Цели при проектировании БД

1. Максимальная нормализация БД для закрепления навыков.
2. Функциональное соответствие проекта моему старому магазину по охранным системам. Оринентируюсь на вопрос, как бы я сейчас делал сайт на C# с учетом полученных знаний , а не на Joomla/Virtuemart.
3. Минимизация данных, получаемых из базы. Пока покупатель не заинтересовался конкретным товаром должно включаться минимальное количество таблиц и загружаться меньший объем данных.

Описание структуры БД

1. Данные одного товара, которые могут быть показаны в каталоге, вносятся в две таблицы:

а) **dbo.Products** - содержит параметры общие для всех цветов, первичный ключ ProductId.

б) **dbo.ColoredProducts** – содержит парамеры, зависимые от цвета товара, первичный ключ составной ProductId + ColorId.

Таблица **dbo.ColoredProducts** содержит поле с уменьшенной первой картинкой товара из основной таблицы изображений dbo.Images для быстрого показа в каталоге.

1. Таблица **dbo.Images** содержит по нескольку картинок для каждого первичного ключа табл. **dbo.ColoredProducts**
2. Таблица **dbo.Colors** предоставляет Id и название цвета для каждого товара из **dbo.ColoredProducts**.
3. Каждый товар из табл. Products содержит отдельный набор характеристик, специфичный для данного типа (категории) товара. Поэтому созданы 3 таблицы :

а) со списком категорий - **dbo.Categories**,

б) со списком характеристик – **dbo.Characteristics**,

в) с соответствием набора характеристик конкретной категории - **dbo.CategoryCharacteristics**, чтобы каждому товару могли присваиваться только те характеристики, которые вообще могут быть у данного товара.

1. Таблица **dbo. CharacteristicValues** – это список соответствий, какие вообще значения могут быть у характеристики в категории. Эти значения должны появляться в выпадающем списке при заполнении характеристик администратором. Такая функция будет очень удобна при заполнении большого количества товаров потому что значений именно для характеристики в категории немного и их не придется печатать как правило. Имена значений вынесены в отдельную табл. **dbo.Values**.

Можно дополнительно акцентировать, что задаются значения характеристик именно в категории (связь таблиц **dbo. CharacteristicValues** - **dbo.CharacteristicsInCategory**), а не просто для всех характеристик. Таким образом:

а) Уменьшается количество значений в выпадающем списке

б) Мы избегаем появления возможных значений для характеристики, которые синтаксически похожи на правильные, но недопустимы. Например, характеристике “Размер матрицы” могут в разделе “Домофоны” соответствовать одни размеры в сантиметрах, а в разделе “Видеокамеры” – другие размеры в дюймах. Или, например, характеристики “Горизонтальный угол обзора ” и “Вертикальный угол обзора” имеют похожие стандартные значения в категориях “Датчики движения” и “Видеокамеры” и их будет трудно различать, если смешивать значения в категориях.

1. Таблица **dbo.ProductValues** – это по сути результат выбора администратором строк из табл. **dbo. CharacteristicValues** для конкретного товара (ключ товара- только ProductId т. к. цвет – не индивидуальный параметр). И это фиксирование конкретных индивидуальных значений для каждого товара.
2. Работа с заказами. При наполнении корзины заказанные товары складываются в сессию. В момент подтверждения заказа они отправляются в таблицу **dbo.Orders** с общим значением OrderId.

После выполнения заказа записи могут оставаться со статусом “Выполнено” для сбора статистики.

1. Установка индексов. Абсолютно точно нужно проконтроллировать индексы на столбцах, по которым часто будет идти поиск и которые нечасто изменяются. Это столбцы характеристик CharacteristicName и их значений Value, соответственно принадлежащие таблицам **dbo.Characteristics** и **dbo.Values**. Они сразу обозначены, как UNIQUE, поэтому, для них уже по умолчанию созданы уникальные некластерные индексы.