03-复杂而又重要的购物车系统，应该如何设计？

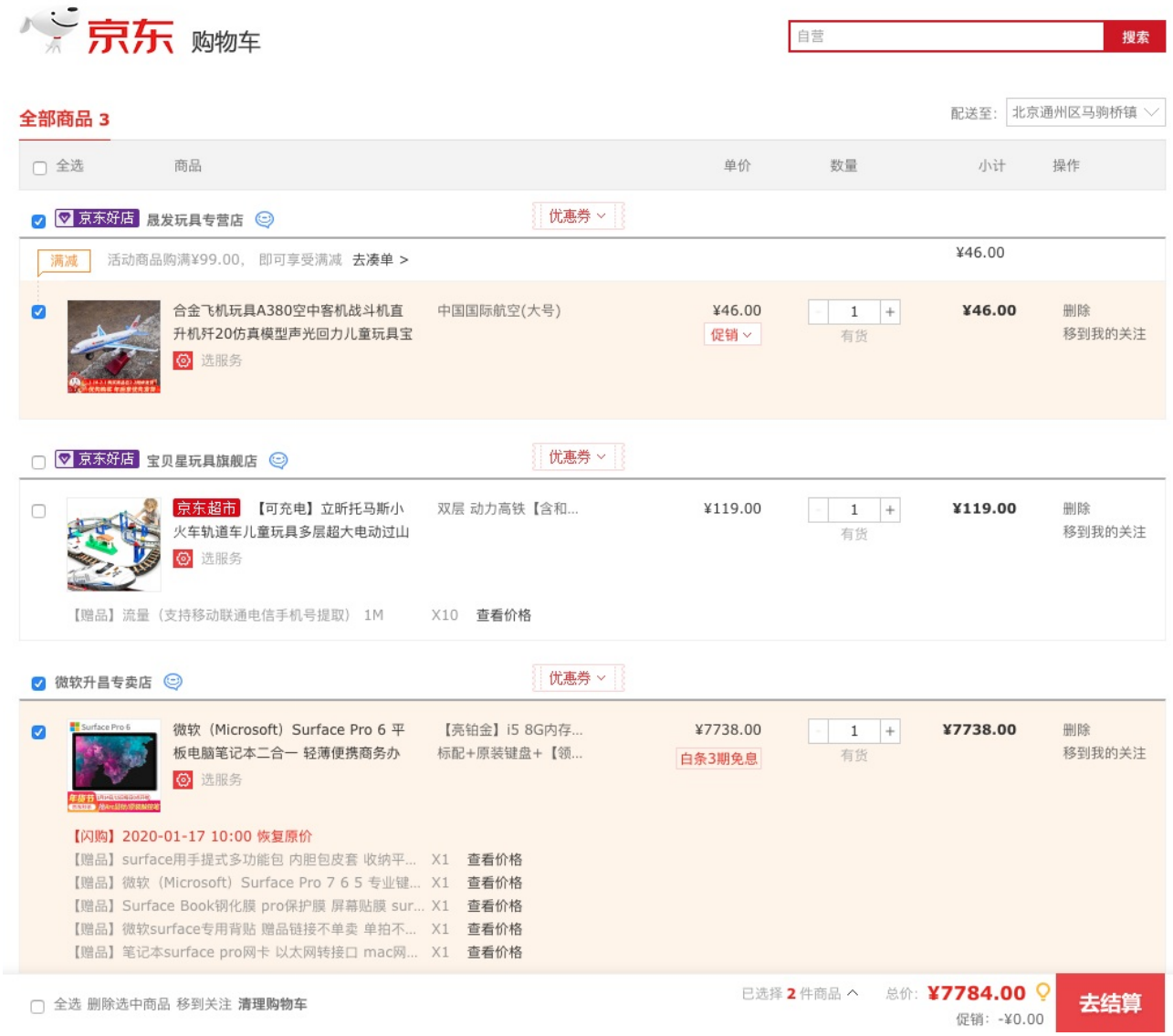
你好，我是李玥。

今天这节课我们来说一下购物车系统的存储该如何设计。

首先，我们来看购物车系统的主要功能是什么。就是在用户选购商品时，下单之前，暂存用户想要购买的商品。购物车对数据可靠性要求不高，性能也没有特别的要求，在整个电商系统中，看起来是相对比较容易设计和实现的一个子系统。

购物车系统的功能，主要的就三个：把商品加入购物车（后文称“加购”）、购物车列表页、发起结算下单，再加上一个在所有界面都要显示的购物车小图标。

支撑购物车的这几个功能，对应的存储模型应该怎么设计？很简单，只要一个“购物车”实体就够了。它的主要属性有什么？你打开京东的购物车页面，对着抄就设计出来了：SKUID（商品ID）、数量、加购时间和勾选状态。



（备注：图片来源于网络，仅供本文介绍、评论及说明某问题，适当引用。）

这个“勾选状态”属性，就是在购物车界面中，每件商品前面的那个小对号，表示在结算下单时，是不是要包含这件商品。至于商品价格和总价、商品介绍等等这些信息，都可以实时从其他系统中获取，不需要购物

车系统来保存。

购物车的功能虽然很简单，但是在设计购物车系统的存储时，仍然有一些特殊的问题需要考虑。

设计购物车存储时需要把握什么原则？

比如下面这几个问题：

1. 用户没登录，在浏览器中加购，关闭浏览器再打开，刚才加购的商品还在不在？
2. 用户没登录，在浏览器中加购，然后登录，刚才加购的商品还在不在？
3. 关闭浏览器再打开，上一步加购的商品在不在？
4. 再打开手机，用相同的用户登录，第二步加购的商品还在不在呢？

上面这几个问题是不是有点儿绕？没关系，我们先简单解释一下这四个问题：

1. 如果用户没登录，加购的商品也会被保存在用户的电脑里，这样即使关闭浏览器再打开，购物车的商品仍然存在。
2. 如果用户先加购，再登录，登录前加购的商品就会被自动合并到用户名下，所以登录后购物车中仍然有登录前加购的商品。
3. 关闭浏览器再打开，这时又变为未登录状态，但是之前未登录时加购的商品已经被合并到刚刚登录的用户名下了，所以购物车是空的。
4. 使用手机登录相同的用户，看到的就是该用户的购物车，这时无论你在手机App、电脑还是微信中登录，只要是相同的用户，看到是同一个购物车，所以第二步加购的商品是存在的。

所以，上面这四个问题的答案依次是：存在、存在、不存在、存在。

如果你没有设计或者开发过购物车系统，你可能并不会想到购物车还有这么多弯弯绕。但是，作为一个开发者，如果你不仔细把这些问题考虑清楚，用户在使用购物车的时候，就会感觉你的购物车系统不好用，不是加购的商品莫名其妙地丢了，就是购物车莫名其妙地多出来一些商品。

要解决上面这些问题，其实只要在存储设计时，把握这几个原则就可以了：

1. 如果未登录，需要临时暂存购物车的商品；
2. 用户登录时，把暂存购物车的商品合并到用户购物车中，并且清除暂存购物车；
3. 用户登陆后，购物车中的商品，需要在浏览器、手机APP和微信等等这些终端中都保持同步。

实际上，购物车系统需要保存两类购物车，**一类是未登录情况下的“暂存购物车”，一类是登录后的“用户购物车”**。

如何设计“暂存购物车”的存储？

我们先来看下暂存购物车的存储该怎么实现。暂存购物车应该存在客户端还是存在服务端？

如果保存在服务端，那每个暂存购物车都需要有一个全局唯一的标识，这个标识并不太容易设计，并且，存在服务端还要浪费服务端的资源。所以，肯定是保存在客户端好，既可以节约服务器的存储资源，也没有购物车标识的问题，因为每个客户端就保存它自己唯一一个购物车就可以了，不需要标识。

客户端的存储可以选择的不太多：Session、Cookie和LocalStorage，其中浏览器的LocalStorage和App的本地存储是类似的，我们都以LocalStorage来代表。

存在哪儿最合适？SESSION是不太合适的，原因是，SESSION的保留时间短，而且SESSION的数据实际上还是保存在服务端的。剩余的两种存储，Cookie和LocalStorage都可以用来保存购物车数据，选择哪种方式更好呢？各有优劣。

在我们这个场景中，使用Cookie和LocalStorage最关键的区别是，客户端和服务端的每次交互，都会自动带着Cookie数据往返，这样服务端可以读写客户端Cookie中的数据，而LocalStorage里的数据，只能由客户端来访问。

使用Cookie存储，实现起来比较简单，加减购物车、合并购物车的过程中，由于服务端可以读写Cookie，这样全部逻辑都可以在服务端实现，并且客户端和服务端请求的次数也相对少一些。

使用LocalStorage存储，实现相对就复杂一点儿，客户端和服务端都要实现一些业务逻辑，但LocalStorage的好处是，它的存储容量比Cookie的4KB上限要大得多，而且不用像Cookie那样，无论用不用，每次请求都要带着，可以节省带宽。

所以，选择Cookie或者是LocalStorage来存储暂存购物车都是没问题的，你可以根据它俩各自的优劣势来选择。比如你设计的是个小型电商，那用Cookie存储实现起来更简单。再比如，你的电商是面那种批发的行业用户，用户需要加购大量的商品，那Cookie可能容量不够用，选择LocalStorage就更合适。

不管选择哪种存储，暂存购物车保存的数据格式都是一样的，参照我们实体模型来设计就可以，我们可以直接用JSON表示：

```
{
    "cart": [
        {
            "SKUID": 8888,
            "timestamp": 1578721136,
            "count": 1,
            "selected": true
        },
        {
            "SKUID": 6666,
            "timestamp": 1578721138,
            "count": 2,
            "selected": false
        }
    ]
}
```

如何设计“用户购物车”的存储?

接下来，我们再来看下用户购物车的存储该怎么实现。因为用户购物车必须要保证多端的数据同步，所以数据必须保存在服务端。常规的思路是，设计一张购物车表，把数据存在MySQL中。这个表的结构同样可以参照刚刚讲的实体模型来设计：

列名	数据类型	主键	非空	说明
id	BIGINT	是	是	自增主键
user_id	BIGINT		是	用户ID
sku_id	BIGINT		是	商品ID
count	INT		是	商品数量
timestamp	DATE		是	加购时间
selected	TINYINT (1)			购选状态

注意，需要在user_id上建一个索引，因为查询购物车表时，都是以user_id作为查询条件来查询的。

你也可以选择更快的Redis来保存购物车数据，以用户ID作为Key，用一个Redis的HASH作为Value来保存购物车中的商品。比如：

```
{
    "KEY": 6666,
    "VALUE": [
        {
            "FIELD": 8888,
            "FIELD_VALUE": {
                "timestamp": 1578721136,
                "count": 1,
                "selected": true
            }
        },
        {
            "FIELD": 6666,
            "FIELD_VALUE": {
                "timestamp": 1578721138,
                "count": 2,
                "selected": false
            }
        }
    ]
}
```

这里为了便于你理解，我们用JSON来表示Redis中HASH的数据结构，其中KEY中的值6666是一个用户ID，FIELD里存放的是商品ID，FIELD_VALUE是一个JSON字符串，保存加购时间、商品数量和勾选状态。

大家都知道，从读写性能上来说，Redis是比MySQL快非常多的，那是不是用Redis就一定比用MySQL更好呢？我们来比较一下使用MySQL和Redis两种存储的优劣势：

1. 显然使用Redis性能要比MySQL高出至少一个量级，响应时间更短，可以支撑更多的并发请求，“天下武功，唯快不破”，这一点Redis完胜。
2. MySQL的数据可靠性是要好于Redis的，因为Redis是异步刷盘，如果出现服务器掉电等异常情况，Redis是有可能会丢数据的。但考虑到购物车里的数据，对可靠性要求也没那么苛刻，丢少量数据的后果也就

是，个别用户的购物车少了几件商品，问题也不大。所以，在购物车这个场景下，Redis的数据可靠性不高这个缺点，并不是不能接受的。

3. MySQL的另一个优势是，它支持丰富的查询方式和事务机制，这两个特性，对我们今天讨论的这几个购物车核心功能没什么用。但是，每一个电商系统都有它个性化的需求，如果需要以其他方式访问购物车的数据，比如说，统计一下今天加购的商品总数，这个时候，使用MySQL存储数据，就很容易实现，而使用Redis存储，查询起来就非常麻烦而且低效。

综合比较下来，考虑到需求总是不断变化，还是更推荐你使用MySQL来存储购物车数据。如果追求性能或者高并发，也可以选择使用Redis。

你可以感受到，我们设计存储架构的过程就是一个不断做选择题的过程。很多情况下，可供选择的方案不止一套，选择的时候需要考虑实现复杂度、性能、系统可用性、数据可靠性、可扩展性等等非常多的条件。需要强调的是，**这些条件每一个都不是绝对不可以牺牲的，不要让一些“所谓的常识”禁锢了你的思维。**

比如，一般我们都认为数据是绝对不可以丢的，也就是说不能牺牲数据可靠性。但是，像刚刚讲到的用户购物车的存储，使用Redis替代MySQL，就是牺牲了数据可靠性换取高性能。我们仔细分析后得出，很低概率的情况下丢失少量数据，是可以接受的。性能提升带来的收益远大于丢失少量数据而付出的代价，这个选择就是划算的。

如果说不考虑需求变化这个因素，牺牲一点点数据可靠性，换取大幅性能提升，选择Redis才是最优解。

小结

今天我们讲了购物车系统的存储该如何设计。

购物车系统的主要功能包括：加购、购物车列表页和结算下单。核心的实体就只有一个“购物车”实体，它至少要包括：SKUID、数量、加购时间和勾选状态这几个属性。

在给购物车设计存储时，为了确保购物车内的数据在多端保持一致，以及用户登录前后购物车内商品能无缝衔接，除了每个用户的“用户购物车”之外还要实现一个“暂存购物车”保存用户未登录时加购的商品，并在用户登录后自动合并“暂存购物车”和“用户购物车”。

暂存购物车存储在客户端浏览器或者App中，可以选择存放到Cookie或者LocalStorage中。用户购物车保存在服务端，可以选择使用Redis或者是MySQL存储，使用Redis存储会有更高的性能，可以支撑更多的并发请求，使用MySQL是更常规通用的方式，便于应对变化，系统的扩展性更好。

思考题

课后请你思考一下，既然用户的购物车数据存放在MySQL或者是Redis中各有优劣势。那能不能把购物车数据存在MySQL中，并且用Redis来做缓存呢？这样不就可以兼顾两者的优势了么？这样做是不是可行？如果可行，如何来保证Redis中的数据和MySQL中的数据是一样的呢？

欢迎你在留言区与我讨论，如果你觉得今天学到的知识对你有帮助，也欢迎把它分享给你的朋友。

精选留言：

- 李玥 2020-03-03 10:06:51
  hi，我是李玥。

上节课我给你留了一道思考题，是这样的。如果说，用户下单这个时刻，正好赶上商品调价，就有可能出现这样的情况：我明明在商详页看到的价格是10块钱，下单后，怎么变成15块了？你的系统是不是偷偷在坑我？给用户的体验非常不好。你不要以为这是一个小概率事件，当你的系统用户足够多的时候，每时每刻都有人在下单，这几乎是个必然出现的事件。该怎么来解决这个问题？

关于这个问题，我是这样看的。

首先，商品系统需要保存包含价格的商品基本信息的历史数据，对每一次变更记录一个自增的版本号。在下单的请求中，不仅要带上SKUID，还要带上版本号。订单服务以请求中的商品版本对应的价格来创建订单，就可以避免“下单时突然变价”的问题了。

但是，这样改正之后会产生一个很严重的系统漏洞：黑客有可能会利用这个机制，以最便宜的历史价格来下单。所以，我们在下单之前需要增加一个检测逻辑：请求中的版本号只能是当前版本或者上一个版本，并且使用上一个版本要有一个时间限制，比如说调价5秒之后，就不再接受上一个版本的请求。这样就可以避免这个调价漏洞了。

- 肥low 2020-03-03 09:53:29
  我觉得完全可行 而且有时候比如MySQL主从架构下是有数据延迟更新问题的 用Redis我可以尽量避免这一点 不过有对用户加购的维护成本 [1赞]

  作者回复2020-03-03 10:28:33
  我会在下节课的评论区说一下我的理解，请关注。

- 黄海峰 2020-03-03 09:08:33
  感觉购物车是写多于读，也就是经常变，用cache aside的方式保持一致性的话就经常删缓存，db压力减轻不了多少，还要多写一次缓存，没什么必要

- 刘楠 2020-03-03 09:03:44
  可行，更新购物车的时候写mysql同时删除缓存，读的时候优先读redis，没有在打到db，同时cache一份到redis，这样应该可以保证一致性
  同时，能不能每节课时把上节课的思考师解答下，谢谢

- 墨雨 2020-03-03 08:49:49
  不考虑复杂性和服务器成本的话，我认为是可行的。跟老师之前讲的方法一样，每次查询购物车先在 redis 里查，查不到再到 mysql 中查同时更新 redis 中数据。更新用户购物车数据时删除 redis 中数据。但我有一个问题是:用户本身购物车没数据的时候会导致 redis 和 mysql 查两遍……

- Cranliu 2020-03-03 08:08:04
  当然是可以的。一致性的问题采用主动更新缓存解决。

- aoe 2020-03-03 02:48:27
  思考题是可行的，但是复杂，例如需要考虑：
  1. Redis的容量可能远小于数据库容量，需要缓存策略缓存数据
  2. 要处理老师提到的一致性问题
  3. 性价比

  另外请教老师一个问题：
  在电商系统中，订单的“商品快照”（商品名称、数量、详情页所有信息）一般是怎么存储的？
  例如：有订单、商品2个子系统，订单的商品快照一般是由哪个系统生成和保存？

作者回复2020-03-03 10:14:42
我个人的看法是，商品子系统存储商品快照更合理一些。

- leslie 2020-03-03 01:29:33
这个问题的回答应当从两种数据库特性去说起吧：redis的特点是存储于内存，但是数据落地刷盘、、、mysql的特性是数据存储于硬盘。
由于存于内存故而查询速度非常可观：购物车环节其实商品变革的频率蛮高的，此时如果直接每次增删商品都访问硬盘数据库，这个代价就、、、尤其是在高并发场景下，真正与金额直接产生的交互的环节是结算环节，即付款；我记得老师曾经在消息队列的期中考试中考过什么场景下数据会丢失
故而今天课程的答案：我个人倾向在结算之前不落mysql数据库：购物车环节直接用redis以减少购物车频繁改动而带来大量的IO消耗。
另外有一点我不太确定：LocalStorage的安全性如何？cookie的安全性问题一直颇受争议，此时在购物车环节去直接访问结算数据库，觉得欠妥。
谢谢老师的分享：去年的课程受益匪浅，最近正在准备抽空过第二遍；希望这门课程同样能收获不一样的东西；谢谢。

- Sephiroth 2020-03-03 00:54:43
应该可行，更新购物车的时候写mysql同时修改redis，读的时候优先读redis，没有在打到db，同时cache一份到redis，这样应该可以保证一致性