


[АНАЛИЗ И ПРОЕКТИРОВАНИЕ СИСТЕМ*, SQL*](#)

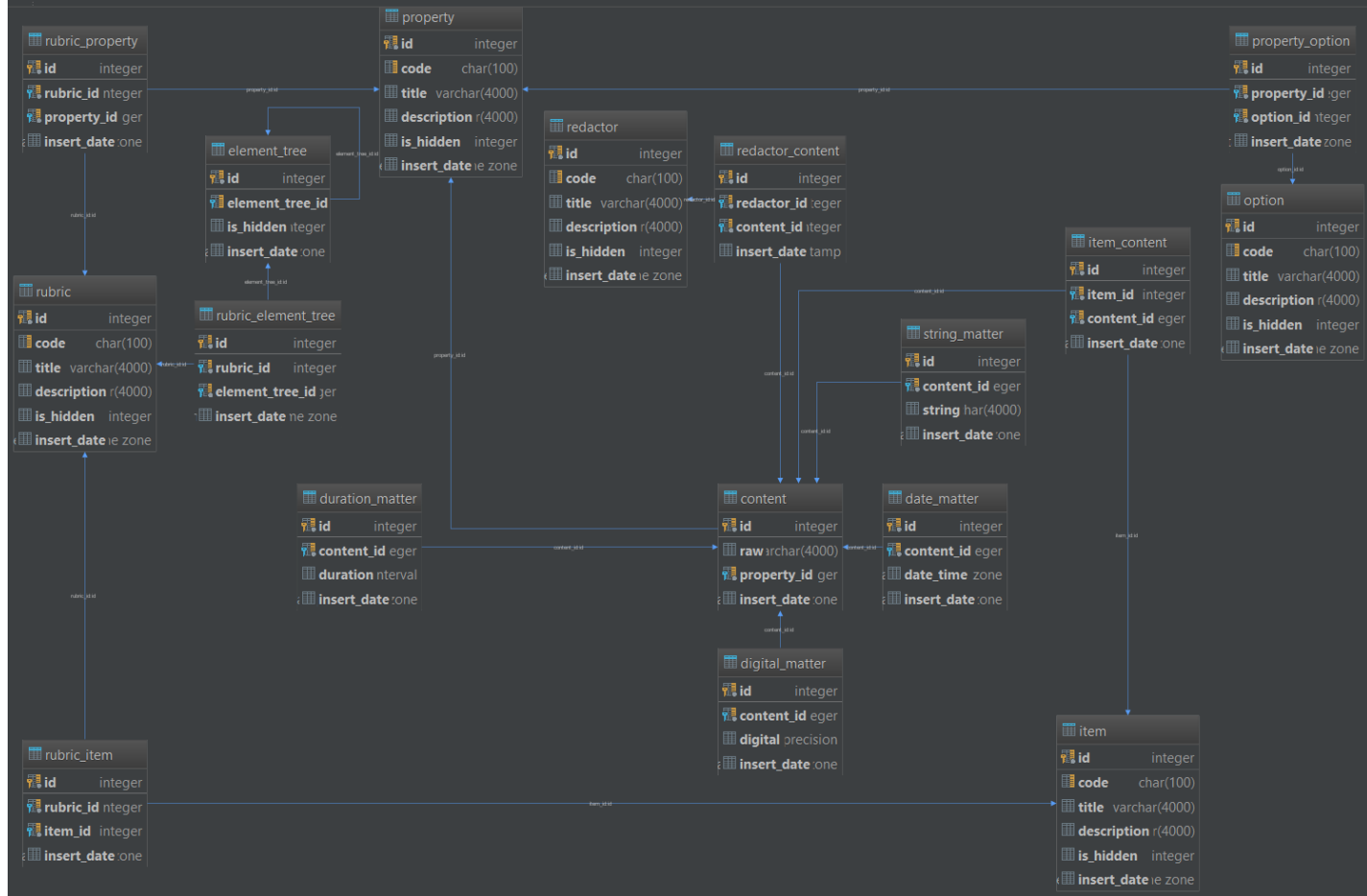
Пилим каталог товаров не притрагиваясь к реляционной алгебре

vintage 15 марта 2017 в 15:07  11,9k

Здравствуйте, меня зовут Дмитрий Карловский и я... давно не занимался бэкендом, но на днях вдруг наткнулся на мучения  [SbWereWolf](#) по [натягиванию ужа на ежа](#) и не смог удержаться от соблазна сдуть пыль со своего мульти-инструмента [OrientDB](#) да оттяпать им чего-нибудь этакого.

Итак, мастерить мы сегодня будем базу данных для интернет-магазина с поиском товаров по параметрам, полнотекстовым поиском, локализацией, автоматическим формированием рубрикатора и мастера добавления товара.

Разбирать мы будем вот этот вот реляционный звездолёт:



А собирать вот такой вот графовый скворечник:

- **slug** — человекопонятный идентификатор
- **created** — время создания сущности
- **searchable** — будет ли сущность находиться при поиске
- **title** — человекопонятное название сущности в виде словаря "язык-текст"
- **description** — человекопонятное описание сущности в виде словаря "язык-текст"

Чтобы не повторять эти поля в каждой сущности, как это сделано в звездолёте, мы просто создадим абстрактный класс "Object" от которого в дальнейшем будем наследовать остальные сущности:

```
Create class Object abstract

Create property Object.slug string ( collate ci , notnull true )
Create property Object.created datetime ( readonly true , default sysdate() )
Create property Object.searchable boolean ( default false )

Create property Object.title embeddedmap string ( collate ci )
Create property Object.description embeddedmap string ( collate ci )
```

slug должен быть уникальным для каждой сущности, а по названиям и описаниям нам потребуется поиск, поэтому добавим на эти поля соответствующие индексы:

```
Create index Object.slug unique

Create index Object.title on Object( title by value ) fulltext
engine lucene metadata {
  "analyzer" : "org.apache.lucene.analysis.ru.RussianAnalyzer"
}

Create index Object.description on Object( description by value )
fulltext
engine lucene metadata {
  "analyzer" : "org.apache.lucene.analysis.ru.RussianAnalyzer"
}
```

Славно, что движок полнотекстового поиска уже встроен в СУБД и нам не надо заниматься ручным перекладыванием данных из своей основной базы данных в какой-нибудь [ElasticSearch](#), построенный на том же [Lucene](#). При изменении сущности СУБД сама позаботится об актуализации полнотекстового индекса.

Самое главное в нашем магазине — товары, а каждый товар помимо стандартных свойств имеет ещё и цену:

```
Create class Product extends Object  
  
Create property Product.price decimal
```

В звездолёте используется иерархия рубрик в виде дерева, но мы поступим интересней — всю иерархию мы разобьём на 2 типа узлов:

- **Тег** — флаг, характеризующий конкретный товар.
- **Аспект** — группа взаимоисключающих тегов, образующая характеристику товаров

В иерархии эти два типа узлов идут попеременно:

- внутри тега могут находиться аспекты, но не теги
- внутри аспекта могут находиться теги, но не аспекты

Примеры аспектов (а в скобках — тегов):

- Вид товара (еда, одежда, техника)
- Производитель (Плюшкин Инкорпорейтед, Макинтош Лимитед, Экскаватор Трейдинг)
- Цвет (Красный, Синий, Зелёный)

При этом выбор "цвета" имеет смысл давать только для тегов "одежда" и "техника", но не "еда".

Добавим эти две сущности и провяжем их друг с другом:

```
Create class Aspect extends Object
Create class Tag extends Object

Create property Aspect.tag linkset Tag
Create property Aspect.tag_sub linkset Tag

Create property Tag.aspect linkset Aspect
Create property Tag.aspect_sub linkset Aspect
```

Как видно, все связи у нас двусторонние "многие-ко-многим". Каждая сущность имеет ссылку на связанные с нею сущности. Осталось лишь связать нашу иерархию с товарами. Связь будет односторонней, чтобы не захламлять тег списком связанных с ним товаров, которым мы всё-равно не будем пользоваться:

```
Create property Product.tag linkset Tag
```

Например, для товара "Скатерть-самобранка" может быть выставлен тег "техника", который откроет нам аспект "цвет" и, как следствие, возможность выбрать "красный".

Чтобы поиск по тегам у нас не тормозил, мы добавим по ним индекс:

```
Create index Product.tag notunique
```

Кроме флагов, у сущностей должны быть и атрибуты других типов: строковые, целочисленные, десятичные, временные и тд. Чтобы описать эти атрибуты, введём соответствующую сущность:

```
Create class Attribute extends Object  
  
Create property Attribute.type string ( default "string" )
```

Атрибуты у нас будут привязываться не к товарам, как можно было бы подумать, а к тегам:

```
Create property Tag.attribute linkset Attribute  
Create property Attribute.tag linkset Tag
```

Например, если установлен тег "еда", то для товара становится доступен атрибут "срок годности". Само значение атрибута конкретного товара мы будем хранить в самом же товаре. В схему мы это выносить не будем, так как у каждого товара фактически может быть свой набор атрибутов в зависимости от того, какие теги ему были установлены.

Пользовательские сценарии

Полнотекстовой поиск

Если пользователь ввёл поисковой запрос, то мы сразу ищем все объекты, которые ему соответствуют:

```
Select from Object
where searchable = true
    and ( title lucene "Ска*" or description lucene "Ска*" )
```

Но если выдача получится слишком большой, то разумно будет предложить ему детализировать запрос по тегам из товаров в выдаче. Для этого запросим вместе с собственно найденными объектами ещё и связанные с ними теги и связанные с последними атрибуты и аспекты:

```
Select from Object
where searchable = true
    and ( title lucene "Ска*" or description lucene "Ска*" )
fetchplan *:0 slug:0 title:0 tag.slug:0 tag.title:0
tag.aspect.title:0 tag.attribute.title:0 tag.attribute.type:0
```

Таким образом, рядом с поисковой выдачей мы можем разместить уточняющий рубрикатор, переход по пунктам которого сузит выдачу, но гарантированно не приведёт к её полной очистке. Поэтому мы смело добавляем фильтрацию по выбранным пользователем тегам и значениям атрибутов:

```
Select from Object
where searchable = true
    and ( title lucene "Ска*" or description lucene "Ска*" )
    and ( tag in ( Select from Tag where slug = "tag=tech" and
slug="color=red" ) )
    and ( weight between 100 and 200 )
fetchplan *:0 slug:0 title:0 tag.slug:0 tag.title:0
tag.aspect.title:0 tag.attribute.title:0 tag.attribute.type:0
```

Навигация по каталогу

Когда пользователь только открыл сайт, разумно сразу предложить ему несколько направлений движения и очертить ассортимент. Поэтому мы выведем все корневые аспекты и возможные для них теги:

```
Select from Aspect
where ( tag is null )
fetchplan *:0 title:0 tag_sub.slug:0 tag_sub.title:0
```

Дальнейшее путешествие пользователя аналогично случаю полнотекстового поиска, но без собственно полнотекстового поиска:

```
Select from Product
where searchable = true
      and ( tag in ( Select from Tag where slug = "tag=tech" and
slug="color=red" ) )
      and ( weight between 100 and 200 )
fetchplan *:0 slug:0 title:0 tag.slug:0 tag.title:0
tag.aspect.title:0 tag.attribute.title:0 tag.attribute.type:0
```

Как можно заметить, погружаясь всё глубже в кроличью нору, мы не переходим с одной рубрики на другую (вложенную), а добавляем в фильтрацию дополнительный тег. Например, на странице "красные ботинки" мы будем искать по тегам "одежда", "обувь", "ботинки", "красные", а на странице "красненькие ноутбуки" — "техника", "компьютеры", "ноутбуки", "красные". В обоих случаях "красные" — это один и тот же тег.

Создание товара

При создании товара, нет никакого смысла выводить для заполнения все возможные для товаров параметры. Например, параметр "соотношение сигнал/шум" совершенно бессмысленно для "ботинок". Поэтому точно так же, как с каталогом, мы выводим лишь корневые аспекты, а дополнительные аспекты становятся доступными лишь по мере выбора тегов пользователем, добавляющим товар. Список доступных атрибутов и аспектов с их тегами по списку выбранных тегов получается достаточно тривиально:

```
Select from Aspect
where ( tag is null )
      or ( tag in ( Select from Tag where slug = "tag=tech" and
slug="color=red" ) )
fetchplan *:0 title:0 tag_sub.slug:0 tag_sub.title:0
tag_sub.attribute.title:0 tag_sub.attribute.type:0
```

Тут же мы можем предоставить пользователю возможность добавлять не только товар, но и расширять набор атрибутов, аспектов и тегов.

Создадим, например, аспект "Вид товара":

```
Insert into Aspect set slug = "aspect=kind" , title = { "ru" : "Вид
товара" }
```

Теперь добавим к нему, например, тег "Одежда":

```
Insert into Tag
set
    slug = "tag=wear" ,
    searchable = true ,
```

```
title = { "ru" : "Одежда" } ,  
aspect = ( Select from Aspect where slug = "aspect=kind" )
```

Update Aspect

```
add tag_sub = ( Select from tag where slug = "tag=wear" )  
where slug = "aspect=kind"
```

Другие теги добавляются аналогично. Добавление вложенного в тег аспекта аналогично. Например, добавим аспект "Цвет" в теги "Одежда" и "Техника":

Insert into Aspect

set

```
slug = "aspect=color" ,  
title = { "ru" : "Цвет" } ,  
tag = ( Select from Tag where slug = "tag=wear" or slug =  
"tag=tech" )
```

Update Tag

```
add aspect_sub = ( Select from Aspect where slug = "aspect=color" )  
where slug = "tag=wear"
```

Update Tag

```
add aspect_sub = ( Select from Aspect where slug = "aspect=color" )  
where slug = "tag=tech"
```

Ну и, наконец, самое главное — добавление товара. Для примера, добавим "Скатерть-самобранку" красного цвета:

Insert into Product

set

```
slug = "product=2" ,  
searchable = true ,  
title = { "ru" : "Скатерть-самобранка" } ,  
price = 999 ,  
tag = ( Select from Tag where slug = "tag=tech" or slug =  
"tag=red" )
```

Удаление товара

Удаление товара вовсе не должно приводить к удалению записи об этом товаре из базы данных, так в дальнейшем может потребоваться восстановить этот товар или найти данные о нём по идентификатору из какого-нибудь лога. Да даже чтобы выдавать 410(Gone) вместо 404(Not found) нужно, чтобы какая-то запись о товаре всё же оставалась. Кроме того есть такая сложная проблема как обеспечение того, чтобы никакая другая запись не ссылалась на удаляемую. Поэтому лучшее решение — изменять запись так, чтобы она исключалась из определённых процессов. Например, чтобы товар не находился ни в глобальном поиске, ни в каталоге, достаточно изменить флаг `searchable` на `false`. Именно поэтому во всех поисковых запросах мы указывали дополнительное условие `where searchable = true`.

```
Update Product set searchable = false where slug = "product=2"
```

Другой вариант "удаления" — удаление ссылок на сущность, вместо удаления самой сущности. Например, список тегов аспекта у нас хранится в свойстве `tag_sub`. Если мы хотим, чтобы больше нельзя было выбирать тег "Серобуромолиновый" в аспекте "Цвет", то просто удаляем его из `tag_sub`, но связь от тега к аспекту оставляем нетронутой. Таким образом, при просмотре товара с этим странным цветом ничего не сломается — будет показываться "Цвет: Серобуромалиновый", но при создании нового товара выбрать этот цвет будет невозможно.

```
Update Aspect  
remove tag_sub = ( Select from Tag where slug = "tag=gray-brown-  
magenta" )  
where slug = "aspect=color"
```

Резюме

Итого, у нас получилось 4 сущности: товар, тег, аспект и атрибут. Между собой они имеют 8 типов связей. И всего этого достаточно, чтобы реализовать свой Яндекс.Маркет с поиском, фильтрами и волшебницами всего за один беспокойный вечер.

Проголосовать:



+16



Поделиться:



Сохранить:



Комментарии (35)

Похожие публикации

«Яндекс.Маркет»: фотоаппараты подорожали на 20%, пылесосы — на 10%, а ноутбуки после декабря 2014 года не дешевели

7

Технологии «Яндекса» для анализа больших данных сэкономят Магнитогорскому металлургическому комбинату \$3–9 млн в год

5

semen_grinshtein • 19 августа 2015 в 14:44

Облачно, возможны базы данных по требованию

4

warlog • 28 июля 2015 в 11:00

Популярное за сутки

Яндекс открывает Алису для всех разработчиков. Платформа Яндекс.Диалоги (бета)

69

BarakAdama • вчера в 10:52

Почему следует игнорировать истории основателей успешных стартапов

20

ПЕРЕВОД

m1rko • вчера в 10:44

Как получить телефон (почти) любой красоты в Москве, или интересная особенность MT_FREE

24

ИЗ ПЕСОЧНИЦЫ

sab404 • вчера в 20:27

Java и Project Reactor

10

Пользовательские агрегатные и оконные функции в PostgreSQL и Oracle

6

erogov • вчера в 12:46

Лучшее на Geektimes

Как фермеры Дикого Запада организовали телефонную сеть на колючей проволоке

31

NAGru • вчера в 10:10

Энтузиаст сделал новую материнскую плату для ThinkPad X200s

49

alizar • вчера в 15:32

Кто-то посылает секс-игрушки с Amazon незнакомцам. Amazon не знает, как их остановить

85

Pochtoycom • вчера в 13:06

Илон Маск продолжает убеждать в необходимости создания колонии людей на Марсе

139

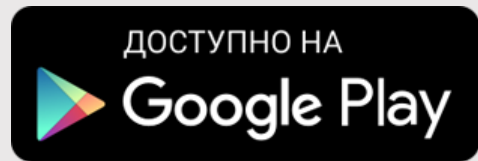
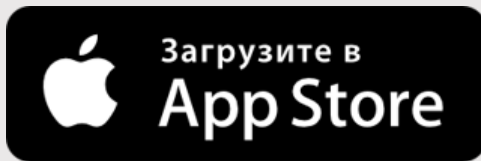
marks • вчера в 14:19

Дела шпионские (часть 1)

16

TashaFridrih • вчера в 13:16

Мобильное приложение



Полная версия

2006 – 2018 © TM