Discount Model

Maxim & Pavel

March 30, 2016

1 Purpose

Customers want to be able to discount products and run promotions to help drive sales. Promotions are a merchandising tool and having a flexible one allows the customer to be creative in their business.

2 Forces

- Provide flexibility without sacraficing integrity.
- This is a combinatorial problem.
- Must only allow valid configurations.

3 Architecture

3.1 Parts

The discounting system has two parts.

1. Merchandising Part
2. Execution Part

The merchandising part will use our Object System to store information that can change by context. The execution part will be composed of several pieces.

- Qualifier and Offer Algebras
- Algebra Compiler

- Discount Executor
- Promotion Matching

3.2 Flow

1. The promotions/discounts are stored in our DB.
2. Given an order, the system will find a promotion.
3. Take promotion and compile discounts.
4. Execute discounts on order to produce line item adjustments.

Figure 1: Flow

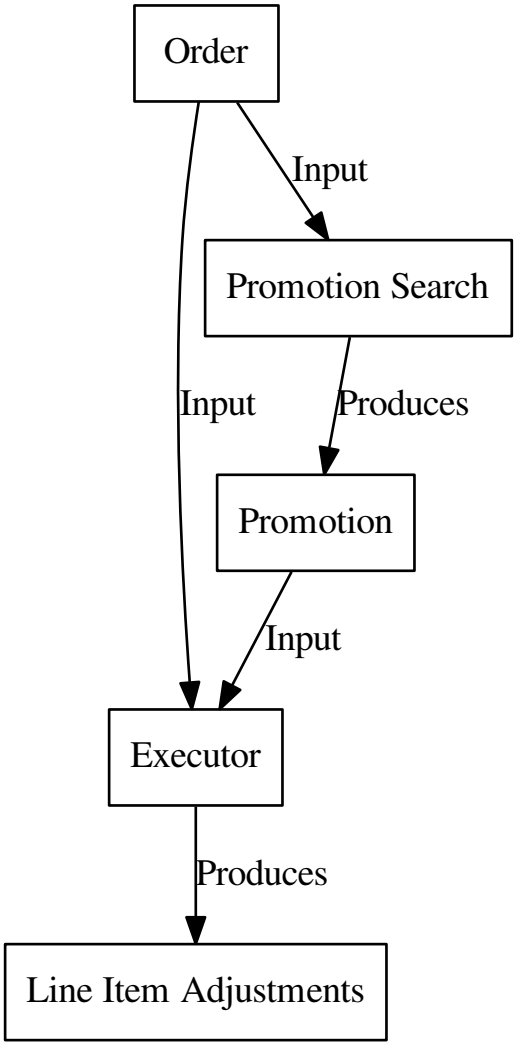
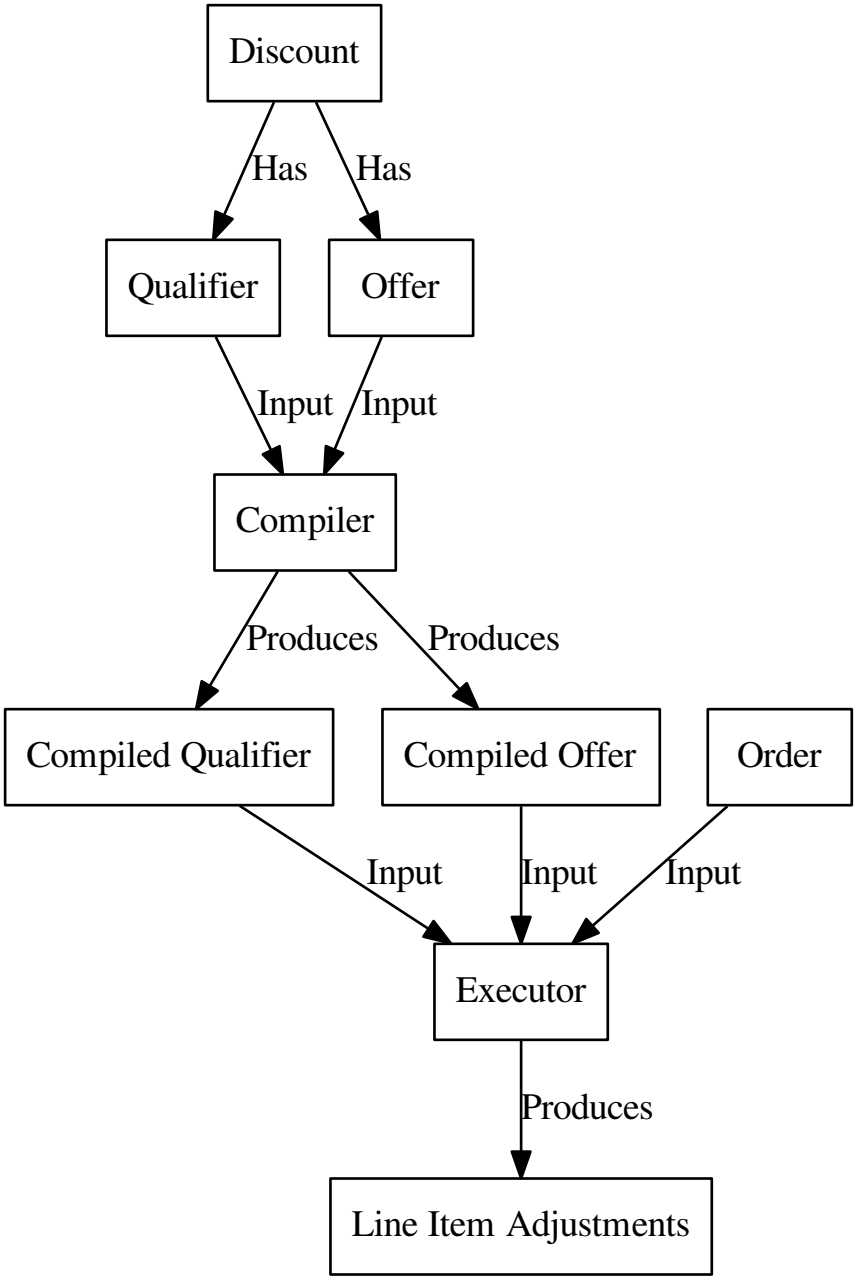


Figure 2: Applying Discounts



4 Models

The central concept is the discount. The discounts are bundled in a promotion and a promotion can be made public via a coupon.

5 Discounts

Discounts have three parts.

1. Merchandising part.
2. Qualifier part.
3. Offer part.

5.1 Merchandising Discounts

Each discount will have merchandising data attached to it that can be used for organizing and merchandising. For example, a title, a description, and tags. The merchandising information can change based on the context.

5.2 Qualifying Discounts

A discount can only be used by a customer if they qualify for it. The qualification is going to be expressed as a predicate function which takes a customer/order bundle as a parameter.

This predicate function will be expressed in a simple algebra which allows the user to configure the predicate function.

5.3 Offers

A discount also has an offer, which describes how the order changes. For example, "30

The offer will be expressed as a function that takes a customer/order bundle as a parameter and produces line item adjustments.

This offer function will be expressed as a simple algebra which allows the user to configure the offer to their liking.

6 Promotions

Promotions are bundles of discounts and have two parts

1. Merchandising part.
2. Discount bundle.

6.1 Merchandising Promotions

Promotions can have merchandising information such as title, description, images, and tags. This information is used for both display and organization. Merchandising information will change based on context.

6.2 Discount Bundle

The essence of the promotion is that it is composed of one or more discounts bundled together. This is a simple set of discount references.

7 Coupons

Coupons are a way to restrict access to a promotion that isn't available publicly. Coupons have three parts.

1. Coupon code.
2. Merchandising part.
3. Promotion.

7.1 Coupon code

The coupon code is a way to provide a handle to a hidden promotion. The code can be shared to a specific customer, or to many customers. The code is generated on demand when a coupon is created or copied. We may provide facilities to bulk generate codes.

7.2 Merchandising a Coupon

Coupons may be displayed to a customer to entice them to use the code. For example The coupon may be sent via an email and have a title, description, and image.

The merchandising information will change depending on context.

7.3 Promotion

The coupon finally points to a specific promotion.

8 Data Model

Here we will describe the data model of discounts, promotions, and coupons as they would appear to a client. The description is in JSON.

Since the discount, promotion, and coupon may change depending on context, we will store the information in our existing Object storage system. This will give us the power of change based on context but also versioning.

8.1 Discount

```
//MVP Discount
{
    title: { type: "string", ... }
    description: { type: "richText", ... }
    tags: { type: "tags" ... }
    qualifier: { type: "qualifier" ... }
    type: { type: "discountType" ... }
    offer: { type: "offer" ... }
}
```

The qualifier and offer will be stored as a JSON Object which represents the AST of the respective algebra grammars.

8.2 Promotion

```
//MVP Promotion
{
    title: { type: "string", ref : "..." }
    description: { type: "richText", ref : "..." }
    tags: { type: "tags" ref : "..." }
    images: { type: "images" ref : "..." }
    activeFrom: { type: "date" ref : "..." }
    activeTo: { type: "date" ref : "..." }
    isPublic: { type: "bool" ref : "..." }
}
```

The set of discounts will be stored using our ObjectLink system.

9 Coupon

```
    //MVP Promotion
    {
        code: { type: "string", ref : "..." }
        title: { type: "richText", ref : "..." }
        description: { type: "richText", ref : "..." }
        tags: { type: "tags" ref : "..." }
        images: { type: "images" ref : "..." }
        maxUses: { type: "number" ref : "..." }
        maxCustomerUses: { type: "number" ref : "..."
}
    }
```

Additionally, the Coupon in a context will have the following stored

```
class CouponUsage(id: Int, couponId: Int, total: Int)
class CouponCustomerUsage(id: Int, usageId: Int,
    customerId: Int, total: Int)
```

This is to keep track of how many times it was used in total and by a specific customer.

10 Discussion

- Should the coupon merchandising information simply come from the promotion instead of storing it at the coupon level?