Курсовой проект по курсу «Базы данных»

# Информация:

Заведение: Geek University , факультет Искусственного интеллекта

Студент: Ершов Виталий

Преподаватель: Виктор Щупоченко

Февраль 2020 г

# 1.Описание задачи

## 1.1 Общее описание

Есть реально действующий интернет магазин (zubr-online.ru). Существует проблема с поддержанием актуальных данных в этом магазине. Необходимо обновление цен, остатков, добавлять новые, изменять существующие. Магазин реализован на «движке» opencart, использует mysql. . Есть доступ к базе данных поставщика, она реализована SQLite. Структура и сами данные сильно отличаются.

## 1.2. Конкретизация задачи.

1. Процедура проверки цен и остатков товаров в БД магазина. Необходима процедура, которая каждую товарную позицию в БД магазина, проверит в БД поставщика и сделает следующее:

1.1. Найдет данную позицию в БД поставщика. Если позиция не найдена, то установить в поле «количество» значение 0

1.2. Проверить наличие данной позиции. Если в БД поставщика указано, что товар в наличии, то установить в поле «количество» значение 100

1.3. Исходя из данных в БД поставщика пересчитать цену товара.

2. Процедура добавления нового товара.

2.1. Найти в БД поставщика товары, которых нет в БД магазина

2.2. Отобрать из этих товаров, те которые можно вносить в магазин (хватает данных)

2.3. Загрузить новые товары (которые можно загрузить автоматически)

## 1.3. Инструменты и подходы

База данных интернет магазина находиться у хостинг провайдера. Есть возможность внешнего доступа. База данных поставщика находиться на моем локальном компьютере, поскольку обновляется через приложение установленное на нем и используется в этом приложении.

Поскольку будем работать с разными СУБД, разными БД и на разных серверах буду использовать графический клиент DBeaver. Используя его инструменты буду осуществлять импорт/экспорт данных, что бы можно было сохранять файлы на локальной машине. Так же в DBeaver есть «задачи», которые удобно сохранить и использовать для постоянных операций

Общий подход следующий:

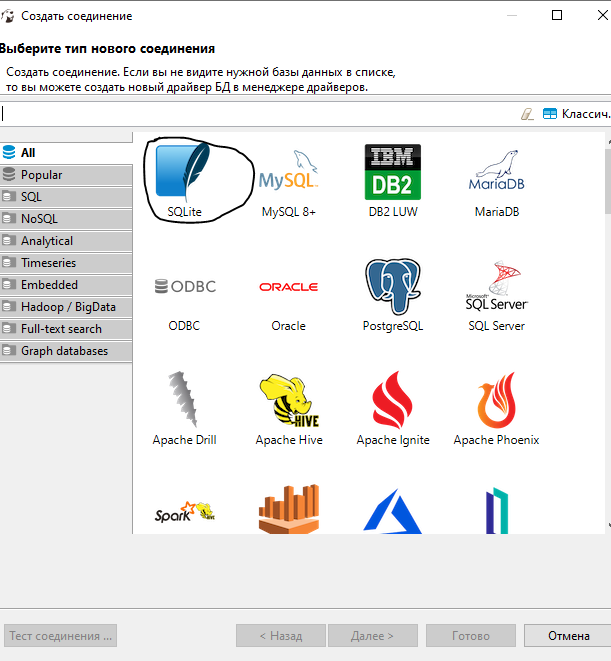
1. Подключаю DBeaver к БД поставщика
2. Делаю тестовую копию БД магазина
3. Подключаюсь к ней DBeaver
4. Изучаю БД магазина, нахожу таблицы и данные, которые нужно заполнять
5. Изучаю БД поставщика, нахожу таблицы данные которые нужны.
6. Делаю на сервере хостер провайдера еще служебную БД, в которую буду переносить нужные данные из БД поставщика. Создаю в ней нужные таблицы.
7. Организую посредствам DBeaver эк спорт данных из БД поставщика в файл CSV
8. Организую посредствам DBeaver импорт данных в служебную БД из файл CSV
9. Создание процедуры обновления товара в БД магазина
10. Создание процедуры поиска товаров, которых нет в магазине
11. Загрузка новых товаров

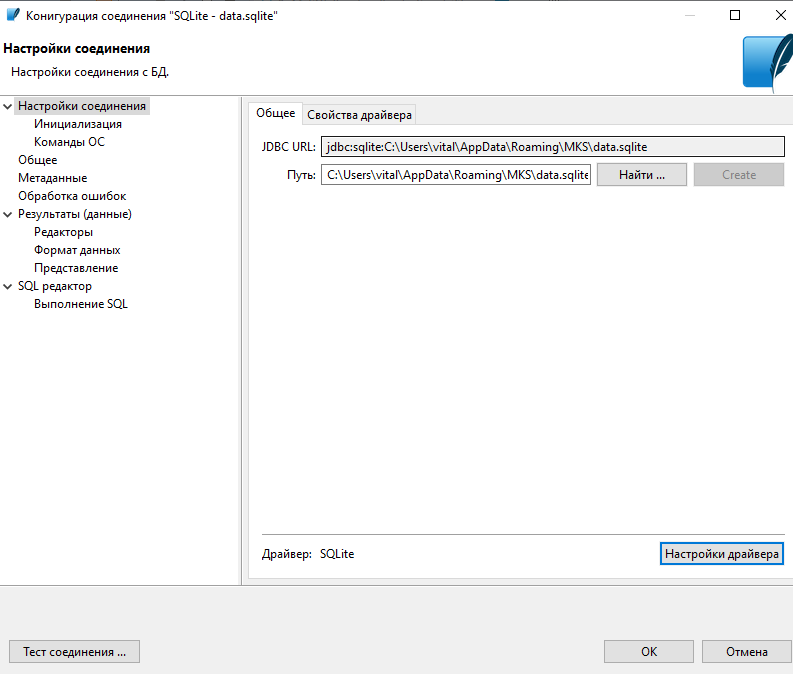
# 2. Реализация

## 2.1. Подключение к БД поставщика

База данных Поставщика реализована на SQLite.

В графическом клиенте DBeaver жмем кнопку новое соединение, выбираем подключение SQLlite и указываем путь к файлу базы данных. База появляется в интерфейсе.





## 2.2. Делаем копию БД магазина на сервере хостера.

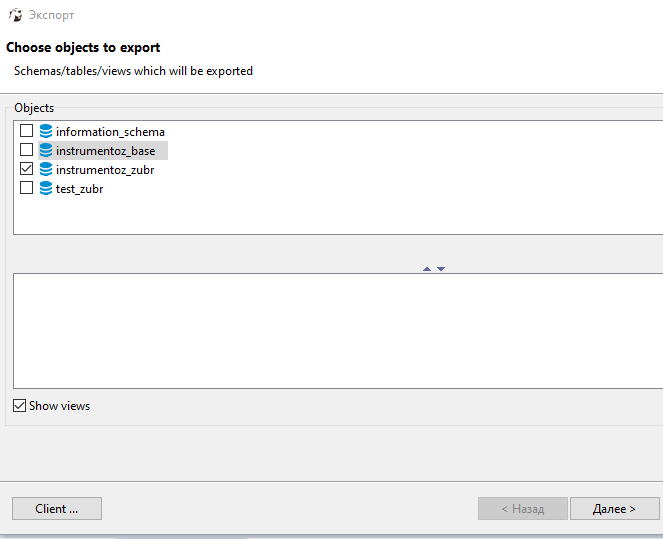
1. Подключаем DBeaver к серверу БД магазина. Для этого получаем у провайдера информацию как сделать соединение и создаем соединение к mysql через SSH

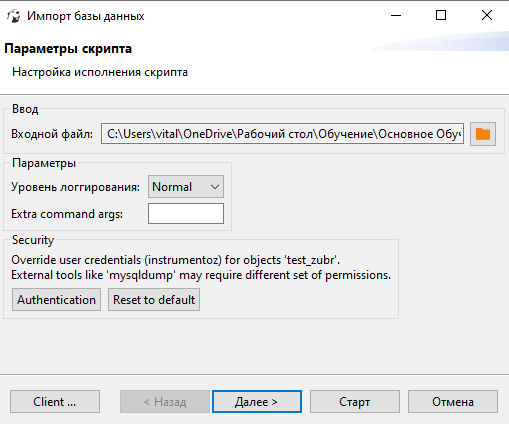
2. Создаем тесовую БД.

**CREATE DATABASE test\_zubr;**

3. Далее средствами DBeaver делаем Dump основной БД магазина (DUMP\_DB\_SHOP.rar)

4. Загружаем Dump в тестовую БД.





4. **use test\_zubr;**

## 2.3. Анализ БД Магазина

**1. show tables;**

Всего 138 таблиц. Список таблиц и схема в отдельных прилагаемых файлах(СписокТаблицМагазина.txt)

Главная таблица в которой содержится товар это таблица “product”

**SHOW CREATE TABLE product;**

CREATE TABLE `product` (

`product\_id` int(11) NOT NULL AUTO\_INCREMENT,

`model` varchar(64) NOT NULL,

`sku` varchar(64) NOT NULL,

`upc` varchar(255) NOT NULL,

`ean` varchar(255) NOT NULL,

`jan` varchar(255) NOT NULL,

`isbn` varchar(255) NOT NULL,

`mpn` varchar(255) NOT NULL,

`location` varchar(128) NOT NULL,

`quantity` int(4) NOT NULL DEFAULT '0',

`stock\_status\_id` int(11) NOT NULL,

`image` varchar(255) DEFAULT NULL,

`manufacturer\_id` int(11) NOT NULL,

`shipping` tinyint(1) NOT NULL DEFAULT '1',

`price` decimal(15,4) NOT NULL DEFAULT '0.0000',

`points` int(8) NOT NULL DEFAULT '0',

`tax\_class\_id` int(11) NOT NULL,

`date\_available` date NOT NULL DEFAULT '0000-00-00',

`weight` decimal(15,2) NOT NULL DEFAULT '0.00',

`weight\_class\_id` int(11) NOT NULL DEFAULT '0',

`length` decimal(15,2) NOT NULL DEFAULT '0.00',

`width` decimal(15,2) NOT NULL DEFAULT '0.00',

`height` decimal(15,2) NOT NULL DEFAULT '0.00',

`length\_class\_id` int(11) NOT NULL DEFAULT '0',

`subtract` tinyint(1) NOT NULL DEFAULT '1',

`minimum` int(11) NOT NULL DEFAULT '1',

`sort\_order` int(11) NOT NULL DEFAULT '0',

`status` tinyint(1) NOT NULL DEFAULT '0',

`viewed` int(5) NOT NULL DEFAULT '0',

`date\_added` datetime NOT NULL,

`date\_modified` datetime NOT NULL,

PRIMARY KEY (`product\_id`)

) ENGINE=MyISAM AUTO\_INCREMENT=22404 DEFAULT CHARSET=utf8

К сожалению в БД не прописаны внешние ключи, поэтому буду искать связи сам.

Важные для нас поля, которые нужно заполнять:

1. **model** – артикул товара. Он же уникальный идентификатор товара, через него обеспечивается связь с базой поставщика
2. **quantity** остаток товара на складе. При нуле, система считает что товара нет в наличии
3. **stock\_status\_id** – внешний ключ для таблицы stock\_status. В данной талице хряница информация, что буде считать система в случае отсутствия товара

**SHOW CREATE TABLE stock\_status;**

CREATE TABLE `stock\_status` (

`stock\_status\_id` int(11) NOT NULL AUTO\_INCREMENT,

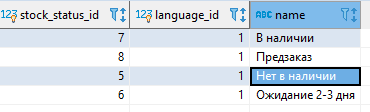
`language\_id` int(11) NOT NULL,

`name` varchar(32) NOT NULL,

PRIMARY KEY (`stock\_status\_id`,`language\_id`)

) ENGINE=MyISAM AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8

**select \* from stock\_status;**



По сути это ENUM

Мы не будем использовать эту таблицу, а в таблице product будем записывать цифру 5, которая соответствует «Нет в наличии». Это то что нам нужно.

1. **Image** в этом поле будет храниться адрес ссылки на фото товара. Данное поле мы будем генерировать по принципу **CANCAT (‘catalog/images\_small/’,model)**
2. **manufacturer\_id –** внешний ключ к таблице manufacturer

**SHOW CREATE TABLE manufacturer;**

CREATE TABLE `manufacturer` (

`manufacturer\_id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(64) NOT NULL,

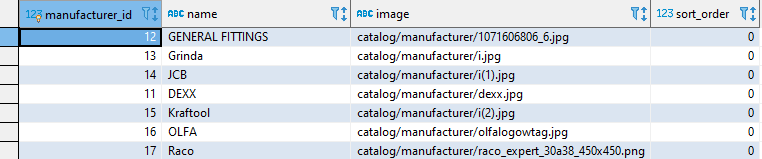
`image` varchar(255) DEFAULT NULL,

`sort\_order` int(3) NOT NULL,

PRIMARY KEY (`manufacturer\_id`)

) ENGINE=MyISAM AUTO\_INCREMENT=42 DEFAULT CHARSET=utf8

**SELECT \* FROM manufacturer;**



Данная таблица нам нужна, для загрузки данных. Так как у каждого товара может быть свой производитель, производители могу добавляться. При загрузке данных в эту таблицу мы будем заполнять только имя Производителя.

1. **Shipping**. Доставка. По сути это Булево. Ставим всегда **1.**
2. **Price**. Цена товара которая будет указана в интернет магазине. Данное поле расчетное, рассчитывается по собой логике, на основании входящей цены поставщика. По следующей формуле. Если цена до 500 р, наценка 25 %, до 1000 – 20%, до 3000 – 15%, после 10%
3. **weight\_class\_id**. Устанавливаем в этом поле **1**.
4. **length\_class\_id**. Устанавливаем в этом поле **1**
5. **minimum**. Минимальная партия товара (упаковка). Берем данные от поставщика
6. **status**. Указываем **1**
7. **date\_added.** Добавляем Now()
8. **date\_modified.** Добавляем Now()

Следующая важная таблица это **category**

**SHOW CREATE TABLE category;**

CREATE TABLE `category` (

`category\_id` int(11) NOT NULL AUTO\_INCREMENT,

`image` varchar(255) DEFAULT NULL,

`parent\_id` int(11) NOT NULL DEFAULT '0',

`top` tinyint(1) NOT NULL,

`column` int(3) NOT NULL,

`sort\_order` int(3) NOT NULL DEFAULT '0',

`status` tinyint(1) NOT NULL,

`date\_added` datetime NOT NULL,

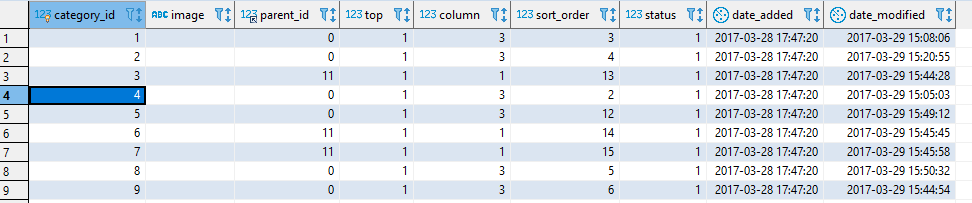
`date\_modified` datetime NOT NULL,

PRIMARY KEY (`category\_id`),

KEY `parent\_id` (`parent\_id`)

) ENGINE=MyISAM AUTO\_INCREMENT=3725 DEFAULT CHARSET=utf8

Эту таблицу так же нужно грузить и синхронизировать с поставщиком



Следующая нужная таблица category\_description

**SHOW CREATE TABLE category\_description;**

CREATE TABLE `category\_description` (

`category\_id` int(11) NOT NULL,

`language\_id` int(11) NOT NULL,

`name` varchar(255) NOT NULL,

`description` text NOT NULL,

`meta\_title` varchar(255) NOT NULL,

`meta\_h1` varchar(255) NOT NULL,

`meta\_description` varchar(255) NOT NULL,

`meta\_keyword` varchar(255) NOT NULL,

PRIMARY KEY (`category\_id`,`language\_id`),

KEY `name` (`name`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8

Все данные этой таблицы формируются из названия категории.

Таблица **category\_to\_store**

CREATE TABLE `category\_to\_store` (

`category\_id` int(11) NOT NULL,

`store\_id` int(11) NOT NULL,

PRIMARY KEY (`category\_id`,`store\_id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8

В случае добавления категории, добавляется запись id категории, второе значение ноль

Таблица **manufacturer\_description** загружается, в случае добавления нового производителя, вместе с таблицей **manufacturer**

Таблица **manufacturer\_to\_store** так же добавляются данные при загрузке нового производителя. Id ноль

Таблица **product\_description**, заполняется вместе с таблицей **product**.

Таблица **product\_to\_category**. Важная таблица которая связывает Товар и категории.

**CREATE TABLE `product\_to\_category` (**

`product\_id` int(11) NOT NULL,

`category\_id` int(11) NOT NULL,

`main\_category` tinyint(1) NOT NULL DEFAULT '0',

PRIMARY KEY (`product\_id`,`category\_id`),

KEY `category\_id` (`category\_id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8

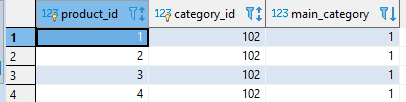


Таблица **product\_to\_store** . В эту таблицу добавляем при добавлении товара. Id и 0

Таблица **url\_alias**. Очень важная таблица. При добавлении любой сущности мы должны добавить ее отображение url в виде ЧПУ.

CREATE TABLE `url\_alias` (

`url\_alias\_id` int(11) NOT NULL AUTO\_INCREMENT,

`query` varchar(255) NOT NULL,

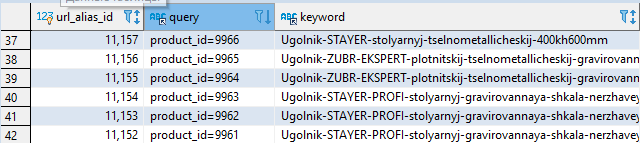
`keyword` varchar(255) NOT NULL,

PRIMARY KEY (`url\_alias\_id`),

KEY `query` (`query`),

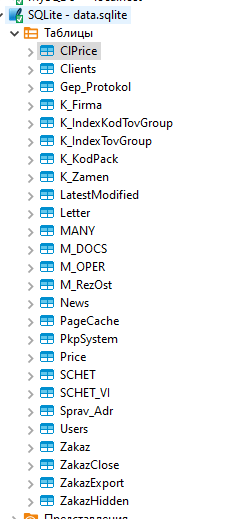
KEY `keyword` (`keyword`)

) ENGINE=MyISAM AUTO\_INCREMENT=23780 DEFAULT CHARSET=utf8



Остальные таблицы перечислять здесь не будем, они нам не потребуются

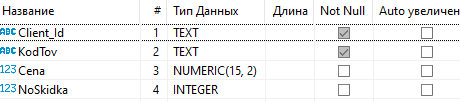
## 2.4. Анализ БД поставщика

 Состоит из 26 таблиц

Внешние ключи так же не прописаны

Таблица **ClPrice**.

Из таблицы мы получаем цену товара без скидки и информацию о том применяется скидка к данному товару или нет



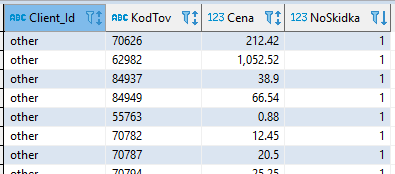
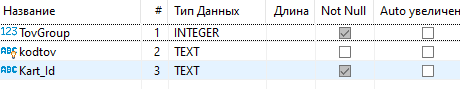


Таблица **K\_IndexKodTovGroup**. Таблица в которой привязка товара к группе товара.



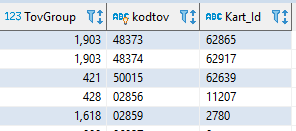
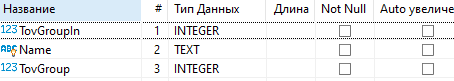


Таблица **K\_IndexTovGroup**. Привязка товарных подкатегорий к основным категориям товара.



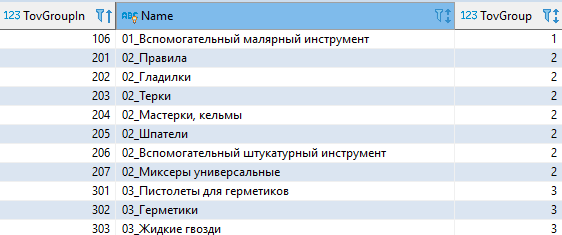
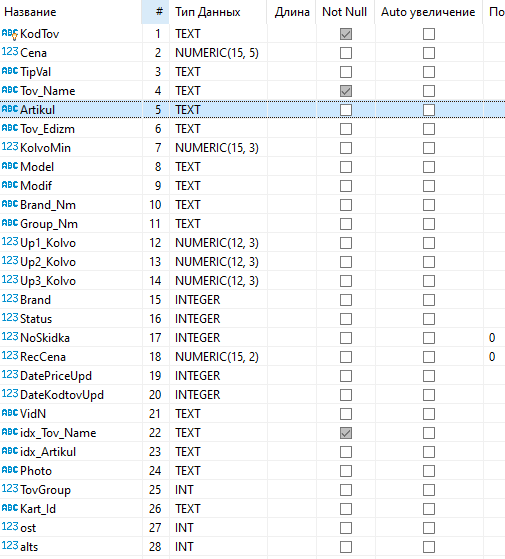


Таблица **M\_RezOst**. В таблице только одно поле Код товара, там хранятся только те товары которые есть в наличии.

Таблица **Price**. Самая главная для нас таблица. В ней храниться основная информация о товаре



Что нам отсюда понадобиться:

1. KodTov
2. Tov\_Name
3. Artikul. Это основное поле по которому связывается товар в интернет магазине и в таблице поставщика, к сожалению он имеет тип текст.
4. KolvoMin минимальная количество отгрузки (упаковка)
5. Model модель
6. TovGroup – номер группы товара
7. Ost наличие товара

## 2.5. Создание служебной БД

Создадим на сервере хостер провайдера служебную БД для импорта и экспорта данных

CREATE DATABASE test\_imp CHARACTER SET utf8 COLLATE utf8\_general\_ci;

Почему то без явного указания кодировки создает в кодировке latin

## 2.6. Реализация обновления остатков

Создадим таблицу в которой будем хранить остатки и цены в служебной базе данных

**CREATE TABLE product\_balances (**

**id serial PRIMARY KEY,**

**artikul varchar(64) NOT NULL UNIQUE,**

**price decimal(15,2) NOT NULL,**

**balance int DEFAULT 0,**

**UNIQUE unique\_artikul(artikul));**

Сформируем запрос из базы данных поставщика в текстовый файл, для загрузки в созданную таблицу в сервисной базе данных на сервере магазина.

Нам нужно выгрузить три поля. Артикул, остаток товара и цену. Остаток товара в БД поставщика либо 0 либо 1. Мы будем выгружать 0 , если ноль и 100 если 1. Что бы в нашем магазине числился товар в наличии условно 100 шт.

Цену поставщика нужно рассчитать. Она берётся из таблицы ClPrice, далее если к товару применима скидка ( в поле NoSkidka стоит ноль), то из этой цены вычитается скидка. Если там единица, то скидка не предоставляется.

**SELECT artikul,**

**CASE WHEN c.NoSkidka=1 THEN c.Cena ELSE (c.Cena\*(100-(SELECT Proc\_Skid from Clients limit 1))/100 ) END as price,**

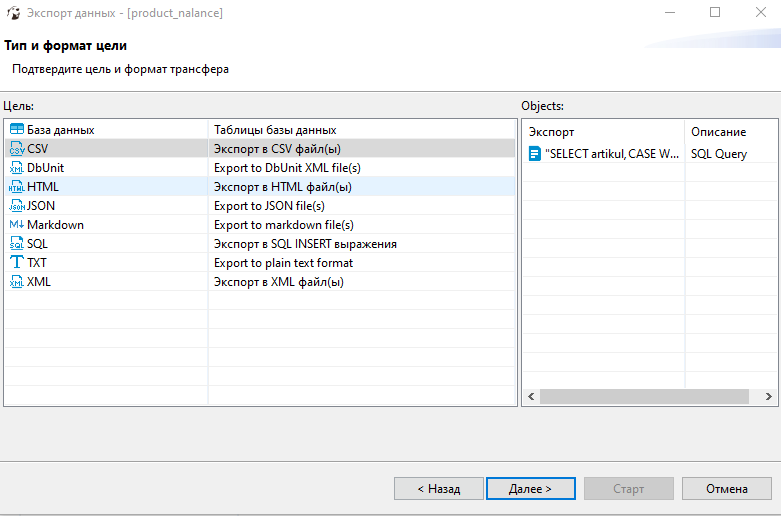
**CASE WHEN ost=1 THEN 100 ELSE 0 END as balance**

**FROM ClPrice c**

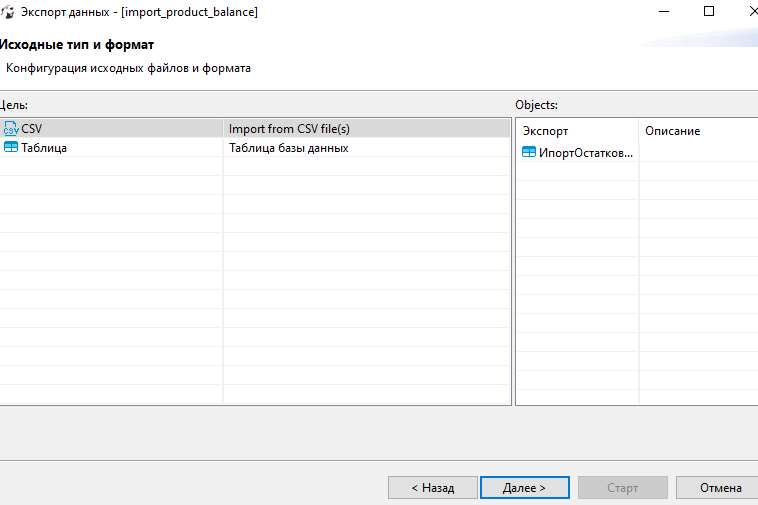
**JOIN Price p**

**ON c.KodTov=p.KodTov;**

Настроим экспорт в csv файл в DBeaver



Далее настроим импорт из файла.



Файл импорта будет приложен к работе.

Далее сформируем саму процедуру обновления данных в таблице магазина **product**.

1. Нужно обновить цены и наличие у тех позиций, которые присутствуют так же в таблице сервисной БД. Это можно сделать запросом:

**UPDATE product p JOIN**

**test\_imp.product\_balances as tp ON p.model = tp.artikul**

**SET**

**p.quantity=tp.balance,**

**p.price=(CASE**

**WHEN tp.price < 500 THEN tp.price\*1.25**

**WHEN tp.price < 1000 THEN tp.price\*1.2**

**WHEN tp.price < 3000 THEN tp.price\*1.15**

**ELSE tp.price\*1.1**

**END);**

1. Те позиции которые отсутствуют в сервисной базе, мы удалять не будем, просто покажем что их нет в наличии, для этого в поле количество укажем ноль.

**UPDATE product p LEFT JOIN**

**test\_imp.product\_balances as tp ON p.model = tp.artikul**

**SET**

**p.quantity=0**

**WHERE tp.id IS NULL;**

Для автоматической работы, создадим и сохраним процедуру

**CREATE PROCEDURE update\_price\_count()**

**BEGIN**

**START TRANSACTION;**

**UPDATE product p JOIN**

**test\_imp.product\_balances as tp ON p.model = tp.artikul**

**SET**

**p.quantity=tp.balance,**

**p.price=(CASE**

**WHEN tp.price < 500 THEN tp.price\*1.25**

**WHEN tp.price < 1000 THEN tp.price\*1.2**

**WHEN tp.price < 3000 THEN tp.price\*1.15**

**ELSE tp.price\*1.1**

**END);**

**UPDATE product p LEFT JOIN**

**test\_imp.product\_balances as tp ON p.model = tp.artikul**

**SET**

**p.quantity=0**

**WHERE tp.id IS NULL;**

**COMMIT;**

**END;**

Проверим:

**CALL update\_price\_count();**

Все работает.

Переключим магазин на тестовую БД для проверки.

Проверка показала, что все прекрасно работает. Переносим на основную БД.

Таким образом, что бы произвести обновление остатков товара и цен необходимо проделать следующее:

1. Запустить в DBeaver задачу экспорта остатков.
2. Запустить в DBeaver задачу импорта остатков
3. Запустить процедуру обновления

## 2.7. Реализация добавления новых элементов

1. Создадим в сервисной БД таблицу, в которой будем хранить артикулы, которые есть в таблице product\_balances, но нет в таблице product БД магазина.

Запрос для этого:

**TRUNCATE test\_imp.lost;**

**INSERT INTO test\_imp.lost (artikul)**

**SELECT artikul FROM test\_imp.product\_balances pb**

**LEFT JOIN product p ON pb.artikul=p.model**

**WHERE p.model IS NULL;**

Создадим в БД магазина процедуру, что бы быстро исполнять данную операцию.

**CREATE PROCEDURE find\_lost()**

**BEGIN**

**START TRANSACTION;**

**TRUNCATE test\_imp.lost;**

**INSERT INTO test\_imp.lost (artikul)**

**SELECT artikul FROM test\_imp.product\_balances pb**

**LEFT JOIN product p ON pb.artikul=p.model**

**WHERE p.model IS NULL;**

**COMMIT;**

**END;**

Перенесем эту таблицу в Базу данных поставщика

Создадим запрос для выгрузки данных из БД поставщика товаров, которых нет

Создадим таблицу для импорта данных из БД поставщика

Автоматически мы будем грузить только те записи, в которых определяется категория и производитель. То есть они существуют в нашей базе магазина. Оставшиеся записи требуют ручной обработки данных, что бы определить и добавить производителя и категорию товара.

Загрузка будет осуществляться в таблицы

1. **Product**
2. **product\_description**
3. **product\_to\_store**
4. **product\_to\_category.**
5. **url\_alias**

В таблицу url\_alias, необходимо внести ЧПУ товара, который мы будем формировать из названия товара. Для этого создадим функцию, которая сделает транслитерацию и уберет лишние символы.

Сформируем процедуру и будем загружать в тарнзакции.

Заполняемые таблицы:

1. **Product**
2. **product\_description**
3. **product\_to\_store**
4. Manufacturer
5. manufacturer\_description
6. manufacturer\_to\_store
7. Category
8. category\_description
9. category\_to\_store
10. **product\_to\_category**.
11. **url\_alias**

Нужная информация о товаре:

1. Разобраться в структуре база данных магазина

2. Выделить, ту часть которая отвечает за товар

3. Сформировать модель - Базу данных, которая имеет только ту подсистему, с которой мне нужно работать . Для экспериментов

4. Разобраться в базе данных поставщика. Понять где и как брать данные

5. Разработать механизмы получения данных, их обработки до нужной структуры и загрузки в модельную базу

6. Протестировать и убедиться в корректности работы всех систем

7. реализовать на рабочую тестовую базу магазина.

8. Убедится что приложение работает правильно.