Decision Model for Sales Order Promotions

Bruce Silver, methodandstyle.com

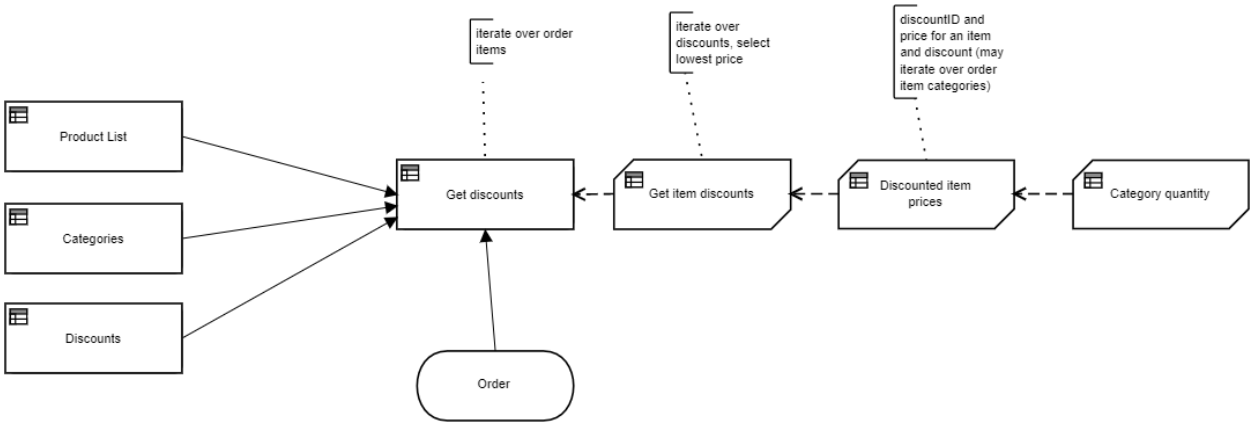
I chose to model in DMN the functionality used on my own website, WooCommerce Dynamic Pricing. WooCommerce is the default ecommerce plugin for Wordpress websites, and Dynamic Pricing is a premium extension integrated with the WooCommerce object model. In that model, each *product* may have multiple *variations* and may belong to zero or more *categories*. Other parameters used in Dynamic Pricing are the *order date* and the buyer's *role*, which could be either a loyalty program level ("Gold", "Silver", etc.,) or "Partner", "Employee", and so forth. In my model I ignored product variations. In other respects, the DMN model more or less conforms to the functionality of WooCommerce Dynamic Pricing.

Dynamic Pricing supports several named discount types:

- Order Totals: Percentage discount for eligible products if order date and role are eligible and total value of order exceeds threshold
- Roles: Fixed or percentage discount on entire order for eligible role
- Simple Category: Fixed or percentage discount for all items in selected category
- Single product: Fixed or percentage discount or fixed price for eligible products if order date and role are eligible and quantity of qualifying products in range or quantity of items in qualifying category in range
- Advanced Category: Fixed or percentage discount or fixed price for items in eligible categories if order date and role are eligible and quantity of qualifying products in range or quantity of items in qualifying category in range

The first 3 are simple to configure in Dynamic Pricing and relatively simple to implement in DMN. The last 2 are complex to configure and to model in DMN.

DMN Model



The DRD is shown above. The three zero-input decisions modeled as relations are static data embedded in the model. *Product List* is a list of products and prices. *Categories* is a list of categories and the products that comprise them. *Discounts* is a list of current discounts available in the store.

The input data *Order* specifies the Customer, including customer role, and a list of order items, each specified by unique name and quantity.

Product List
tProductList

name	SKU	unit price
tProductName *"Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket", "Red sprocket", "Blue sprocket", "White sprocket", "Any"*	*Text*	*Number*
"Red widget"	"R001"	19.95
"Blue widget"	"B001"	21.95
"White widget"	"W001"	14.95
"Red sprocket"	"R002"	47.05
"Blue sprocket"	"B002"	51.17
"White sprocket"	"W002"	27.99
"Red trinket"	"R003"	1.75
"Blue trinket"	"B003"	1.28
"White trinket"	"W003"	2.05

Categories
tCategoryList

name	products
tCategoryName *"widgets", "sprockets", "trinkets", "red stuff", "blue stuff", "white stuff", "Any"*	*tProductNameList* *"Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket", "Red sprocket", "Blue sprocket", "White sprocket", "Any"*
"widgets"	["Red widget", "Blue widget", "White widget"]
"sprockets"	["Red sprocket", "Blue sprocket", "White sprocket"]
"trinkets"	["Red trinket", "Blue trinket", "White trinket"]
"red stuff"	["Red widget", "Red sprocket", "Red trinket"]
"blue stuff"	["Blue widget", "Blue sprocket", "Blue trinket"]
"white stuff"	["White widget", "White sprocket", "White trinket"]

Discounts
tDiscountList

ID	name	type	amount	appliesToProducts	appliesToCategories	appliesToRoles	startDate	endDate	qualifyingTotalPrice	qualifyingMinQty	qualifyingMaxQty	qualifyingProducts	qualifyingCategories
Text	*tDiscountName* *"Order totals", "Roles", "Single product", "Simple category", "Advanced category"*	*tDiscountType* *"Fixed discount", "Percentage discount", "Fixed price"*	*Number*	*tProductNameList* *"Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket", "Red sprocket", "Blue sprocket", "White sprocket", "Any"*	*tCategoryNameList* *"widgets", "sprockets", "trinkets", "red stuff", "blue stuff", "white stuff", "Any"*	*tCustomerRoleList* *"Partner", "Gold", "Silver", "None", "Any"*	*Date*	*Date*	*Number*	*Number*	*Number*	*tProductNameList* *"Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket", "Red sprocket", "Blue sprocket", "White sprocket", "Any"*	*tCategoryNameList* *"widgets", "sprockets", "trinkets", "red stuff", "blue stuff", "white stuff", "Any"*
"1a"	"Order totals"	"Percentage discount"	1	["Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket"]	null	["Silver", "None"]	date("2018-01-01")	date("2018-12-31")	1000	null	null	["Any"]	["Any"]
"1b"	"Order totals"	"Percentage discount"	9	["Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket"]	null	["Gold", "Partner"]	date("2018-01-01")	date("2018-12-31")	1000	null	null	["Any"]	["Any"]
"2a"	"Roles"	"Percentage discount"	13	["Any"]	null	["Partner"]	date("2018-01-01")	date("2018-12-31")	null	null	null	["Any"]	["Any"]
"3a"	"Single product"	"Fixed price"	1.50	["White trinket"]	null	["Any"]	date("2018-01-01")	date("2018-12-31")	null	5	null	["White widget"]	null
"3b"	"Single product"	"Fixed discount"	5	["Red sprocket", "Blue sprocket"]	null	["Any"]	date("2018-01-01")	date("2018-12-31")	null	10	null	null	["white stuff"]
"3c"	"Single product"	"Fixed discount"	5	["Blue sprocket"]	null	["Any"]	null	null	null	2	4	["Blue sprocket"]	null
"3d"	"Single product"	"Fixed discount"	8	["Blue sprocket"]	null	["Any"]	null	null	null	5	null	["Blue sprocket"]	null
"4a"	"Simple category"	"Percentage discount"	12	null	["sprockets"]	null	null	null	null	null	null	null	["Any"]
"4b"	"Simple category"	"Percentage discount"	10	null	["blue stuff"]	null	null	null	null	null	null	null	["Any"]
"5a"	"Advanced category"	"Percentage discount"	15	null	["widgets", "sprockets"]	["Gold"]	date("2018-01-01")	date("2018-03-01")	null	100	null	null	["trinkets"]

I've defined 10 discounts:

1a, 1b. If the total order value is over 1000 and role is Silver or None and order date in 2018, discount red, blue, and white widget and trinket by 1%. If the role is Gold or Partner discount those items by 9%.

2a. If the role is Partner and order date in 2018, discount all items by 13%.

3a. If order date in 2018 and order contains at least 5 White widget, offer fixed price of $1.50 for White trinket.

3b. If order date in 2018 and order contains at least 10 items in category White stuff, discount Blue sprocket and Red sprocket by $5.

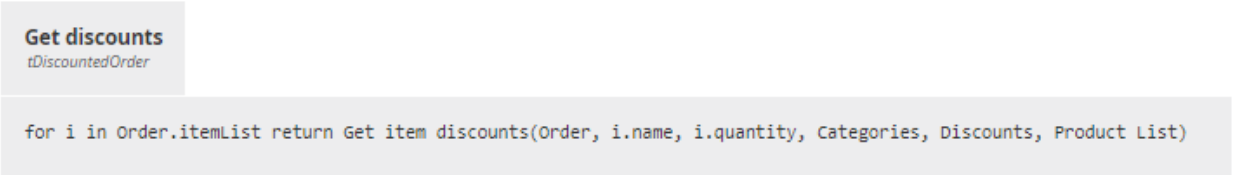
3c, 3d. If order contains 2-4 Blue sprocket, discount Blue sprocket by $5. If order contains 5 or more Blue sprocket, discount Blue sprocket by $10.

4a. Discount any item in category sprockets by 12%.

4b. Discount any item in category blue stuff by 10%.

5a. If order date January-March 2018 and buyer role is Gold and order contains at least 100 items in category trinkets, discount item in categories widgets and sprockets by 15%.

The decision *Get discounts* iterates the BKM *Get item discounts* over each order item



Since more than one discount may apply to each order item, the model selects the lowest discounted price. For each order item, *Get item discounts* determines the items categories and unit price, then iterates over the various Discounts, invoking *Discounted item prices* to generate a list of discounted prices for the item, and finally creates a row returned to *Get discounts*, selecting the lowest discounted price for the item.

The BKM *Discounted item prices* does the real logic of determining whether a discount is applicable to the item and if so, applying it. The tricky part is when item A is eligible based on the quantity of item B or category C, involving nested loops, table joins, and similar nasty things that FEEL can do but is a little obscure.

If the discount is based on the quantity of qualifying items in a category, *Discounted item prices* must iteratively invoke *Category quantity* over the qualifying categories for the discount. It's like one of those Russian doll puzzles.

Get item discounts

tdiscountedItemRow

(order *tOrder* , itemName *tProductName* *"Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket", "Red sprocket", "Blue sprocket", "White sprocket", "Any"* , itemQty *Number* , categoriesList *tCategoryList* , discountsList *tDiscountList* , productsList *tProductList*)

1	itemCategoriesRaw *tCategoryList*	`for j in categoriesList return if list contains(j.products, itemName) then j else null`
2	itemCategories *tCategoryList*	`itemCategoriesRaw[item!=null]`
3	itemUnitPrice *Number*	`productsList[name=itemName].unit price`
4	itemDiscountsList *tDiscountedPriceList*	`for i in discountsList return Discounted item prices(i, itemName, itemQty, itemUnitPrice, itemCategories, order, categoriesList, productsList)`
5	discountedItemRow *tdiscountedItemRow*	

1	item name *tProductName* *"Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket", "Red sprocket", "Blue sprocket", "White sprocket", "Any"*	`itemName`
2	item quantity *Number*	`itemQty`
3	regular price *Number*	`itemUnitPrice`
4	discounted price *Number*	`min(itemDiscountsList.price)`
5	discountID *Text*	`if discounted price < regular price then itemDiscountsList[price=discounted price].ID[1] else "None"`

5	discountedItemRow *tdiscountedItemRow*	

1	item name *tProductName* *"Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket", "Red sprocket", "Blue sprocket", "White sprocket", "Any"*	`itemName`
2	item quantity *Number*	`itemQty`
3	regular price *Number*	`itemUnitPrice`
4	discounted price *Number*	`min(itemDiscountsList.price)`
5	discountID *Text*	`if discounted price < regular price then itemDiscountsList[price=discounted price].ID[1] else "None"`
6	regular total *Number*	`item quantity*regular price`
7	discounted total *Number*	`item quantity*discounted price`

`Result`

`discountedItemRow`

Discounted item prices

tDiscountedPrice

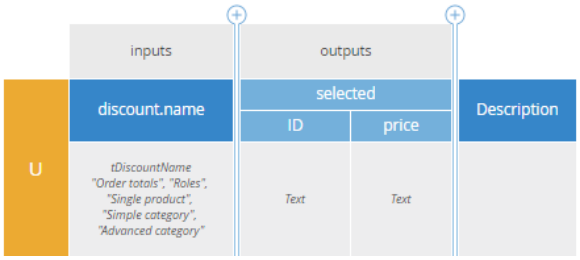
(discount *tDiscount* , itemName *tProductName* *"Red widget", "Blue widget", "White widget", "Red trinket", "Blue trinket", "White trinket", "Red sprocket", "Blue sprocket", "White sprocket", "Any"* , itemQty *Number* , itemUnitPrice *Number* , itemCategories *tCategoryList* , order *tOrder* , categoriesList *tCategoryList* , productsList *tProductList*)

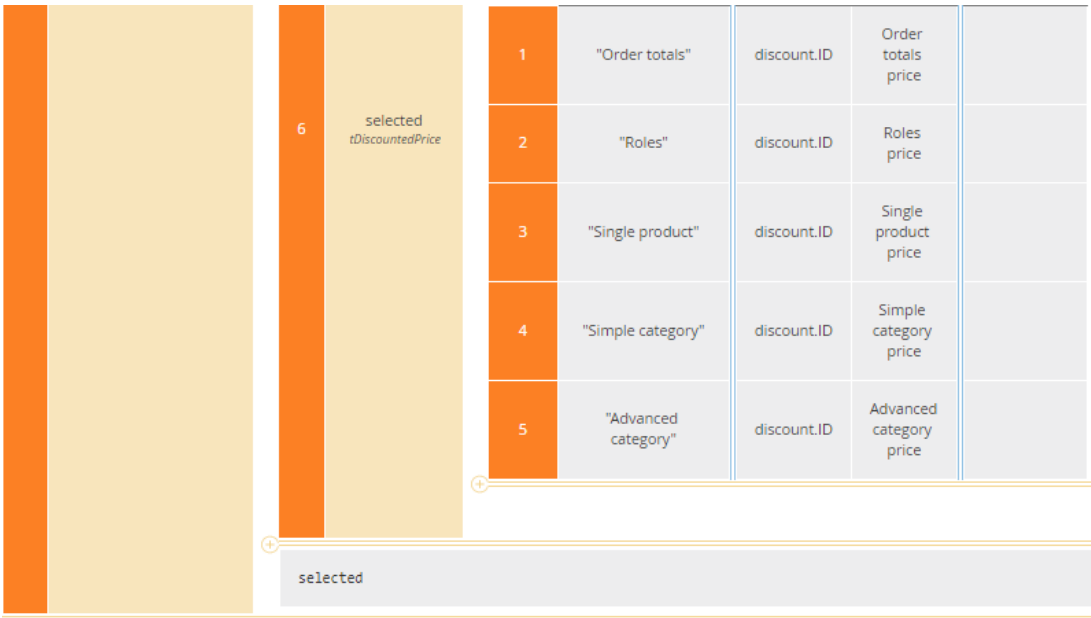
1	isDateValid *Boolean*	`(discount.startDate=null or order.date>=discount.startDate) and (discount.endDate=null or order.date<=discount.endDate)`
2	isRoleApplicable *Boolean*	`list contains(discount.appliesToRoles, order.customer.role) or list contains(discount.appliesToRoles,"Any") or discount.appliesToRoles=null`
3	isProductApplicable *Boolean*	`list contains(discount.appliesToProducts, itemName) or list contains(discount.appliesToProducts,"Any") or discount.appliesToProducts=null`
4	isCategoryApplicable *Boolean*	`discount.appliesToCategories=null or list contains(discount.appliesToCategories,"Any") or some i in discount.appliesToCategories satisfies list contains(itemCategories.name, i)`
5	doesOrderTotalQualify *Boolean*	`sum(for i in order.itemList return i.quantity*productsList[name=i.name][1].unit price) > discount.qualifyingTotalPrice`
6	doesProductQtyQualify *Boolean*	`discount.qualifyingProducts=null or list contains(discount.qualifyingProducts,"Any") or (some i in order.itemList satisfies (list contains(discount.qualifyingProducts, i.name) and (i.quantity>=discount.qualifyingMinQty or discount.qualifyingMinQty=null) and (i.quantity<=discount.qualifyingMaxQty or discount.qualifyingMaxQty=null)))`
7	doesCategoryQtyQualify *Boolean*	

1	orderCategories *tCategoryList*	`for i in order.itemList, j in categoriesList return if list contains(j.products, i.name) then j.name else null`
2	orderCategoryQuantities *tCategoryQuantityList*	`for i in orderCategories[item!=null] return Category quantity(i, order, categoriesList)`

8	discounted price *tDiscountedPrice*	

1	Order totals price *Number*	`if isDateValid and isRoleApplicable and isProductApplicable and doesOrderTotalQualify then (if discount.type="Fixed discount" then itemUnitPrice - discount.amount else if discount.type="Fixed price" then discount.amount else decimal(itemUnitPrice*(1-.01*discount.amount),2)) else itemUnitPrice`
2	Roles price *Number*	`if isDateValid and isRoleApplicable and isProductApplicable then (if discount.type="Percentage discount" then decimal(itemUnitPrice* (1-.01*discount.amount),2) else if discount.type="Fixed discount" then itemUnitPrice - discount.amount else discount.amount) else itemUnitPrice`
3	Single product price *Number*	`if isDateValid and isRoleApplicable and isProductApplicable and doesProductQtyQualify and doesCategoryQtyQualify then (if discount.type="Fixed discount" then itemUnitPrice - discount.amount else if discount.type="Fixed price" then discount.amount else decimal(itemUnitPrice*(1-.01*discount.amount),2)) else itemUnitPrice`
4	Simple category price *Number*	`if isDateValid and isCategoryApplicable and discount.type="Percentage discount" then decimal(itemUnitPrice*(1-.01*discount.amount),2) else itemUnitPrice`
5	Advanced category price *Number*	`if isDateValid and isRoleApplicable and isCategoryApplicable and doesCategoryQtyQualify then (if discount.type="Fixed discount" then itemUnitPrice - discount.amount else if discount.type="Fixed price" then discount.amount else decimal(itemUnitPrice*(1-.01*discount.amount),2)) else itemUnitPrice`





Category quantity
tCategoryquantity

(catName *tCategoryName* "widgets", "sprockets", "trinkets", "red stuff", "blue stuff", "white stuff", "Any" , order *tOrder* , categoriesList *tCategoryList*)

1	qty *Number*	sum(for i in order.itemList return (if list contains(categoriesList[name=catName][1].products, i.name) then i.quantity else 0))

Result

Execution

Test case 2

Inputs

Order
ID
5678
date
2018-01-25
customer
name
Bruce
ID
1234
role
Silver

name	quantity
Red widget	10
White widget	6
Blue trinket	50
White trinket	10
Red sprocket	13
Blue sprocket	3

item name	item quantity	regular price	discounted price	discountID	regular total	discounted total
Red widget	10	19.95	19.75	1a	199.50	197.50
White widget	6	14.95	14.80	1a	89.70	88.80
Blue trinket	50	1.28	1.15	4b	64.00	57.50
White trinket	10	2.05	1.50	3a	20.50	15.00
Red sprocket	13	47.05	41.40	4a	611.65	538.20
Blue sprocket	3	51.17	45.03	4a	153.51	135.09

Test case 3

Inputs

Order

ID
5678

date
2018-01-25

customer

name
Bruce

ID
1234

role
Gold

name	quantity
Red widget	10
White widget	6
Blue trinket	150
White trinket	10
Red sprocket	13
Blue sprocket	3

item name	item quantity	regular price	discounted price	discountID	regular total	discounted total
Red widget	10	19.95	16.96	5a	199.50	169.60
White widget	6	14.95	12.71	5a	89.70	76.26
Blue trinket	150	1.28	1.15	4b	192.00	172.50
White trinket	10	2.05	1.50	3a	20.50	15.00
Red sprocket	13	47.05	39.99	5a	611.65	519.87
Blue sprocket	3	51.17	43.49	5a	153.51	130.47

Test case 4

Inputs

Order

ID
5678

date
2018-01-25

customer

name
Bruce

ID
1234

role
Gold

name	quantity
Red widget	10
White widget	6
Blue trinket	50
White trinket	10
Red sprocket	13
Blue sprocket	3

item name	item quantity	regular price	discounted price	discountID	regular total	discounted total
Red widget	10	19.95	18.15	1b	199.50	181.50
White widget	6	14.95	13.60	1b	89.70	81.60
Blue trinket	50	1.28	1.15	4b	64.00	57.50
White trinket	10	2.05	1.50	3a	20.50	15.00
Red sprocket	13	47.05	41.40	4a	611.65	538.20
Blue sprocket	3	51.17	45.03	4a	153.51	135.09