# Discount

折扣可以应用到Category, Manufacturer, Product中

一个Discount 有多个DiscountRequirement （ICollection<DiscountRequirement> DiscountRequirements）

/// Gets or sets the maximum product quantity which could be discounted

/// Used with "Assigned to products" or "Assigned to categories" type

int? MaximumDiscountedQuantity

ICollection<Category> AppliedToCategories

ICollection<Manufacturer> AppliedToManufacturers

ICollection<Product> AppliedToProducts

**UsePercentage** 为true时 decimal DiscountPercentage 使用百分比算折扣, 并且使用MaximumDiscountAmount限制最大折扣

否则 DiscountAmount 使用具体数字来折扣

DiscountTypeId ： 参考 Nop.Core.Domain.Discounts.DiscountType

IsCumulative：折扣是否可以与其它DiscountTypeId相同的折扣一起使用

public enum DiscountType

{

/// <summary>

/// Assigned to order total

/// </summary>

AssignedToOrderTotal = 1,

/// <summary>

/// Assigned to products (SKUs)

/// </summary>

AssignedToSkus = 2,

/// <summary>

/// Assigned to categories (all products in a category)

/// </summary>

AssignedToCategories = 5,

/// <summary>

/// Assigned to manufacturers (all products of a manufacturer)

/// </summary>

AssignedToManufacturers = 6,

/// <summary>

/// Assigned to shipping

/// </summary>

AssignedToShipping = 10,

/// <summary>

/// Assigned to order subtotal

/// </summary>

AssignedToOrderSubTotal = 20,

}

DiscountLimitationId： 0 为不限制次数，15 => N Times Only 25 =>N Times Per Customer

LimitationTimes ： 使用次数限制 （used when Limitation is set to "N Times Only" or "N Times Per Customer")

public enum DiscountLimitationType

{

/// <summary>

/// None

/// </summary>

Unlimited = 0,

/// <summary>

/// N Times Only

/// </summary>

NTimesOnly = 15,

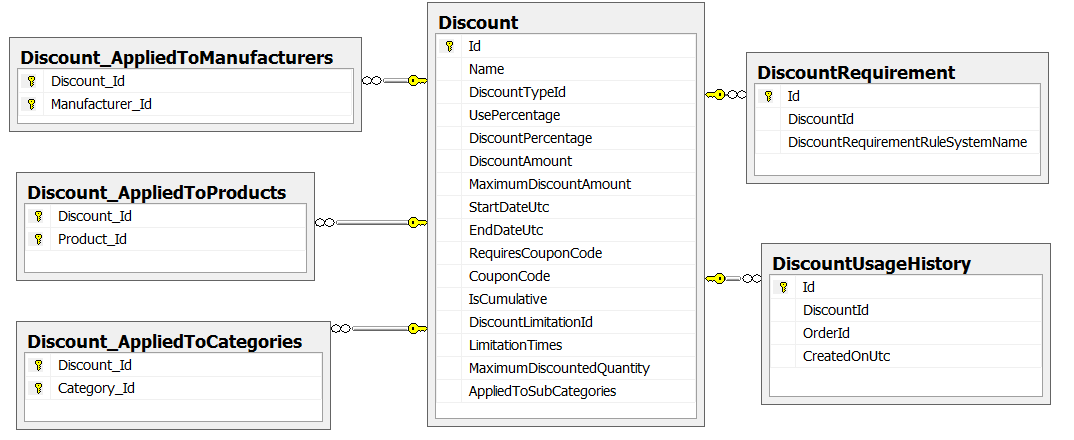
/// <summary>

/// N Times Per Customer

/// </summary>

NTimesPerCustomer = 25,

}



# Porduct

## 几个概念：

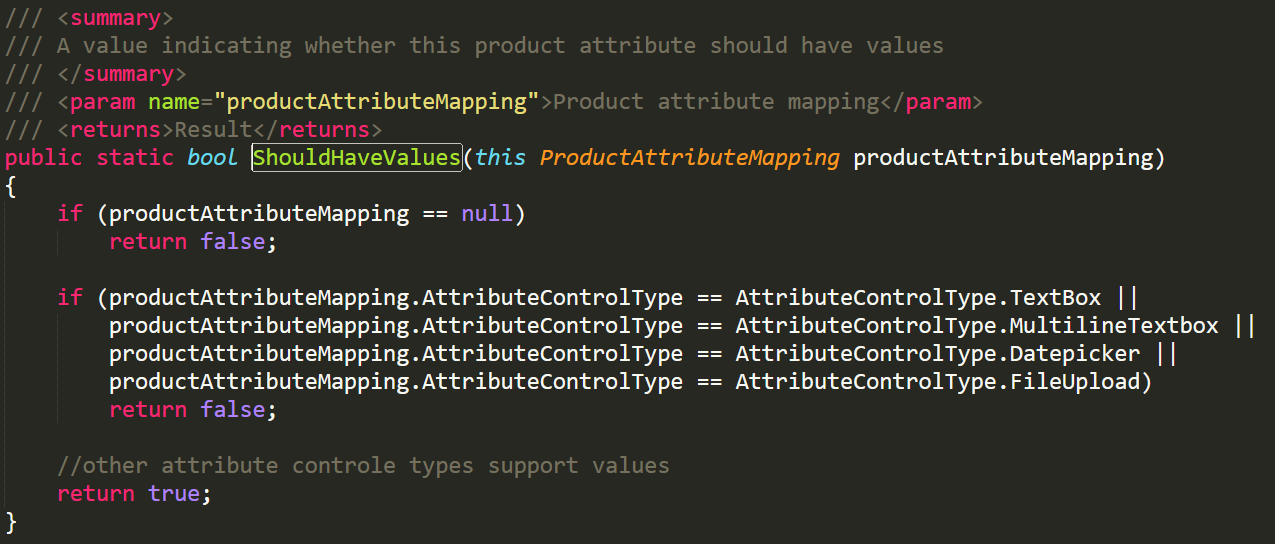
### 不可以合并的属性：non-combinable attributes

var attributes = \_productAttributeService.GetProductAttributeMappingsByProductId(product.Id)

//ignore non-combinable attributes for combinations

.Where(x => !x.**IsNonCombinable**())

.ToList();

需要用户输入的属性, 比TextBox MultilineTextbox Datepicker FileUpload 

MultilineTextbox = 10,public enum AttributeControlType

Datepicker = 20,{

FileUpload = 30, DropdownList = 1,

RadioList = 2,

Checkboxes = 3,

TextBox = 4,

MultilineTextbox = 10,

Datepicker = 20,

FileUpload = 30,

ColorSquares = 40,

ImageSquares = 45,

ReadonlyCheckboxes = 50,

}

## Product的navigation property

* ICollection<ProductCategory> ProductCategories

ProductId, CategoryId, IsFeaturedProduct, DisplayOrder

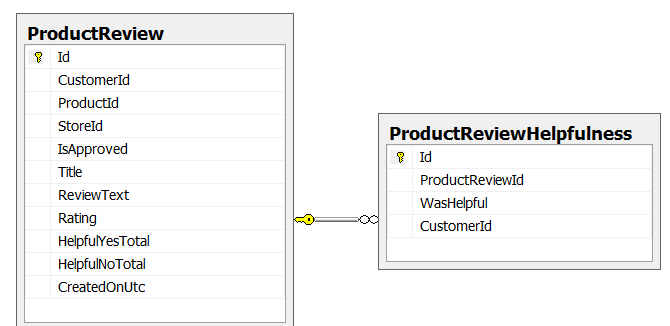
* ICollection<ProductManufacturer> ProductManufacturers

ProductId, ManufacturerId, IsFeaturedProduct, DisplayOrder

* ICollection<ProductPicture> ProductPictures

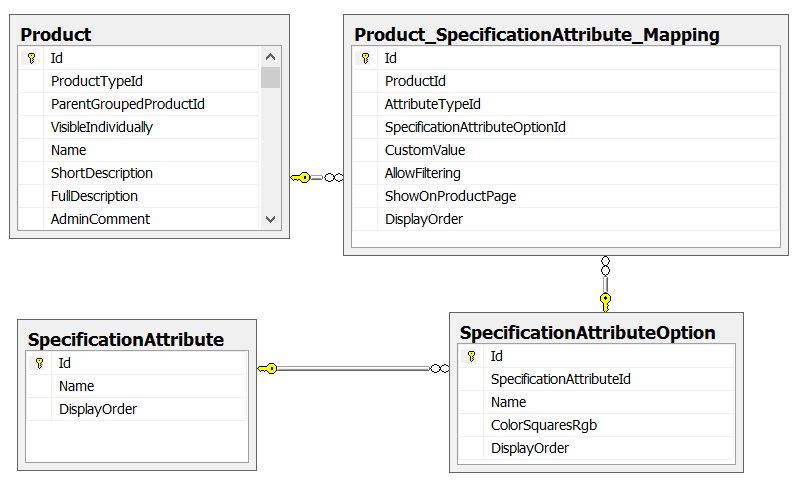
ProductId, PictureId, DisplayOrder

* ICollection<ProductReview> ProductReviews



* ICollection<ProductSpecificationAttribute> ProductSpecificationAttributes

ProductId



public enum SpecificationAttributeType

{

/// <summary>

/// Option

/// </summary>

Option = 0,

/// <summary>

/// Custom text

/// </summary>

CustomText = 10,

/// <summary>

/// Custom HTML text

/// </summary>

CustomHtmlText = 20,

/// <summary>

/// Hyperlink

/// </summary>

Hyperlink = 30

}

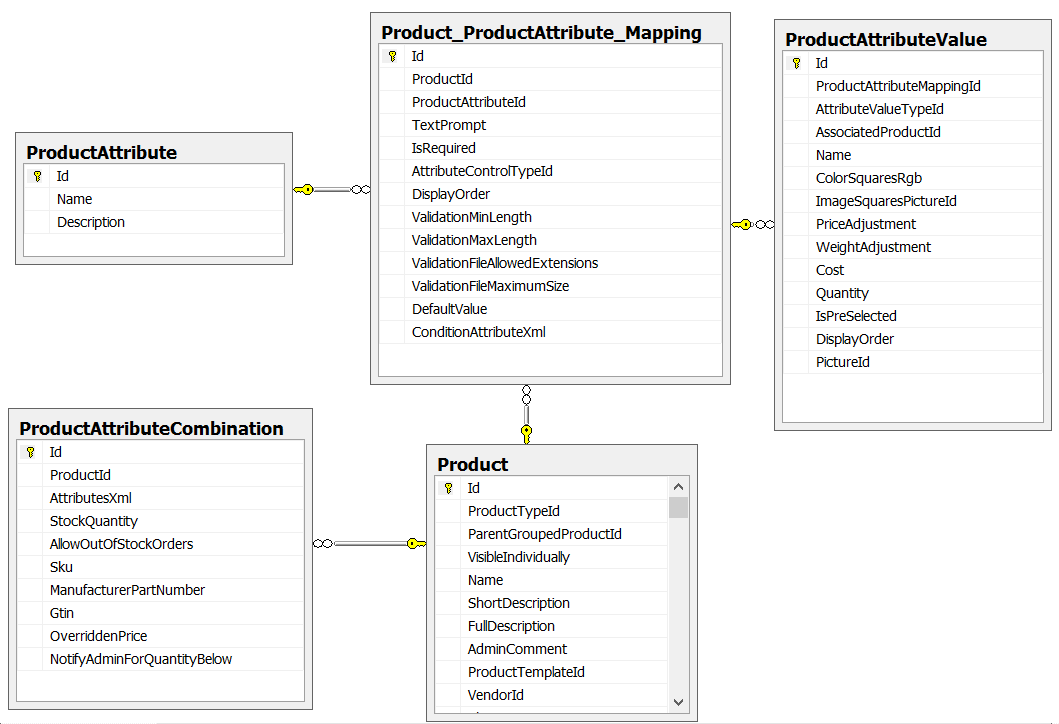
* ICollection<ProductTag> ProductTags

ProductTab只有Name 属性 （apparel， awesome 。。。 ）

多对多关系： Product\_ProductTag\_Mapping

* 属性： ICollection<ProductAttributeMapping> ProductAttributeMappings

部分属性： ProductId， ProductAttributeId， AttributeControlTypeId， ICollection<ProductAttributeValue> ProductAttributeValues



public enum AttributeControlType

{

DropdownList = 1,

RadioList = 2,

Checkboxes = 3,

TextBox = 4,

MultilineTextbox = 10,

Datepicker = 20,

FileUpload = 30,

ColorSquares = 40,

ImageSquares = 45,

ReadonlyCheckboxes = 50,

}

* ICollection<ProductAttributeCombination> ProductAttributeCombinations

ProductId，AttributesXml， StockQuantity， AllowOutOfStockOrders， Sku， ManufacturerPartNumber， Gtin（Global Trade Item Number）， decimal？ OverriddenPrice， NotifyAdminForQuantityBelow

* ICollection<TierPrice> TierPrices : 买一定数量的价格

ProductId，StoreId， int? CustomerRoleId, **Quantity**, Price

* ICollection<Discount> AppliedDiscounts
* ICollection<ProductWarehouseInventory> ProductWarehouseInventory

We use it only when "UseMultipleWarehouses" is set to "true" and ManageInventoryMethod" to "ManageStock"

Warehouse : Name, AdminComment, AddressId

ProductWarehouseInventory: ProductId, WarehouseId, StockQuantity, ReservedQuantity

## ProductAttribute 相关:

一个ProductAttribute 有多个Product\_ProductAttribute\_Mapping

一个Product\_ProductAttribute\_Mapping有多个ProductAttributeValue

一个Product有多个Product\_ProductAttribute\_Mapping

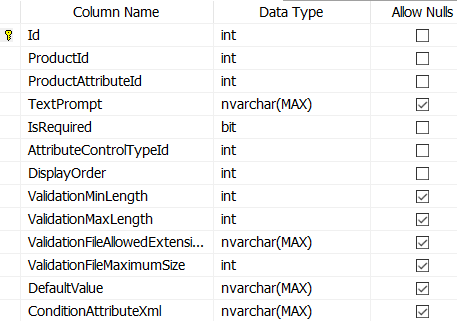
一个Product有多个ProductAttributeCombination

例如 ProductAttribute有属性 Color

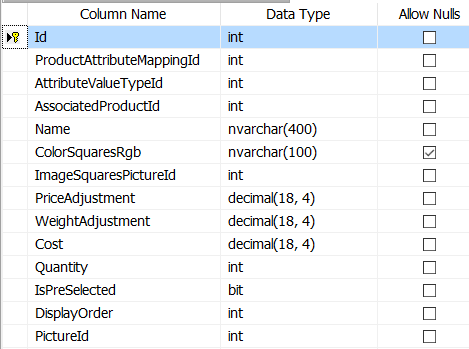
当前Product可以有多个Color属性 （Product\_ProductAttribute\_Mapping），每个Product具体的Color (蓝色， 红色) 的具体信息保存在ProductAttributeValue

**ProductAttribute** ： Name, Description

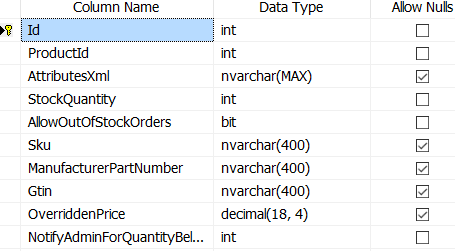
**Product\_ProductAttribute\_Mapping**



**ProductAttributeValue**



**ProductAttributeCombination**



## Product 用的 enum

public enum ProductType

{

/// <summary>

/// Simple

/// </summary>

SimpleProduct = 5,

/// <summary>

/// Grouped (product with variants)

/// </summary>

GroupedProduct = 10,

}

public enum BackorderMode

{

/// <summary>

/// No backorders

/// </summary>

NoBackorders = 0,

/// <summary>

/// Allow qty below 0

/// </summary>

AllowQtyBelow0 = 1,

/// <summary>

/// Allow qty below 0 and notify customer

/// </summary>

AllowQtyBelow0AndNotifyCustomer = 2,

}

public enum DownloadActivationType

{

/// <summary>

/// When order is paid

/// </summary>

WhenOrderIsPaid = 1,

/// <summary>

/// Manually

/// </summary>

Manually = 10,

}

public enum GiftCardType

{

/// <summary>

/// Virtual

/// </summary>

Virtual = 0,

/// <summary>

/// Physical

/// </summary>

Physical = 1,

}

public enum LowStockActivity

{

/// <summary>

/// Nothing

/// </summary>

Nothing = 0,

/// <summary>

/// Disable buy button

/// </summary>

DisableBuyButton = 1,

/// <summary>

/// Unpublish

/// </summary>

Unpublish = 2,

}

public enum ManageInventoryMethod

{

/// <summary>

/// Don't track inventory for product

/// </summary>

DontManageStock = 0,

/// <summary>

/// Track inventory for product

/// </summary>

ManageStock = 1,

/// <summary>

/// Track inventory for product by product attributes

/// </summary>

ManageStockByAttributes = 2,

}

public enum RecurringProductCyclePeriod

{

/// <summary>

/// Days

/// </summary>

Days = 0,

/// <summary>

/// Weeks

/// </summary>

Weeks = 10,

/// <summary>

/// Months

/// </summary>

Months = 20,

/// <summary>

/// Years

/// </summary>

Years = 30,

}

public enum RentalPricePeriod

{

/// <summary>

/// Days

/// </summary>

Days = 0,

/// <summary>

/// Weeks

/// </summary>

Weeks = 10,

/// <summary>

/// Months

/// </summary>

Months = 20,

/// <summary>

/// Years

/// </summary>

Years = 30,

}

# AclService StoreMappingService

AclService

\_aclService.GetCustomerRoleIdsWithAccess(product)

public virtual bool Authorize<T>(T entity, Customer customer) where T : BaseEntity, IAclSupported

{

if (entity == null)

return false;

if (customer == null)

return false;

if (\_catalogSettings.IgnoreAcl)

return true;

if (!entity.SubjectToAcl)

return true;

foreach (var role1 in customer.CustomerRoles.Where(cr => cr.Active))

foreach (var role2Id in GetCustomerRoleIdsWithAccess(entity))

if (role1.Id == role2Id)

//yes, we have such permission

return true;

//no permission found

return false;

}

public virtual int[] GetCustomerRoleIdsWithAccess<T>(T entity) where T : BaseEntity, IAclSupported

{

if (entity == null)

throw new ArgumentNullException("entity");

int entityId = entity.Id;

string entityName = typeof(T).Name;

string key = string.Format(ACLRECORD\_BY\_ENTITYID\_NAME\_KEY, entityId, entityName);

return \_cacheManager.Get(key, () =>

{

var query = from ur in \_aclRecordRepository.Table

where ur.EntityId == entityId &&

ur.EntityName == entityName

select ur.CustomerRoleId;

return query.ToArray();

});

}

StoreMapping

public virtual bool Authorize<T>(T entity, int storeId) where T : BaseEntity, IStoreMappingSupported

{

if (entity == null)

return false;

if (storeId == 0)

//return true if no store specified/found

return true;

if (\_catalogSettings.IgnoreStoreLimitations)

return true;

if (!entity.LimitedToStores)

return true;

foreach (var storeIdWithAccess in GetStoresIdsWithAccess(entity))

if (storeId == storeIdWithAccess)

//yes, we have such permission

return true;

//no permission found

return false;

}

public virtual int[] GetStoresIdsWithAccess<T>(T entity) where T : BaseEntity, IStoreMappingSupported

{

if (entity == null)

throw new ArgumentNullException("entity");

int entityId = entity.Id;

string entityName = typeof(T).Name;

string key = string.Format(STOREMAPPING\_BY\_ENTITYID\_NAME\_KEY, entityId, entityName);

return \_cacheManager.Get(key, () =>

{

var query = from sm in \_storeMappingRepository.Table

where sm.EntityId == entityId &&

sm.EntityName == entityName

select sm.StoreId;

return query.ToArray();

});

}

验证权限取得ProductCategory：

其中验证acl部分的linq为 (删去了where语句, 关联Category表为了!c.SubjectToAcl, 关联AclRecords 查看 CustomerRoleId)

from pc in Product\_Category\_Mappings

join c in Categories on pc.CategoryId equals c.Id

join acl in AclRecords

on new {c1=c.Id, c2="Category"} equals new {c1=acl.EntityId, c2=acl.EntityName}

into c\_acl

from acl in c\_acl.DefaultIfEmpty()

select pc

**LEFT OUTER JOIN格式: into, DefaultIfEmpty()**

into c\_acl

from acl in c\_acl.DefaultIfEmpty()

select pc

对应的sql为

==》

DECLARE @p0 NVarChar(1000) = 'Category'

-- EndRegion

SELECT [t0].[Id], [t0].[ProductId], [t0].[CategoryId], [t0].[IsFeaturedProduct], [t0].[DisplayOrder]

FROM [Product\_Category\_Mapping] AS [t0]

INNER JOIN [Category] AS **[t1]** ON [t0].[CategoryId] = [t1].[Id]

LEFT OUTER JOIN [AclRecord] AS [t2] ON ([t1].[Id] = [t2].[EntityId]) AND (@p0 = [t2].[EntityName])

public virtual IPagedList<ProductCategory> GetProductCategoriesByCategoryId(int categoryId,

int pageIndex = 0, int pageSize = int.MaxValue, bool showHidden = false)

{

if (categoryId == 0)

return new PagedList<ProductCategory>(new List<ProductCategory>(), pageIndex, pageSize);

string key = string.Format(PRODUCTCATEGORIES\_ALLBYCATEGORYID\_KEY, showHidden, categoryId, pageIndex, pageSize, \_workContext.CurrentCustomer.Id, \_storeContext.CurrentStore.Id);

return \_cacheManager.Get(key, () =>

{

var query = from pc in \_productCategoryRepository.Table

join p in \_productRepository.Table on pc.ProductId equals p.Id

where pc.CategoryId == categoryId &&

!p.Deleted &&

(showHidden || p.Published)

orderby pc.DisplayOrder

select pc;

if (!showHidden && (!\_catalogSettings.IgnoreAcl || !\_catalogSettings.IgnoreStoreLimitations))

{

if (!\_catalogSettings.IgnoreAcl)

{

//ACL (access control list)

var allowedCustomerRolesIds = \_workContext.CurrentCustomer.GetCustomerRoleIds();

query = from pc in query

join c in \_categoryRepository.Table on pc.CategoryId equals c.Id

join acl in \_aclRepository.Table

on new { c1 = c.Id, c2 = "Category" } equals new { c1 = acl.EntityId, c2 = acl.EntityName } into c\_acl

from acl in c\_acl.DefaultIfEmpty()

where !c.SubjectToAcl || allowedCustomerRolesIds.Contains(acl.CustomerRoleId)

select pc;

}

if (!\_catalogSettings.IgnoreStoreLimitations)

{

//Store mapping

var currentStoreId = \_storeContext.CurrentStore.Id;

query = from pc in query

join c in \_categoryRepository.Table on pc.CategoryId equals c.Id

join sm in \_storeMappingRepository.Table

on new { c1 = c.Id, c2 = "Category" } equals new { c1 = sm.EntityId, c2 = sm.EntityName } into c\_sm

from sm in c\_sm.DefaultIfEmpty()

where !c.LimitedToStores || currentStoreId == sm.StoreId

select pc;

}

//only distinct categories (group by ID)

query = from c in query

group c by c.Id

into cGroup

orderby cGroup.Key

select cGroup.FirstOrDefault();

query = query.OrderBy(pc => pc.DisplayOrder);

}

var productCategories = new PagedList<ProductCategory>(query, pageIndex, pageSize);

return productCategories;

});

}