关于选中功能，业界有两种做法，各有优劣，我们来看一下。淘宝的产品选中状态是保存在客户端的，并且默认不选中，刷新、重新打开APP状态会消失；京东、苏宁这一类是保存在服务端，会记录用户选中状态。针对这两种情况各有优劣

**客户端：**

1. 性能，选中/不选中的逻辑直接放在本地做，减少网络请求
2. 体验，多端不能同步，但是购物车相对来说更像是一个收藏夹，每次用户自己选择也无可厚非
3. 计算，价格计算时需要上传本地选中商品（也可以本地计算）
4. 实现，主要靠客户端实现，与服务端无关，研发解耦合

**服务端：**

1. 性能，每次操作选中都需要调用服务端，而该操作可能很频繁，除了网络损耗，服务端也需要考虑该如何快速找到修改的商品
2. 体验，多端同步状态，记录历史状态
3. 计算，服务端可获取数据，请求时无须上传额外数据
4. 实现，服务端与客户端需要商定如何交互，以及返回数据（每次选中会导致价格变化），耦合在一起

**2.购物车内修改购买属性**

**3.促销提醒 - 暂时未做实现**

购物车内的数据在用户过一段时间后，再次查看可以看到促销信息，这样有助于成单

**4.  技术解决方案**

**1.存储**

**基于Redis存储购物车数据**

**基于Mongo存储购物车数据**

**2.商品销售类型发生变化**

这是什么意思呢？大家想一下：比如我把A商品加入到购物车，但是一直没有结算。这时运营说针对A商品搞一个活动，拿出10个库存5折购。那么问题来了，对于之前购物车中就有该商品的用户该如何处理？**这里解决的主要问题是：购物车有该商品的用户不能直接以5折买**。几种方案，我们来看一下：

**方案一：** 促销配置后，所有购物车中有该商品的用户失效或删除，这个方案首先被pass，操作成本太高，并且用户体验差

**方案二：** 购物车中要区分同一个SKU，不同销售类型。也就是说在我们的购物车中不是按照SKU的纬度来加商品，而是通过 **SKU+售卖类型** 来生成一个唯一识别码。

可以看到 **方案二** 解决了同一个sku在购物车并存的问题，并且库存之前互相不影响。不过这里又有一个问题？商品的售卖类型（或者说这个标记），该怎么什么地方设置？好像商品系统可以设计、促销系统也可以设置。我们的逻辑中会在促销系统中进行配置。因为商品属于基础逻辑，如果一改就是全局库存受到影响。活动结束后很难做到自动正常售卖。因此这个标记应该落到活动中进行设置（活动设置时会通过促销系统获取该商品之前的活动是否互斥，以确保配置的活动不会互相矛盾）

**3.失效与排序**

失效： 商品该如何失效，比如：库存没有了、下架了；

这很像一个软删除操作，一旦设置，用户侧看到的商品将是无法进行结算的，只能进行删除操作。

排序我们会采用的设计是：根据某个店铺在购物车中最后发生操作的时间，最新的操作肯定在最上面

**5. 数据结构设计**

基于Redis的存储的数据结构设计

@NoArgsConstructor

@Data

@Document(value = "shopping\_cart")

public class ShoppingCart implements Serializable {

private static final long serialVersionUID = 3077125665949233592L;

@MongoId(targetType = FieldType.STRING)

private String shoopingCartId;

/\*\*购物车商品集合,默认初始化为空集合,不需要再次初始化 \*/

private List<CartItem> items;

/\*\* 更新时间 \*/

private Long updateTime;

/\*\* 预留字段,版本 \*/

private Integer version;

}

@Data

@ApiModel(value="购物车商品元素", description="购物车商品元素")

public class CartItem implements Serializable {

private static final long serialVersionUID = 4110741463154569712L;

/\*\* 购物车中某个商品的唯一标记 \*/

@ApiModelProperty(value = "购物车中某个商品的唯一标记")

private String itemId;

@ApiModelProperty(value = "商品 SKU ID")

private Long skuId;

/\*\* 商品 SKU 编号 \*/

@ApiModelProperty(value = "商品 SKU 编号")

private String skuNo;

/\*\* 商品SPU编号 \*/

@ApiModelProperty(value = "商品SPU编号")

private String spuNo;

/\*\* 商品SKU的规格集合，json \*/

@ApiModelProperty(value = "商品SKU的规格集合")

private String skuSpec;

/\*\* 商品SKU数量 \*/

@ApiModelProperty(value = "商品SKU数量")

private Integer amount;

/\*\* 商品SKU状态:0-有库存，上架状态;1-无库存，下架状态\*/

@ApiModelProperty(value = "商品SKU状态")

private Byte status;

/\*\* 记录加车时候的销售价格 \*/

@ApiModelProperty(value = "记录加车时候的销售价格")

private BigDecimal salePrice;

/\*\* 指定价格加购物车 \*/

@ApiModelProperty(value = "指定价格加购物车")

private BigDecimal specialPrice;

/\*\* 是否免邮,0-否，1-是 \*/

@ApiModelProperty(value = "是否免邮")

private Byte postFree;

@ApiModelProperty(value = "默认商品照片")

private String image;

@ApiModelProperty(value = "商品标题")

private String title;

@ApiModelProperty(value = "对应商品Id")

private Long relProductId;

@ApiModelProperty(value = "对应商品的三级分类ID")

private Long categoryId;

/\*\* 添加时间 \*/

private Long addTime;

/\*\* 更新时间 \*/

private Long updateTime;

/\*\* 绑定的父item id \*/

@Deprecated

private String parentItemId;

/\*\* 绑定的订单号\*/

@Deprecated

private String orderId;

/\*\* 商品来源渠道 \*/

@Deprecated

private String channel;

**6. 开发、压测结果**

**7. 文档说明**

1. 编写目的: 省去常规开发中的冗余环节，抓取核心开发流程，重点在梳理开发思路、解决方案、细节考量
2. 文档命名规范: 项目名 - 模块名 - 开发文档。例如: 咚宝商城 - 支付模块 - 开发流程文档