**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
 РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет информационных технологий**

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

«Реализация приложения в архитектуре клиент-сервер. Информационная система фотоцентра»

студента 3 курса, 22201 группы

**Галкина Константина Сергеевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Е. И. Вершинина

Новосибирск 2025

**Оглавление**

[**ОПИСАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ФОТОЦЕНТРА** 3](#_Toc199360711)

[**ЗАПРОСЫ** 4](#_Toc199360713)

[**СУЩНОСТИ** 5](#_Toc199360726)

[**СУЩНОСТИ и АТРИБУТЫ** 7](#_Toc199360727)

[**СУЩНОСТИ и СВЯЗИ** 13](#_Toc199360728)

[**ER-диаграмма** 15](#_Toc199360729)

[**ПОЛЬЗОВАТЕЛИ и РОЛИ** 16](#_Toc199360730)

[**ПРАВА ДОСТУПА К ТАБЛИЦАМ** 17](#_Toc199360731)

[**ПРИВЕДЕННЫЕ НОРМАЛЬНЫЕ ФОРМЫ** 18](#_Toc199360732)

[**ПЕРЕХОД ОТ КОНЦЕПТУАЛЬНОЙ СХЕМЫ К ЛОГИЧЕСКОЙ СХЕМЕ БД** 19](#_Toc199360733)

[**ТРИГГЕРЫ И ХРАНИМЫЕ ПРОЦЕДУРЫ** 25](#_Toc199360734)

[**SQL скрипты создания ролей** 27](#_Toc199360735)

[**SQL скрипты создания таблиц** 28](#_Toc199360736)

[**SQL скрипты создания триггеров и хранимых процедур** 34](#_Toc199360737)

[**SQL скрипты запросов** 43](#_Toc199360738)

[**КОНКУРЕТНЫЙ ДОСТУП К БД** 51](#_Toc199360751)

[**ПРИНЯТЫЕ МЕРЫ** 52](#_Toc199360752)

[**АРХИТЕКТУРА ПРИЛОЖЕНИЯ** 53](#_Toc199360753)

## **ОПИСАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ФОТОЦЕНТРА**

Фотоцентр имеет главный офис и сеть филиалов и киосков приёма заказов, расположенных по определённым адресам. Филиалы и киоски различаются количеством рабочих мест. В киосках осуществляется только приём заказов, поэтому каждый киоск прикреплён к определённому филиалу, в котором эти заказы выполняются. В филиалах имеется необходимое оборудование для проявки пленок и печати фотографий. Филиалы и киоски принимают заказы на проявку пленок, печать фотографий и проявку, и печать вместе. В заказе на печать указывается количество фотографий с каждого кадра, общее количество фотографий, формат, тип бумаги и срочность выполнения заказа. При заказе большого количества фотографий предоставляются скидки. Срочные заказы принимаются только в филиалах и они имеют цену в два pаза больше, чем обычный заказ. Пpи пpиобpетении дисконтной каpты клиент получает значительные скидки на печать фотогpафий. Пленка, пpиобpетенная в том же филиале, куда она пpинесена на пpоявку, пpоявляется бесплатно. Клиентов можно pазделить на пpофессионалов и любителей. Пpофессионалам, пpиносящим заказы в один и тот же филиал, могут быть пpедложены пеpсональные скидки. Фотомагазины и киоски пpедлагают к пpодаже pазличные фототоваpы: фотопленки, фотоаппаpаты, альбомы и дpугие фотопpинадлежности. Фотомагазины также пpедлагают дополнительные виды услуг: фотографии на документы, pеставрация фотографий, прокат фотоаппаратов, художественное фото, пpедоставление услуг пpофессионального фотографа. Сведения о выполненных заказах и пpодаже pазличных фототоваpов собиpаются и обpабатываются, и на основе этой инфоpмации делается общий заказ на поставку pасходных матеpиалов (фотобумага, фотопленка, химические pеактивы), фототоваpов и обоpудования. Полученные товаpы и матеpиалы pаспpеделяются в соответствии с запpосами по киоскам и магазинам. У фотоцентpа может быть несколько поставщиков, котоpые специализиpуются на pазличных поставках, либо на поставках фототоваpов pазличных фирм.

## **ЗАПРОСЫ**

1. Получить перечень и общее число пунктов приема заказов на фотоработы по филиалам, по киоскам приема заказов, в целом по фотоцентру.
2. Получить перечень и общее число заказов на фотоработы по филиалам, киоскам приема заказов, в целом по фотоцентру, поступивших в течение некоторого периода времени.
3. Получить перечень и общее число заказов (отдельно простых и срочных) на отдельные виды фоторабот по указанному филиалу, киоску приема заказов, поступивших в течение некоторого периода времени.
4. Получить сумму выручки с заказов (отдельно простых и срочных) на отдельные виды фоторабот по указанному филиалу, киоску приема заказов, поступивших в течение некоторого периода времени.
5. Получить количество отпечатанных фотографий в рамках простых и срочных заказов по указанному филиалу, киоску приема заказов, фотоцентру в целом за некоторый период времени.
6. Получить количество проявленных фотопленок в рамках простых и срочных заказов по указанному филиалу, киоску приема заказов, фотоцентру в целом за некоторый период времени.
7. Получить перечень поставщиков в целом по фотоцентру, поставщиков отдельных видов фототоваров, сделавших поставки в некоторый период, поставки определенного объема.
8. Получить список клиентов в целом по фотоцентру, клиентов указанного филиала, имеющих скидки, сделавших заказы определенного объема.
9. Получить сумму выручки от реализации фототоваров в целом по фотоцентру, по указанному филиалу, проданных в течение некоторого периода времени.
10. Получить перечень фототоваров и фирм, их производящих, которые пользуются наибольшим спросом в целом по фотоцентру, в указанном филиале.
11. Получить перечень реализованных фототоваров и объемы их реализации в целом по фотоцентру, по указанному филиалу, проданных в течение некоторого периода времени.
12. Получить перечень рабочих мест фотоцентра в целом и указанного профиля.

## **СУЩНОСТИ**

Ремарка:  
Так как о главном офисе недостаточно информации, он убран из представления.

|  |  |  |
| --- | --- | --- |
| **№** | **Сущность** | **Описание** |
| 1 | branches | Перечень филиалов фотоцентра |
| 2 | outlets | Перечень торговых точек |
| 3 | outlet\_types | Перечень типов торговых точек(филиал, киоск, фотомагазин) |
| 4 | photo\_stores | Перечень фотомагазинов |
| 5 | kiosks | Перечень киосков |
| 6 | clients | Информация о клиентах, включая их тип (любитель/профессионал), наличие скидок |
| 7 | items | Информация о товарах(название, цена, фирма) |
| 8 | service\_types\_outlets | Перечень типов услуг для каждого типа торговой точки |
| 9 | service\_types | Перечень всех типов услуг, предоставляемых фотоцентром |
| 10 | orders | Информация о заказах(суммарная цена, дата, торговая точка и т.д.) |
| 11 | firms | Информация о фирмах(название) |
| 12 | films | Информация о плёнках |
| 13 | service\_orders | Информация о заказах услуг |
| 14 | film\_development\_orders | Информация о заказах на проявку плёнок |
| 15 | vendors | Информация о поставщиках |
| 16 | vendor\_items | Информация о товарах поставщика(цена, количество, товар) |
| 17 | print­\_discounts | Информация о скидках на печати |
| 18 | paper\_types | Информация о типах бумаг |
| 19 | paper\_sizes | Информация о форматах бумаг |
| 20 | print\_orders | Информация о заказах на печать |
| 21 | print\_prices | Информация о ценах на печать |
| 22 | storages | Информация о хранилищах(вместимость, главный ли склад и т.д.) |
| 23 | storage\_items | Информация о товарах в хранилище |
| 24 | frames | Информация о кадрах для печати |
| 25 | deliveries | Информация о доставках товаров |
| 26 | delivery\_items | Информация о доставленных товарах |
| 27 | service\_types\_needed\_items | Информация о товарах необходимых для каждого типа услуги |

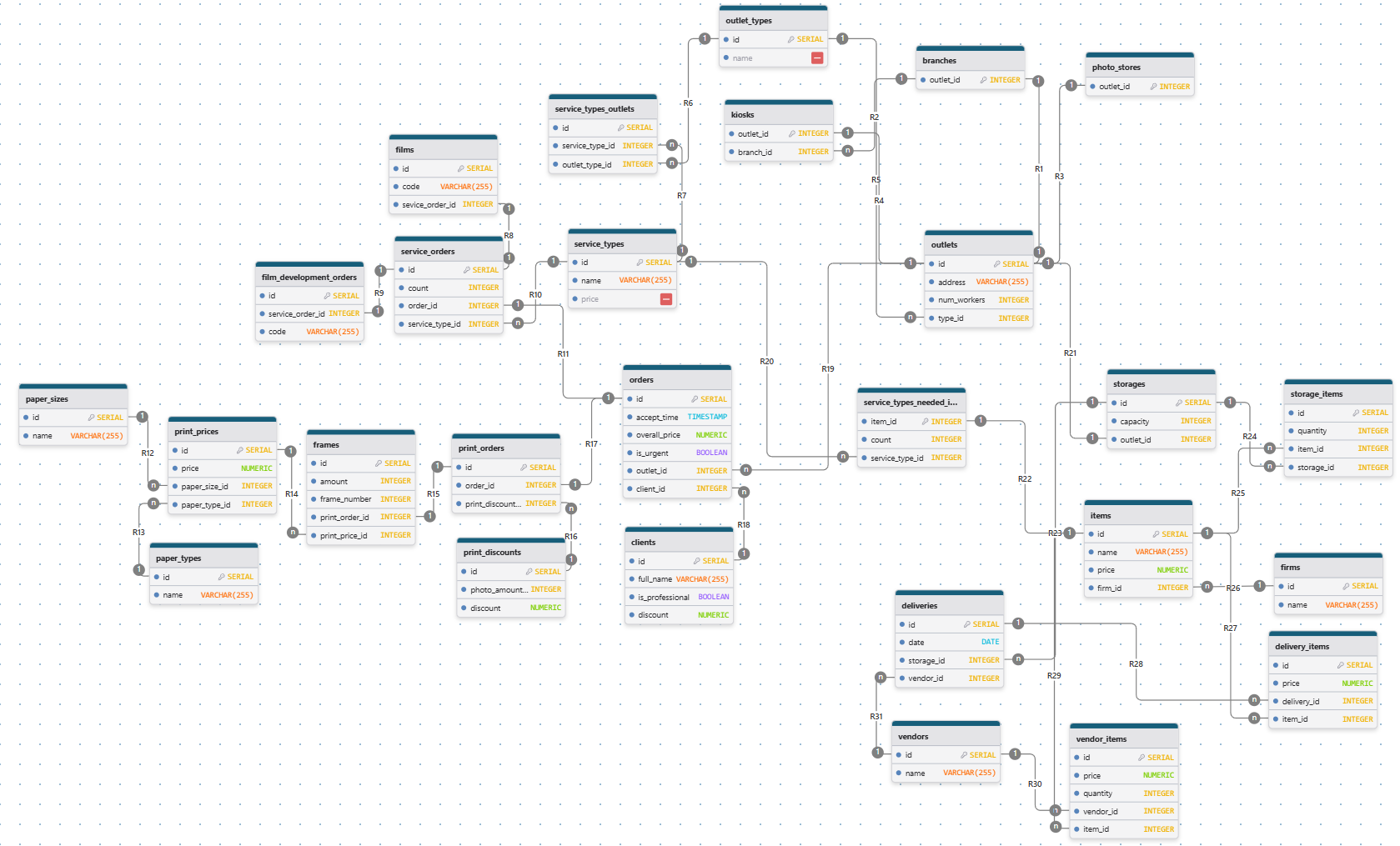
## **СУЩНОСТИ и АТРИБУТЫ**

|  |  |  |
| --- | --- | --- |
| **Сущность** | **Атрибуты** | **Описание** |
| branches | outlet\_id | Уникальный идентификатор торговой точки/здания (первичный ключ, внешний ключ на outlets) |
| outlet\_types | id  name | Уникальный идентификатор типа торговой точки(первичный ключ)  Тип торговой точки(ФИЛИАЛ, ФОТОМАГАЗИН, КИОСК) |
| outlets | id  address  num\_workers  type\_id | Уникальный идентификатор торговой точки(первичный ключ)  Адрес торговой точки  Количество рабочих мест  Идентификатор типа торговой точки(внешний ключ на outlet\_types) |
| kiosks | outlet\_id  branch\_id | Уникальный идентификатор торговой точки/здания (первичный ключ, внешний ключ на outlets)  Идентификатор филиала(внешний ключ на branches) |
| photo\_stores | outlet\_id | Уникальный идентификатор торговой точки/здания (первичный ключ, внешний ключ на outlets) |
| service\_types | id  name  price | Уникальный идентификатор типа услуги(первичный ключ)  Имя услуги  Цена услуги |
| service\_types\_outlets | id  service\_type\_id  outlet\_type\_id | Уникальный идентификатор типа услуг для торговой точки/здания (первичный ключ)  Идентификатор типа услуги(внешний ключ на service\_types)  Идентификатор типа торговой точки(внешний ключ на outlet\_types) |
| firms | id  name | Уникальный идентификатор фирмы (первичный ключ)  Название фирмы |
| items | id  name  price  firm\_id | Уникальный идентификатор товара (первичный ключ)  Название товара  Цена товара  Идентификатор фирмы (внешний ключ на firms) |
| clients | id  full\_name  is\_professional  discount | Уникальный идентификатор клиента (первичный ключ)  ФИО клиента  Профессионал или любитель  Персональная скидка |
| orders | id  accept\_date  overall\_price  is\_urgent  outlet\_id  client\_id | Уникальный идентификатор заказа (первичный ключ)  Дата заказа  Финальная стоимость с учётом скидок  Срочность  Идентификатор торговой точки/здания (внешний ключ на outlets)  Идентификатор клиента(внешний ключ на clients) |
| print\_orders | id  order\_id  print\_discount\_id | Уникальный идентификатор заказа на печать (первичный ключ)  Идентификатор заказа (внешний ключ на orders)  Идентификатор скидок на печать (внешний ключ на discounts) |
| storages | id  capacity  outlet\_id | Уникальный идентификатор хранилища(первичный ключ)  Вместимость  Идентификатор торговой точки/здания (внешний ключ на outlets) |
| service\_orders | id  count  order\_id  service\_type\_id | Уникальный идентификатор заказа услуг(первичный ключ)  Количество  Идентификатор заказа (внешний ключ на orders)  Идентификатор типа услуги (внешний ключ на service\_types) |
| film\_development\_orders | id  service\_order\_id  code | Уникальный идентификатор заказа на проявку плёнки(первичный ключ)  Идентификатор заказа услуг (внешний ключ на service\_orders)  Код |
| films | id  code  service\_order\_id | Уникальный идентификатор плёнки (первичный ключ)  Код  Идентификатор заказа услуг (внешний ключ на service\_orders) |
| vendors | id  name | Уникальный идентификатор поставщика (первичный ключ)  Имя поставщика |
| vendor\_items | id  price  quantity  vendor\_id  item\_id | Уникальный идентификатор товара поставщика (первичный ключ)  Цена по которой продаётся товар  Количество  Идентификатор поставщика(внешний ключ на vendors)  Идентификатор товара(внешний ключ на items) |
| print\_discounts | id  photo\_amount  discount | Уникальный идентификатор скидки на печать (первичный ключ)  Количество фото  Скидка |
| paper\_types | id  name | Уникальный идентификатор типа бумаг(первичный ключ)  Тип |
| print\_prices | id  price  paper\_size\_id  paper\_type\_id | Уникальный идентификатор цены печати(первичный ключ)  Цена  Идентификатор формата бумаг(внешний ключ на paper\_sizes)  Идентификатор типа бумаг(внешний ключ на paper\_types) |
| paper\_sizes | id  name | Уникальный идентификатор формата бумаги(первичный ключ)  Размер |
| frames | id  amount  frame\_number  print\_order\_id  print\_price\_id | Уникальный идентификатор кадра бумаги(первичный ключ)  Количество фотографий  Номер кадра  Идентификатор заказа печати(внешний ключ на print\_orders)  Идентификатор цены печати(внешний ключ на print\_prices) |
| deliveries | id  date  storage\_id  vendor\_id | Уникальный идентификатор доставки (первичный ключ)  Дата доставки  Идентификатор хранилища(внешний ключ на storages)  Идентификатор поставщика(внешний ключ на vendors) |
| delivery\_items | id  price  quantity  delivery\_id  item\_id | Уникальный идентификатор доставленного товара (первичный ключ)  Цена товара  Количество товара  Идентификатор доставки(внешний ключ на deliveries)  Идентификатор товара(внешний ключ на items) |
| service\_types\_needed\_items | item\_id  count  service\_type\_id | Уникальный идентификатор товара необходимого для типа услуги(первичный ключ, внешний ключ на items)  Количество товара  Идентификатор типа услуги(внешний ключ на service\_types) |
| storage\_items | id  quantity  item\_id  storage\_id | Уникальный идентификатор товара необходимого для типа услуги(первичный ключ)  Количество товара  Идентификатор товара(внешний ключ на items)  Идентификатор хранилища(внешний ключ на storages) |

## **СУЩНОСТИ и СВЯЗИ**

|  |  |  |
| --- | --- | --- |
| **№ пп** | **Связь** | **Описание** |
| R1 | 1:1 | Филиал может быть привязан к одному зданию |
| R2 | 1:N | Филиал может иметь несколько киосков |
| R3 | 1:1 | Фотомагазин может быть привязан только к одной торговой точке |
| R4 | 1:1 | Киоск может быть привязан только к одной торговой точке |
| R5 | 1:N | Один тип торговой точки может быть у нескольких торговых точек |
| R6 | 1:N | Один тип торговой точки может иметь несколько типов услуг торговых точек |
| R7 | 1:N | Один тип услуги может быть привязан к нескольким типам услуг торговых точек |
| R8 | 1:1 | Одна плёнка может быть связана с одним заказом услуг |
| R9 | 1:1 | Один заказ на проявку плёнок может быть связан с одним типом услуг |
| R10 | 1:N | Один тип услуги может быть связан с несколькими типом заказа услуг |
| R11 | 1:1 | Один заказ услуги может быть связан с одним заказом |
| R12 | 1:1 | Один формат бумаги может быть связан с несколькими ценами на печать |
| R13 | 1:1 | Один тип бумаги может быть связан с несколькими ценами на печать |
| R14 | 1:1 | Одна цена печати может быть связан с несколькими кадрами |
| R15 | 1:1 | Один кадр может быть связан с одним заказом на печать |
| R16 | 1:N | Одна скидка на печать может быть связана с несколькими заказами на печать |
| R17 | 1:1 | Один заказ на печать может быть связан с одним заказом |
| R18 | 1:N | Один клиент может быть связан с несколькими заказами |
| R19 | 1:N | Одна торговая точка может быть связана с несколькими заказами |
| R20 | 1:N | Один тип услуги может быть связан с несколькими необходимыми товарами для типов услуг |
| R21 | 1:1 | Одно здание может быть связано с одним складом |
| R22 | 1:1 | Один необходимый товар для типов услуг может быть связан с одним товаром |
| R23 | 1:N | Одно хранилище может быть связано с несколькими доставками |
| R24 | 1:N | Одно хранилище может быть связано с несколькими товарами в хранилище |
| R25 | 1:1 | Один товар может быть связан с одним товаром в хранилище |
| R26 | 1:N | Одна фирма может быть связана с несколькими товарами |
| R27 | 1:N | Один товар может быть связан с несколькими товарами из доставки |
| R28 | 1:N | Одна доставка может быть связана с несколькими товарами из доставки |
| R29 | 1:N | Один товар может быть связан с несколькими товарами поставщика |
| R30 | 1:N | Один поставщик может быть связан с несколькими товарами поставщика |
| R31 | 1:N | Один поставщик может быть связан с несколькими доставками |

## **ER-диаграмма**



## **ПОЛЬЗОВАТЕЛИ и РОЛИ**

|  |  |  |
| --- | --- | --- |
| **№** | **Название роли** | **Описание** |
| 1 | Admin | Особая роль. Полный доступ к БД. |
| 2 | Vendor | Роль поставщика. |
| 3 | Employee | Роль сотрудника(например кассира). |
| 4 | Manager | Роль управляющего торговой точкой. |

## **ПРАВА ДОСТУПА К ТАБЛИЦАМ**

* С – создание новых данных
* R – чтение данных
* U – обновление/редактирование существующих данных
* D – удаление существующих данных

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Vendor** | **Employee** | **Manager** |
| branches | **R** | **R** | **R** |
| outlets | **R** | **R** | **R** |
| outlet\_types | **R** | **R** | **R** |
| photo\_stores | **R** | **R** | **R** |
| kiosks | **R** | **R** | **R** |
| clients |  | **CR** | **CRU** |
| items | **R** | **R** | **CRUD** |
| service\_types\_outlets | **R** | **R** | **CRUD** |
| service\_types | **R** | **R** | **CRUD** |
| orders |  | **CR** | **CRUD** |
| firms | **CRUD** | **R** | **R** |
| films |  | **CR** | **CRUD** |
| service\_orders |  | **CR** | **CRUD** |
| film\_development\_orders |  | **CR** | **CRUD** |
| vendors | **CRUD** | **R** | **R** |
| vendor\_items | **CRUD** | **R** | **R** |
| print­\_discounts |  | **CR** | **CRUD** |
| paper\_types |  | **CR** | **CRUD** |
| paper\_sizes |  | **CR** | **CRUD** |
| print\_orders |  | **CR** | **CRUD** |
| print\_prices |  | **CR** | **CRUD** |
| storages | **R** | **R** | **CRUD** |
| storage\_items | **R** | **R** | **CRUD** |
| frames |  | **CR** | **CRUD** |
| deliveries | **R** | **R** | **CRUD** |
| delivery\_items | **R** | **R** | **CRUD** |
| service\_types\_needed\_items |  | **R** | **CRUD** |

## **ПРИВЕДЕННЫЕ НОРМАЛЬНЫЕ ФОРМЫ**

1. Первая нормальная форма (1НФ)

* Все поля содержат только атомарные (неделимые) значения.
* Нет повторяющихся групп или массивов в одной записи.

1. Вторая нормальная форма (2НФ)

* Выполняется 1НФ.
* Все неключевые атрибуты полностью зависят от всего составного ключа (если ключ составной).

1. Третья нормальная форма (3НФ)

* Выполняется 2НФ.
* Нет транзитивных зависимостей: неключевые атрибуты не зависят друг от друга, а только от ключа.

1. Четвёртая нормальная форма (4НФ)

* Нет многозначных зависимостей (когда одна запись связана с множеством значений в двух разных атрибутах независимо).

**Схема БД находится в 4НФ т.к.:**

**1. Каждая таблица имеет первичный ключ, и все атрибуты зависят только от него (=> 2НФ и 3НФ соблюдаются).**

**2. Нет множественных независимых наборов значений:**

**3. Все потенциальные MVD(Multivalued Dependency) реализованы через отдельные таблицы связи:**

* service\_types\_outlets — связь типа сервиса и типа точки.
* service\_types\_needed\_items — связь сервисов и нужных товаров.
* vendor\_items, delivery\_items, storage\_items — каждый набор сведён к фактам с указанием связей и количеств.

**4. Нигде нет атрибутов, которые представляют собой независимые многозначные факты о ключе. Всё разделено на независимые отношения.**

## **ПЕРЕХОД ОТ КОНЦЕПТУАЛЬНОЙ СХЕМЫ К ЛОГИЧЕСКОЙ СХЕМЕ БД**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Сущность** | **Атрибуты** | **Тип** | **Ограничения** | **Первичный ключ** | **Внешний ключ** |
| branches | outlet\_id | INT | NOT NULL,  UNIQUE | outlet\_id | |  | | --- | |  |  |  | | --- | | outlets(id) ON DELETE CASCADE | |
| outlet\_types | id  name | SERIAL  VARCHAR(255) | NOT NULL, UNIQUE  NOT NULL, UNIQUE | id |  |
| outlets | id  address  num\_workers  type\_id | SERIAL  VARCHAR(255)  INT  INT | NOT NULL, UNIQUE  NOT NULL, UNIQUE  NOT NULL, CHECK(num\_workers >=0)  NOT NULL |  | outlet\_types(id) ON DELETE RESTRICT |
| kiosks | outlet\_id  branch\_id | |  | | --- | |  |  |  |  | | --- | --- | | INT | | |  |  |  | | --- | | INT | | NOT NULL  NOT NULL |  | outlets(id) ON DELETE CASCADE  branches(outlet\_id) ON DELETE CASCADE |
| photo\_stores | outlet\_id | INT | NOT NULL | outlet\_id | outlets(id) ON DELETE CASCADE |
| service\_types | id  name  price | |  | | --- | |  |  |  | | --- | | SERIAL | | VARCHAR(255) |   NUMERIC(10, 2) | NOT NULL  NOT NULL, UNIQUE  NOT NULL, CHECK (price >= 0) | id |  |
| service\_types\_outlets | id  service\_type\_id  outlet\_type\_id | SERIAL  INT  INT | NOT NULL  NOT NULL  NOT NULL | id | service\_types(id) ON DELETE CASCADE  outlet\_types(id) ON DELETE CASCADE |
| firms | id  name | SERIAL   |  | | --- | | VARCHAR(255) |  |  | | --- | |  | | NOT NULL  NOT NULL, UNIQUE | id |  |
| items | id  name  price  firm\_id | SERIAL  VARCHAR(255)  NUMERIC(10, 2)  INT | NOT NULL  NOT NULL  NOT NULL  NOT NULL , CHECK (price >= 0)  NOT NULL | id | |  | | --- | |  |  |  | | --- | | firms(id) ON DELETE SET NULL | |
| clients | id  full\_name  is\_professional  discount | SERIAL   |  | | --- | |  |  |  | | --- | | VARCHAR(255) |   BOOLEAN   |  | | --- | | NUMERIC(5, 2) |  |  | | --- | |  | | NOT NULL  NOT NULL  NOT NULL  NOT NULL | id |  |
| orders | id  accept\_time  overall\_price  is\_urgent  outlet\_id  client\_id | SERIAL  TIMESTAMP  NUMERIC(10, 2)  BOOLEAN  INT  INT | NOT NULL  NOT NULL, DEFAULT NOW()  NOT NULL  NOT NULL  NOT NULL  NOT NULL | id | |  | | --- | |  |  |  | | --- | | outlets(id) |   clients(id) |
| print\_orders | id  order\_id  print\_discount\_id | SERIAL  INT  INT | NOT NULL  NOT NULL  NOT NULL | id | orders(id)  print\_discounts(id) |
| storages | id  capacity  outlet\_id | SERIAL  INT  INT | NOT NULL  NOT NULL  NOT NULL | id | |  | | --- | |  |  |  | | --- | | outlets(id) | |
| service\_orders | id  count  order\_id  service\_type\_id | SERIAL  INT  INT  INT | NOT NULL  NOT NULL  NOT NULL  NOT NULL | id | orders(id)  service\_types(id) |
| film\_development\_orders | id  service\_order\_id  code | SERIAL  INT  VARCHAR(255) | NOT NULL  NOT NULL  NOT NULL | id | |  | | --- | |  |  |  | | --- | | service\_orders(id) | |
| films | id  code  service\_order\_id | SERIAL  VARCHAR(255)  INT | NOT NULL  NOT NULL  NOT NULL | id | service\_orders(id) |
| vendors | id  name | SERIAL  VARCHAR(255) | NOT NULL  NOT NULL | id |  |
| vendor\_items | id  price  quantity  vendor\_id  item\_id | SERIAL  NUMERIC(10,2)  INT  INT  INT | NOT NULL  NOT NULL  NOT NULL  NOT NULL  NOT NULL | id | vendors(id)  items (id) |
| print\_discounts | id  photo\_amount  discount | SERIAL  INT  NUMERIC(5,2) | NOT NULL  NOT NULL  NOT NULL | id |  |
| paper\_types | id  name | SERIAL  VARCHAR(255) | NOT NULL  NOT NULL | id |  |
| print\_prices | id  price  paper\_size\_id  paper\_type\_id | SERIAL  NUMERIC(10,2)  INT  INT | NOT NULL  NOT NULL  NOT NULL  NOT NULL | id | paper\_sizes(id)  paper\_types(id) |
| paper\_sizes | id  name | SERIAL  VARCHAR(255) | NOT NULL  NOT NULL | id |  |
| frames | id  amount  frame\_number  print\_order\_id  print\_price\_id | SERIAL  INT  INT  INT  INT | NOT NULL  NOT NULL  NOT NULL  NOT NULL  NOT NULL | id | print\_orders (id)  print\_prices(id) |
| deliveries | id  date  storage\_id  vendor\_id | SERIAL  DATE  INT  INT | NOT NULL  NOT NULL  NOT NULL  NOT NULL | id | |  | | --- | |  |  |  | | --- | | storages(id) |  |  | | --- | |  |  |  | | --- | | vendors(id) | |
| delivery\_items | id  price  quantity  delivery\_id  item\_id | SERIAL  NUMERIC(10,2)  INT  INT  INT | NOT NULL  NOT NULL  NOT UNLL  NOT NULL  NOT NULL | id | deliveries(id)  items(id) |
| service\_types\_needed\_items | item\_id  count  service\_type\_id | INT  INT  INT | NOT NULL  NOT NULL  NOT NULL | |  | | --- | |  |  |  | | --- | | item\_id | | items(id)   |  | | --- | |  |  |  | | --- | | service\_types(id) | |
| storage\_items | id  quantity  item\_id  storage\_id | SERIAL  INT  INT  INT | NOT NULL  NOT NULL  NOT NULL  NOT NULL | id | storages(id) |

## **ТРИГГЕРЫ И ХРАНИМЫЕ ПРОЦЕДУРЫ**

|  |  |  |
| --- | --- | --- |
| **№** | **Тип события** | **Описание триггера** |
| 1 | **AFTER INSERT OR UPDATE OR DELETE** на **service\_orders**, **print\_orders**, **frames**; **AFTER UPDATE** на **clients.discount**, **print\_discounts.discount** | **trg\_recalculate\_order\_overall\_price:** Автоматически пересчитывает и обновляет **общую стоимость заказа (overall\_price)** в таблице orders. Учитывает стоимость услуг (включая удвоенную цену для срочных), стоимость печати (с учетом скидок клиента и скидок на печать), а также правило о **бесплатной проявке пленки**, если она куплена в том же филиале. |
| 2 | **AFTER INSERT OR UPDATE** на **delivery\_items; AFTER INSERT OR UPDATE** на **service\_orders** | **trg\_update\_storage\_quantity:** Поддерживает актуальное количество товаров (**quantity**) в таблице **storage\_items**. Увеличивает количество при поступлении товаров (**delivery\_items**) и уменьшает при использовании товаров для услуг (**service\_orders**). |
| 3 | **BEFORE INSERT OR UPDATE на storage\_items** | **trg\_check\_storage\_capacity:** Предотвращает превышение емкости склада. Проверяет, что суммарное количество товаров на складе не превысит его **capacity** после операции добавления или изменения количества товара. Если превышает, операция отменяется. |
| 4 | **BEFORE INSERT OR UPDATE** на **clients.is\_professional, clients.discount** | **trg\_set\_professional\_discount:** Автоматически устанавливает или сбрасывает скидку для клиентов в зависимости от их статуса **is\_professional**. Если клиент становится профессионалом, ему назначается скидка по умолчанию; если перестает – скидка сбрасывается. |
| 5 | **BEFORE INSERT OR UPDATE** на **orders.is\_urgent, orders.outlet\_id** | **trg\_enforce\_urgent\_orders\_at\_branches**: **Обеспечивает прием срочных заказов (is\_urgent = TRUE) только в филиалах**. Если срочный заказ пытаются оформить в другом типе торговой точки, операция отменяется. |

|  |  |  |
| --- | --- | --- |
| **№** | **Хранимая процедура** | **Описание** |
| 1 | sp\_get\_service\_price | Получить цену услуги по ее ID |
| 2 | sp\_get\_outlet\_type\_id | Получить ID типа торговой точки по ее названию |
| 3 | sp\_check\_film\_bought\_in\_outlet | Проверить, была ли пленка с данным кодом куплена в указанной торговой точке |
| 4 | sp\_get\_client\_discount | Получить текущую скидку клиента по его ID |

## **SQL скрипты создания ролей**

*-- Создание ролей*

CREATE ROLE vendor NOLOGIN;

CREATE ROLE employee NOLOGIN;

CREATE ROLE manager NOLOGIN;

*-- Создание пользователей и присвоение им ролей*

CREATE USER admin\_user WITH PASSWORD 'admin' SUPERUSER;

CREATE USER vendor\_user WITH PASSWORD 'vendor';

GRANT vendor TO vendor\_user;

CREATE USER employee\_user WITH PASSWORD 'employee';

GRANT employee TO employee\_user;

CREATE USER manager\_user WITH PASSWORD 'manager';

GRANT manager TO manager\_user;

## **SQL скрипты создания таблиц**

*-- Типы торговых точек*

CREATE TABLE outlet\_types (

    id SERIAL PRIMARY KEY,

    name VARCHAR(255) NOT NULL UNIQUE

);

*-- Здания торговых точек*

CREATE TABLE outlets (

    id SERIAL PRIMARY KEY,

    address VARCHAR(255),

    num\_workers INT NOT NULL CHECK (num\_workers > 0),

    type\_id INT NOT NULL,

    CONSTRAINT fk\_outlet\_type FOREIGN KEY (type\_id)

        REFERENCES outlet\_types(id) ON DELETE RESTRICT

);

*-- Филиалы*

CREATE TABLE branches (

    outlet\_id INT NOT NULL UNIQUE PRIMARY KEY,

    CONSTRAINT fk\_branch\_outlet FOREIGN KEY (outlet\_id)

        REFERENCES outlets(id) ON DELETE CASCADE

);

*-- Киоски*

CREATE TABLE kiosks (

    outlet\_id INT NOT NULL,

    branch\_id INT NOT NULL,

    PRIMARY KEY (outlet\_id, branch\_id),

    CONSTRAINT fk\_kiosk\_outlet FOREIGN KEY (outlet\_id)

        REFERENCES outlets(id) ON DELETE CASCADE,

    CONSTRAINT fk\_kiosk\_branch FOREIGN KEY (branch\_id)

        REFERENCES branches(outlet\_id) ON DELETE CASCADE

);

*-- Фотомагазины*

CREATE TABLE photo\_stores (

    outlet\_id INT NOT NULL PRIMARY KEY,

    CONSTRAINT fk\_photo\_store\_outlet FOREIGN KEY (outlet\_id)

        REFERENCES outlets(id) ON DELETE CASCADE

);

*-- Типы услуг*

CREATE TABLE service\_types (

    id SERIAL PRIMARY KEY,

    name VARCHAR(255) NOT NULL UNIQUE,

    price NUMERIC(10, 2) NOT NULL CHECK (price > 0)

);

*-- Типы услуг тороговых точек*

CREATE TABLE service\_types\_outlets (

    id SERIAL PRIMARY KEY,

    service\_type\_id INT NOT NULL,

    outlet\_type\_id INT NOT NULL,

    CONSTRAINT fk\_sto\_service\_type FOREIGN KEY (service\_type\_id)

        REFERENCES service\_types(id) ON DELETE CASCADE,

    CONSTRAINT fk\_sto\_outlet\_type FOREIGN KEY (outlet\_type\_id)

        REFERENCES outlet\_types(id) ON DELETE CASCADE

);

*-- Фирмы*

CREATE TABLE firms (

    id SERIAL PRIMARY KEY,

    name VARCHAR(255) NOT NULL UNIQUE

);

*-- Товары*

CREATE TABLE items (

    id SERIAL PRIMARY KEY,

    name VARCHAR(255) NOT NULL,

    price NUMERIC(10, 2) NOT NULL CHECK (price > 0),

    firm\_id INT,

    CONSTRAINT fk\_item\_firm FOREIGN KEY (firm\_id)

        REFERENCES firms(id) ON DELETE SET NULL

);

*-- Клиенты*

CREATE TABLE clients (

    id SERIAL PRIMARY KEY,

    full\_name VARCHAR(255) NOT NULL,

    is\_professional BOOLEAN NOT NULL,

    discount NUMERIC(5, 2) NOT NULL

);

*-- Заказы*

CREATE TABLE orders (

    id SERIAL PRIMARY KEY,

    accept\_time TIMESTAMP NOT NULL DEFAULT NOW(),

    overall\_price NUMERIC(10, 2) NOT NULL,

    is\_urgent BOOLEAN NOT NULL,

    outlet\_id INT NOT NULL,

    client\_id INT NOT NULL,

    CONSTRAINT fk\_order\_outlet FOREIGN KEY (outlet\_id)

        REFERENCES outlets(id) ON DELETE CASCADE,

    CONSTRAINT fk\_order\_client FOREIGN KEY (client\_id)

        REFERENCES clients(id) ON DELETE CASCADE

);

*-- Скидки на печать*

CREATE TABLE print\_discounts (

    id SERIAL PRIMARY KEY,

    photo\_amount INT NOT NULL,

    discount NUMERIC(5, 2) NOT NULL

);

*-- Типы бумаг*

CREATE TABLE paper\_types (

    id SERIAL PRIMARY KEY,

    name VARCHAR(255) NOT NULL

);

*-- Форматы бумаг*

CREATE TABLE paper\_sizes (

    id SERIAL PRIMARY KEY,

    name VARCHAR(255) NOT NULL

);

*-- Цены на печать*

CREATE TABLE print\_prices (

    id SERIAL PRIMARY KEY,

    price NUMERIC(10, 2) NOT NULL,

    paper\_size\_id INT NOT NULL,

    paper\_type\_id INT NOT NULL,

    CONSTRAINT fk\_print\_price\_paper\_size FOREIGN KEY (paper\_size\_id)

        REFERENCES paper\_sizes(id),

    CONSTRAINT fk\_print\_price\_paper\_type FOREIGN KEY (paper\_type\_id)

        REFERENCES paper\_types(id)

);

*-- Заказы на печать*

CREATE TABLE print\_orders (

    id SERIAL PRIMARY KEY,

    order\_id INT NOT NULL,

    print\_discount\_id INT NOT NULL,

    CONSTRAINT fk\_print\_order\_order FOREIGN KEY (order\_id)

        REFERENCES orders(id) ON DELETE CASCADE,

    CONSTRAINT fk\_print\_order\_discount FOREIGN KEY (print\_discount\_id)

        REFERENCES print\_discounts(id) ON DELETE CASCADE

);

*-- Хранилища*

CREATE TABLE storages (

    id SERIAL PRIMARY KEY,

    capacity INT NOT NULL,

    outlet\_id INT NOT NULL,

    CONSTRAINT fk\_storage\_outlet FOREIGN KEY (outlet\_id)

        REFERENCES outlets(id) ON DELETE CASCADE

);

*-- Заказы услуг*

CREATE TABLE service\_orders (

    id SERIAL PRIMARY KEY,

    count INT NOT NULL,

    order\_id INT NOT NULL,

    service\_type\_id INT NOT NULL,

    CONSTRAINT fk\_service\_order\_order FOREIGN KEY (order\_id)

        REFERENCES orders(id) ON DELETE CASCADE,

    CONSTRAINT fk\_service\_order\_service\_type FOREIGN KEY (service\_type\_id)

        REFERENCES service\_types(id) ON DELETE CASCADE

);

*-- Заказы на проявку плёнок*

CREATE TABLE film\_development\_orders (

    id SERIAL PRIMARY KEY,

    service\_order\_id INT NOT NULL,

    code VARCHAR(255) NOT NULL,

    CONSTRAINT fk\_film\_dev\_service\_order FOREIGN KEY (service\_order\_id)

        REFERENCES service\_orders(id) ON DELETE CASCADE

);

*-- Плёнки*

CREATE TABLE films (

    id SERIAL PRIMARY KEY,

    code VARCHAR(255) NOT NULL,

    service\_order\_id INT NOT NULL,

    CONSTRAINT fk\_film\_service\_order FOREIGN KEY (service\_order\_id)

        REFERENCES service\_orders(id) ON DELETE CASCADE

);

*-- Поставщики*

CREATE TABLE vendors (

    id SERIAL PRIMARY KEY,

    name VARCHAR(255) NOT NULL

);

*-- Товары поставщика*

CREATE TABLE vendor\_items (

    id SERIAL PRIMARY KEY,

    price NUMERIC(10, 2) NOT NULL,

    quantity INT NOT NULL,

    vendor\_id INT NOT NULL,

    item\_id INT NOT NULL,

    CONSTRAINT fk\_vendor\_item\_vendor FOREIGN KEY (vendor\_id)

        REFERENCES vendors(id) ON DELETE CASCADE,

    CONSTRAINT fk\_vendor\_item\_item FOREIGN KEY (item\_id)

        REFERENCES items(id) ON DELETE CASCADE

);

*-- Кадры*

CREATE TABLE frames (

    id SERIAL PRIMARY KEY,

    amount INT NOT NULL,

    frame\_number INT NOT NULL,

    print\_order\_id INT NOT NULL,

    print\_price\_id INT NOT NULL,

    CONSTRAINT fk\_frame\_print\_order FOREIGN KEY (print\_order\_id)

        REFERENCES print\_orders(id),

    CONSTRAINT fk\_frame\_print\_price FOREIGN KEY (print\_price\_id)

        REFERENCES print\_prices(id)

);

*-- Доставки*

CREATE TABLE deliveries (

    id SERIAL PRIMARY KEY,

    date DATE NOT NULL,

    storage\_id INT NOT NULL,

    vendor\_id INT NOT NULL,

    CONSTRAINT fk\_delivery\_storage FOREIGN KEY (storage\_id)

        REFERENCES storages(id),

    CONSTRAINT fk\_delivery\_vendor FOREIGN KEY (vendor\_id)

        REFERENCES vendors(id)

);

*-- Товары доставки*

CREATE TABLE delivery\_items (

    id SERIAL PRIMARY KEY,

    price NUMERIC(10, 2) NOT NULL,

        quantity INT NOT NULL,

    delivery\_id INT NOT NULL,

    item\_id INT NOT NULL,

    CONSTRAINT fk\_delivery\_item\_delivery FOREIGN KEY (delivery\_id)

        REFERENCES deliveries(id),

    CONSTRAINT fk\_delivery\_item\_item FOREIGN KEY (item\_id)

        REFERENCES items(id)

);

*-- Товары необходимые для типов услуг*

CREATE TABLE service\_types\_needed\_items (

    item\_id INT NOT NULL,

    service\_type\_id INT NOT NULL,

    count INT NOT NULL,

    PRIMARY KEY (item\_id, service\_type\_id),

    CONSTRAINT fk\_stni\_item FOREIGN KEY (item\_id)

        REFERENCES items(id),

    CONSTRAINT fk\_stni\_service\_type FOREIGN KEY (service\_type\_id)

        REFERENCES service\_types(id)

);

*-- Товары хранилища*

CREATE TABLE storage\_items (

    id SERIAL PRIMARY KEY,

    quantity INT NOT NULL,

    item\_id INT NOT NULL,

    storage\_id INT NOT NULL,

    CONSTRAINT fk\_storage\_item\_storage FOREIGN KEY (storage\_id)

        REFERENCES storages(id),

    CONSTRAINT fk\_storage\_item\_item FOREIGN KEY (item\_id)

        REFERENCES items(id)

);

## **SQL скрипты создания триггеров и хранимых процедур**

*-- Функция для пересчета overall\_price в таблице orders*

CREATE OR REPLACE FUNCTION trg\_recalculate\_order\_overall\_price()

RETURNS TRIGGER AS $$

DECLARE

    v\_order\_id INT;

    v\_new\_overall\_price NUMERIC(10, 2);

    v\_client\_discount NUMERIC(5, 2);

    v\_is\_urgent BOOLEAN;

    v\_film\_development\_service\_type\_id INT;

    v\_outlet\_id INT;

    v\_film\_price\_for\_outlet NUMERIC(10, 2);

    v\_is\_film\_bought\_in\_outlet BOOLEAN;

    v\_film\_service\_order\_id INT;

    v\_film\_code VARCHAR(255);

BEGIN

*-- Определяем order\_id в зависимости от таблицы, вызвавшей триггер*

    IF TG\_TABLE\_NAME = 'service\_orders' THEN

        v\_order\_id := COALESCE(NEW.order\_id, OLD.order\_id);

    ELSIF TG\_TABLE\_NAME = 'print\_orders' THEN

        v\_order\_id := COALESCE(NEW.order\_id, OLD.order\_id);

    ELSIF TG\_TABLE\_NAME = 'frames' THEN

        SELECT po.order\_id INTO v\_order\_id FROM print\_orders po WHERE po.id = COALESCE(NEW.print\_order\_id, OLD.print\_order\_id);

    ELSIF TG\_TABLE\_NAME = 'clients' THEN

*-- Для клиентов, нам нужно найти все их заказы и пересчитать overall\_price*

*-- Это может быть дорого для большого количества заказов.*

*-- Для демонстрации, мы пересчитываем только для одного заказа,*

*-- но в реальной системе может потребоваться асинхронная обработка или более сложная логика.*

        IF TG\_OP = 'UPDATE' AND OLD.discount IS DISTINCT FROM NEW.discount THEN

            FOR v\_order\_id IN (SELECT id FROM orders WHERE client\_id = NEW.id) LOOP

                PERFORM trg\_recalculate\_order\_overall\_price\_for\_order(v\_order\_id);

            END LOOP;

        END IF;

        RETURN NEW;

    ELSIF TG\_TABLE\_NAME = 'print\_discounts' THEN

*-- Для скидок на печать, нужно найти все заказы, использующие эту скидку*

        IF TG\_OP = 'UPDATE' AND OLD.discount IS DISTINCT FROM NEW.discount THEN

            FOR v\_order\_id IN (SELECT po.order\_id FROM print\_orders po WHERE po.print\_discount\_id = NEW.id) LOOP

                PERFORM trg\_recalculate\_order\_overall\_price\_for\_order(v\_order\_id);

            END LOOP;

        END IF;

        RETURN NEW;

    END IF;

*-- Если order\_id не был определен (например, для clients или print\_discounts,*

*-- которые обрабатываются отдельно или через вспомогательную функцию)*

    IF v\_order\_id IS NULL THEN

        RETURN NEW;

    END IF;

*-- Вызов вспомогательной функции для пересчета конкретного заказа*

    PERFORM trg\_recalculate\_order\_overall\_price\_for\_order(v\_order\_id);

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION trg\_recalculate\_order\_overall\_price\_for\_order(p\_order\_id INT)

RETURNS VOID AS $$

DECLARE

    v\_new\_overall\_price NUMERIC(10, 2);

    v\_client\_discount NUMERIC(5, 2);

    v\_is\_urgent BOOLEAN;

    v\_outlet\_id INT;

    v\_film\_development\_service\_type\_id INT;

    v\_film\_price\_for\_outlet NUMERIC(10, 2);

    v\_is\_film\_bought\_in\_outlet BOOLEAN;

    v\_film\_service\_order\_id INT;

    v\_film\_item\_id INT;

    v\_order\_record RECORD; *-- Добавляем переменную для записи заказа*

BEGIN

*-- Получаем текущие данные заказа и БЛОКИРУЕМ строку orders для обновления*

    SELECT o.is\_urgent, o.outlet\_id, c.discount

    INTO v\_is\_urgent, v\_outlet\_id, v\_client\_discount

    FROM orders o

    JOIN clients c ON o.client\_id = c.id

    WHERE o.id = p\_order\_id

    FOR UPDATE OF o; *-- Блокируем строку в таблице orders*

*-- Инициализируем новую общую стоимость*

    v\_new\_overall\_price := 0;

    SELECT COALESCE(SUM(so.count \* st.price \* CASE WHEN v\_is\_urgent THEN 2.0 ELSE 1.0 END), 0)

    INTO v\_new\_overall\_price

    FROM service\_orders so

    JOIN service\_types st ON so.service\_type\_id = st.id

    WHERE so.order\_id = p\_order\_id;

*-- Логика учета бесплатной проявки пленки*

*-- Аналогично, если films или delivery\_items могут быть изменены, подумайте о FOR UPDATE.*

    SELECT id INTO v\_film\_development\_service\_type\_id

    FROM service\_types

    WHERE name = 'Проявка пленки';

    IF v\_film\_development\_service\_type\_id IS NOT NULL THEN

        FOR v\_film\_service\_order\_id, v\_film\_item\_id IN (

            SELECT so\_fd.id, f.item\_id

            FROM service\_orders so\_fd

            JOIN films f ON so\_fd.id = f.service\_order\_id

            WHERE so\_fd.order\_id = p\_order\_id

              AND so\_fd.service\_type\_id = v\_film\_development\_service\_type\_id

        ) LOOP

            IF v\_film\_item\_id IS NOT NULL THEN

                SELECT EXISTS (

                    SELECT 1

                    FROM delivery\_items di

                    JOIN deliveries d ON di.delivery\_id = d.id

                    JOIN storages s ON d.storage\_id = s.id

                    WHERE s.outlet\_id = v\_outlet\_id

                      AND di.item\_id = v\_film\_item\_id

                ) INTO v\_is\_film\_bought\_in\_outlet;

                IF v\_is\_film\_bought\_in\_outlet THEN

                    SELECT price INTO v\_film\_price\_for\_outlet

                    FROM service\_types

                    WHERE id = v\_film\_development\_service\_type\_id;

                    v\_new\_overall\_price := v\_new\_overall\_price - COALESCE(v\_film\_price\_for\_outlet, 0);

                END IF;

            END IF;

        END LOOP;

    END IF;

*-- Добавляем стоимость печати с учетом скидок*

    SELECT v\_new\_overall\_price + COALESCE(SUM(

        f.amount \* pp.price \* (1 - COALESCE(pd.discount, 0) / 100.0) \* (1 - COALESCE(v\_client\_discount, 0) / 100.0)

    ), 0)

    INTO v\_new\_overall\_price

    FROM print\_orders po

    JOIN frames f ON po.id = f.print\_order\_id

    JOIN print\_prices pp ON f.print\_price\_id = pp.id

    LEFT JOIN print\_discounts pd ON po.print\_discount\_id = pd.id

    WHERE po.order\_id = p\_order\_id;

*-- Обновляем overall\_price в таблице orders*

*-- Эта UPDATE будет использовать уже полученную блокировку*

    UPDATE orders

    SET overall\_price = v\_new\_overall\_price

    WHERE id = p\_order\_id;

END;

$$ LANGUAGE plpgsql;

*-- Триггеры AFTER INSERT, UPDATE, DELETE на service\_orders*

CREATE TRIGGER trg\_after\_service\_orders\_change

AFTER INSERT OR UPDATE OR DELETE ON service\_orders

FOR EACH ROW

EXECUTE FUNCTION trg\_recalculate\_order\_overall\_price();

*-- Триггеры AFTER INSERT, UPDATE, DELETE на print\_orders*

CREATE TRIGGER trg\_after\_print\_orders\_change

AFTER INSERT OR UPDATE OR DELETE ON print\_orders

FOR EACH ROW

EXECUTE FUNCTION trg\_recalculate\_order\_overall\_price();

*-- Триггеры AFTER INSERT, UPDATE, DELETE на frames*

CREATE TRIGGER trg\_after\_frames\_change

AFTER INSERT OR UPDATE OR DELETE ON frames

FOR EACH ROW

EXECUTE FUNCTION trg\_recalculate\_order\_overall\_price();

*-- Триггер AFTER UPDATE на clients для изменения скидок клиентов*

CREATE TRIGGER trg\_after\_clients\_discount\_change

AFTER UPDATE OF discount ON clients

FOR EACH ROW

WHEN (OLD.discount IS DISTINCT FROM NEW.discount)

EXECUTE FUNCTION trg\_recalculate\_order\_overall\_price();

*-- Триггер AFTER UPDATE на print\_discounts для изменения скидок на печать*

CREATE TRIGGER trg\_after\_print\_discounts\_change

AFTER UPDATE OF discount ON print\_discounts

FOR EACH ROW

WHEN (OLD.discount IS DISTINCT FROM NEW.discount)

EXECUTE FUNCTION trg\_recalculate\_order\_overall\_price();

*-- Триггер для поддержания актуального количества товаров на складе*

*-- Функция для обновления количества на складе*

CREATE OR REPLACE FUNCTION trg\_update\_storage\_quantity()

RETURNS TRIGGER AS $$

DECLARE

    v\_storage\_id INT;

    v\_item\_id INT;

    v\_quantity\_change INT;

    v\_order\_outlet\_id INT;

BEGIN

    IF TG\_TABLE\_NAME = 'delivery\_items' THEN

*-- Поступление товаров*

        SELECT d.storage\_id INTO v\_storage\_id FROM deliveries d WHERE d.id = NEW.delivery\_id;

        v\_item\_id := NEW.item\_id;

        v\_quantity\_change := NEW.quantity;

        INSERT INTO storage\_items (quantity, item\_id, storage\_id)

        VALUES (v\_quantity\_change, v\_item\_id, v\_storage\_id)

        ON CONFLICT (item\_id, storage\_id) DO UPDATE

        SET quantity = storage\_items.quantity + EXCLUDED.quantity;

    ELSIF TG\_TABLE\_NAME = 'service\_orders' THEN

*-- Использование товаров для услуг*

*-- Определяем storage\_id, связанный с outlet\_id заказа*

        SELECT o.outlet\_id INTO v\_order\_outlet\_id FROM orders o WHERE o.id = NEW.order\_id;

        SELECT s.id INTO v\_storage\_id FROM storages s WHERE s.outlet\_id = v\_order\_outlet\_id;

        IF v\_storage\_id IS NULL THEN

            RAISE EXCEPTION 'Не найдено хранилище для торговой точки заказа ID: %', v\_order\_outlet\_id;

        END IF;

*-- Определяем товары, необходимые для данной service\_type*

        FOR v\_item\_id, v\_quantity\_change IN

            SELECT stni.item\_id, stni.count \* NEW.count

            FROM service\_types\_needed\_items stni

            WHERE stni.service\_type\_id = NEW.service\_type\_id

        LOOP

            UPDATE storage\_items

            SET quantity = quantity - v\_quantity\_change

            WHERE item\_id = v\_item\_id AND storage\_id = v\_storage\_id;

*-- Удаляем запись, если количество стало 0 или меньше (опционально)*

            DELETE FROM storage\_items

            WHERE item\_id = v\_item\_id AND storage\_id = v\_storage\_id AND quantity <= 0;

        END LOOP;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

*-- Создаем UNIQUE ограничение для (item\_id, storage\_id) в storage\_items, если его нет*

*-- Это необходимо для корректной работы ON CONFLICT в триггере*

DO $$ BEGIN

    IF NOT EXISTS (SELECT 1 FROM pg\_constraint WHERE conname = 'uq\_storage\_items\_item\_storage') THEN

        ALTER TABLE storage\_items ADD CONSTRAINT uq\_storage\_items\_item\_storage UNIQUE (item\_id, storage\_id);

    END IF;

END $$;

CREATE TRIGGER trg\_after\_delivery\_items\_change

AFTER INSERT OR UPDATE ON delivery\_items

FOR EACH ROW

EXECUTE FUNCTION trg\_update\_storage\_quantity();

CREATE TRIGGER trg\_after\_service\_orders\_items\_use

AFTER INSERT OR UPDATE ON service\_orders

FOR EACH ROW

EXECUTE FUNCTION trg\_update\_storage\_quantity();

*-- Триггер для предотвращения превышения емкости склада*

CREATE OR REPLACE FUNCTION trg\_check\_storage\_capacity()

RETURNS TRIGGER AS $$

DECLARE

    v\_current\_total\_quantity INT;

    v\_new\_total\_quantity INT;

    v\_storage\_capacity INT;

    v\_item\_id INT;

    v\_storage\_id INT;

    v\_quantity\_change INT;

BEGIN

    IF TG\_OP = 'INSERT' THEN

        v\_item\_id := NEW.item\_id;

        v\_storage\_id := NEW.storage\_id;

        v\_quantity\_change := NEW.quantity;

    ELSIF TG\_OP = 'UPDATE' THEN

        v\_item\_id := NEW.item\_id;

        v\_storage\_id := NEW.storage\_id;

        v\_quantity\_change := NEW.quantity - OLD.quantity;

    ELSE

*-- Для DELETE нет необходимости проверять емкость*

        RETURN OLD;

    END IF;

    SELECT capacity INTO v\_storage\_capacity FROM storages WHERE id = v\_storage\_id;

*-- Суммарное количество товаров на данном складе, исключая текущий item\_id для UPDATE*

    SELECT COALESCE(SUM(quantity), 0)

    INTO v\_current\_total\_quantity

    FROM storage\_items

    WHERE storage\_id = v\_storage\_id

      AND (TG\_OP <> 'UPDATE' OR item\_id <> OLD.item\_id); *-- Исключаем OLD.quantity для данного item\_id при UPDATE*

    v\_new\_total\_quantity := v\_current\_total\_quantity + v\_quantity\_change;

    IF v\_new\_total\_quantity > v\_storage\_capacity THEN

        RAISE EXCEPTION 'Превышена емкость склада (ID: %). Допустимая емкость: %, текущая + новое количество: %', v\_storage\_id, v\_storage\_capacity, v\_new\_total\_quantity;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_before\_storage\_items\_insert\_update

BEFORE INSERT OR UPDATE ON storage\_items

FOR EACH ROW

EXECUTE FUNCTION trg\_check\_storage\_capacity();

*-- Триггер для установки скидки профессиональным клиентам*

CREATE OR REPLACE FUNCTION trg\_set\_professional\_discount()

RETURNS TRIGGER AS $$

BEGIN

    IF NEW.is\_professional IS TRUE AND (OLD.is\_professional IS DISTINCT FROM NEW.is\_professional OR COALESCE(NEW.discount, 0) = 0) THEN

        NEW.discount := 10.00; *-- Устанавливаем скидку по умолчанию для профессионалов*

    ELSIF NEW.is\_professional IS FALSE AND OLD.is\_professional IS TRUE THEN

        NEW.discount := 0.00; *-- Сбрасываем скидку, если клиент перестает быть профессионалом*

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_before\_clients\_update\_professional

BEFORE INSERT OR UPDATE OF is\_professional, discount ON clients

FOR EACH ROW

EXECUTE FUNCTION trg\_set\_professional\_discount();

*-- Триггер для обеспечения приема срочных заказов только в филиалах*

CREATE OR REPLACE FUNCTION trg\_enforce\_urgent\_orders\_at\_branches()

RETURNS TRIGGER AS $$

DECLARE

    v\_outlet\_type\_name VARCHAR(255);

BEGIN

    IF NEW.is\_urgent IS TRUE THEN

        SELECT ot.name

        INTO v\_outlet\_type\_name

        FROM outlets o

        JOIN outlet\_types ot ON o.type\_id = ot.id

        WHERE o.id = NEW.outlet\_id;

        IF v\_outlet\_type\_name IS NULL OR v\_outlet\_type\_name <> 'branch' THEN

            RAISE EXCEPTION 'Срочные заказы могут быть приняты только в филиалах. Текущий тип торговой точки: %', COALESCE(v\_outlet\_type\_name, 'Неизвестен');

        END IF;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_before\_orders\_insert\_update\_urgent

BEFORE INSERT OR UPDATE OF is\_urgent, outlet\_id ON orders

FOR EACH ROW

EXECUTE FUNCTION trg\_enforce\_urgent\_orders\_at\_branches();

*---*

*-- Простые хранимые процедуры (функции)*

*---*

*-- 1. Получить цену услуги по ее ID*

CREATE OR REPLACE FUNCTION sp\_get\_service\_price(

    p\_service\_type\_id INT

)

RETURNS NUMERIC(10, 2) AS $$

DECLARE

    v\_price NUMERIC(10, 2);

BEGIN

    SELECT price INTO v\_price FROM service\_types WHERE id = p\_service\_type\_id;

    RETURN v\_price;

END;

$$ LANGUAGE plpgsql;

*-- 2. Получить ID типа торговой точки по ее названию*

CREATE OR REPLACE FUNCTION sp\_get\_outlet\_type\_id(

    p\_outlet\_type\_name VARCHAR(255)

)

RETURNS INT AS $$

DECLARE

    v\_id INT;

BEGIN

    SELECT id INTO v\_id FROM outlet\_types WHERE name = p\_outlet\_type\_name;

    RETURN v\_id;

END;

$$ LANGUAGE plpgsql;

*-- 3. Проверить, была ли пленка с данным кодом куплена в указанной торговой точке*

*-- (Предполагается, что 'items.name' может хранить 'films.code' для отслеживания покупок)*

CREATE OR REPLACE FUNCTION sp\_check\_film\_bought\_in\_outlet(

    p\_film\_code VARCHAR(255),

    p\_outlet\_id INT

)

RETURNS BOOLEAN AS $$

DECLARE

    is\_bought BOOLEAN;

BEGIN

    SELECT EXISTS (

        SELECT 1

        FROM delivery\_items di

        JOIN deliveries d ON di.delivery\_id = d.id

        JOIN storages s ON d.storage\_id = s.id

        JOIN items i ON di.item\_id = i.id

        WHERE s.outlet\_id = p\_outlet\_id

        AND i.name = p\_film\_code *-- Предполагаем, что name в items может быть кодом пленки*

    ) INTO is\_bought;

    RETURN is\_bought;

END;

$$ LANGUAGE plpgsql;

*-- 4. Получить текущую скидку клиента по его ID*

CREATE OR REPLACE FUNCTION sp\_get\_client\_discount(

    p\_client\_id INT

)

RETURNS NUMERIC(5, 2) AS $$

DECLARE

    v\_discount NUMERIC(5, 2);

BEGIN

    SELECT discount INTO v\_discount FROM clients WHERE id = p\_client\_id;

    RETURN v\_discount;

END;

$$ LANGUAGE plpgsql;

## **SQL скрипты запросов**

1. Получить перечень и общее число пунктов приема заказов на фотоработы по филиалам, по киоскам приема заказов, в целом по фотоцентру.

*-- Филиалы*

SELECT b.outlet\_id, o.address, ot.name AS outlet\_type

FROM branches b

JOIN outlets o ON b.outlet\_id = o.id

JOIN outlet\_types ot ON o.type\_id = ot.id;

*-- Киоски*

SELECT k.outlet\_id, o.address, ot.name AS outlet\_type, k.branch\_id

FROM kiosks k

JOIN outlets o ON k.outlet\_id = o.id

JOIN outlet\_types ot ON o.type\_id = ot.id;

*-- Все пункты приема заказов*

SELECT o.id AS outlet\_id, o.address, ot.name AS outlet\_type

FROM outlets o

JOIN outlet\_types ot ON o.type\_id = ot.id;

*-- Общее число пунктов приема заказов (по фотоцентру)*

SELECT COUNT(\*) AS total\_order\_points FROM outlets;

1. Получить перечень и общее число заказов на фотоработы по филиалам, киоскам приема заказов, в целом по фотоцентру, поступивших в течение некоторого периода времени.

*-- Перечень и общее число заказов на фотоработы по филиалам*

SELECT

  b.outlet\_id AS branch\_outlet\_id,

  COUNT(o.id) AS orders\_count

FROM branches b

LEFT JOIN orders o ON o.outlet\_id = b.outlet\_id

  AND o.accept\_time BETWEEN '2024-05-20 10:00:00'::timestamp AND '2024-05-22 16:45:00'::timestamp

GROUP BY b.outlet\_id

ORDER BY b.outlet\_id;

*-- Перечень и общее число заказов по киоскам приема заказов*

SELECT

  k.outlet\_id AS kiosk\_outlet\_id,

  COUNT(o.id) AS orders\_count

FROM kiosks k

LEFT JOIN orders o ON o.outlet\_id = k.outlet\_id

  AND o.accept\_time BETWEEN '2024-05-20 10:00:00'::timestamp AND '2024-05-22 16:45:00'::timestamp

GROUP BY k.outlet\_id

ORDER BY k.outlet\_id;

*-- Общее число заказов по фотоцентру за период*

SELECT

  COUNT(\*) AS total\_orders

FROM orders

WHERE accept\_time BETWEEN '2024-05-20 10:00:00'::timestamp AND '2024-05-22 16:45:00'::timestamp;

1. Получить перечень и общее число заказов (отдельно простых и срочных) на отдельные виды фоторабот по указанному филиалу, киоску приема заказов, поступивших в течение некоторого периода времени.

WITH filtered\_orders AS (

    SELECT o.\*, so.service\_type\_id, so.count, so.id AS service\_order\_id

    FROM orders o

    LEFT JOIN service\_orders so ON o.id = so.order\_id

    WHERE o.accept\_time BETWEEN '2024-05-20 10:00:00' AND '2024-05-22 16:45:00'

      AND o.outlet\_id IN (1, 3)

)

SELECT

    fo.service\_type\_id,

    st.name AS service\_name,

    fo.is\_urgent,

    COUNT(DISTINCT fo.id) AS orders\_count

FROM filtered\_orders fo

LEFT JOIN service\_types st ON fo.service\_type\_id = st.id

GROUP BY fo.service\_type\_id, st.name, fo.is\_urgent

ORDER BY st.name, fo.is\_urgent;

1. Получить сумму выручки с заказов (отдельно простых и срочных) на отдельные виды фоторабот по указанному филиалу, киоску приема заказов, поступивших в течение некоторого периода времени.

CREATE OR REPLACE VIEW filtered\_orders AS

SELECT o.\*

FROM orders o

WHERE o.accept\_time BETWEEN '2024-05-20 10:00:00'::timestamp AND '2024-05-22 16:45:00'::timestamp

  AND (o.outlet\_id = 1 OR o.outlet\_id = 3);

CREATE OR REPLACE VIEW service\_order\_sums AS

SELECT o.is\_urgent, so.service\_type\_id, SUM(o.overall\_price) AS revenue

FROM filtered\_orders o

JOIN service\_orders so ON o.id = so.order\_id

GROUP BY o.is\_urgent, so.service\_type\_id;

SELECT

    sos.service\_type\_id,

    st.name AS service\_name,

    sos.is\_urgent,

    sos.revenue

FROM service\_order\_sums sos

JOIN service\_types st ON sos.service\_type\_id = st.id

ORDER BY st.name, sos.is\_urgent;

1. Получить количество отпечатанных фотографий в рамках простых и срочных заказов по указанному филиалу, киоску приема заказов, фотоцентру в целом за некоторый период времени.

*-- Запрос для подсчёта отпечатанных фотографий по филиалу (outlet\_id из branches):*

SELECT

    o.is\_urgent,

    SUM(f.amount) AS total\_printed\_photos

FROM frames f

JOIN print\_orders po ON f.print\_order\_id = po.id

JOIN orders o ON po.order\_id = o.id

WHERE o.accept\_time BETWEEN '2024-01-01 10:00:00'::timestamp

                      AND '2024-12-30 16:45:00'::timestamp

  AND o.outlet\_id IN (SELECT outlet\_id FROM branches)

GROUP BY o.is\_urgent;

*--Запрос для киоска (outlet\_id из kiosks):*

SELECT

    o.is\_urgent,

    SUM(f.amount) AS total\_printed\_photos

FROM frames f

JOIN print\_orders po ON f.print\_order\_id = po.id

JOIN orders o ON po.order\_id = o.id

WHERE o.accept\_time BETWEEN '2024-01-01 10:00:00'::timestamp

                      AND '2024-12-30 16:45:00'::timestamp

  AND o.outlet\_id IN (SELECT outlet\_id FROM kiosks)

GROUP BY o.is\_urgent;

*-- Запрос по всему фотоцентру (без фильтра):*

SELECT

    o.is\_urgent,

    SUM(f.amount) AS total\_printed\_photos

FROM frames f

JOIN print\_orders po ON f.print\_order\_id = po.id

JOIN orders o ON po.order\_id = o.id

WHERE o.accept\_time BETWEEN '2024-01-01 10:00:00'::timestamp

                      AND '2024-12-30 16:45:00'::timestamp

GROUP BY o.is\_urgent;

1. Получить количество проявленных фотопленок в рамках простых и срочных заказов по указанному филиалу, киоску приема заказов, фотоцентру в целом за некоторый период времени.

*-- филиал (outlet\_id = 1)*

SELECT

    o.is\_urgent,

    COUNT(f.id) AS total\_films

FROM films f

JOIN service\_orders so ON f.service\_order\_id = so.id

JOIN orders o ON o.id = so.order\_id

WHERE

    o.accept\_time BETWEEN '2024-01-01 10:00:00'::timestamp AND '2024-12-30 16:45:00'::timestamp

    AND o.outlet\_id IN (SELECT outlet\_id FROM branches WHERE outlet\_id = 1)

GROUP BY o.is\_urgent;

*-- По киоску (kiosk\_outlet\_id = 3)*

SELECT

    o.is\_urgent,

    COUNT(f.id) AS total\_films

FROM films f

JOIN service\_orders so ON f.service\_order\_id = so.id

JOIN orders o ON o.id = so.order\_id

WHERE

    o.accept\_time BETWEEN '2024-01-01 10:00:00'::timestamp AND '2024-12-30 16:45:00'::timestamp

    AND o.outlet\_id IN (SELECT outlet\_id FROM kiosks WHERE outlet\_id = 3)

GROUP BY o.is\_urgent;

1. Получить перечень поставщиков в целом по фотоцентру, поставщиков отдельных видов фототоваров, сделавших поставки в некоторый период, поставки определенного объема.

SELECT DISTINCT

    v.id AS vendor\_id,

    v.name AS vendor\_name,

    i.id AS item\_id,

    i.name AS item\_name,

    di.quantity,

    di.price,

    d.date

FROM

    vendors v

JOIN deliveries d ON d.vendor\_id = v.id

JOIN delivery\_items di ON di.delivery\_id = d.id

JOIN items i ON i.id = di.item\_id

WHERE

    d.date BETWEEN '2024-01-01 10:00:00'::timestamp AND '2024-12-30 16:45:00'::timestamp

    AND di.quantity >= 2

ORDER BY v.id;

1. Получить список клиентов в целом по фотоцентру, клиентов указанного филиала, имеющих скидки, сделавших заказы определенного объема.

SELECT DISTINCT

    c.id AS client\_id,

    c.full\_name,

    c.discount,

    o.id AS order\_id,

    o.overall\_price,

    o.outlet\_id

FROM

    clients c

JOIN orders o ON o.client\_id = c.id

WHERE

    c.discount > 0

    AND o.overall\_price >= 5  *--  если такой порог*

    AND o.outlet\_id = 3       *--  если нужен только 1 филиал*

ORDER BY c.id

1. Получить сумму выручки от реализации фототоваров в целом по фотоцентру, по указанному филиалу, проданных в течение некоторого периода времени.

SELECT

    COALESCE(SUM(stni.count \* i.price \* so.count), 0) AS total\_revenue

FROM

    orders o

JOIN service\_orders so ON so.order\_id = o.id

JOIN service\_types\_needed\_items stni ON stni.service\_type\_id = so.service\_type\_id

JOIN items i ON i.id = stni.item\_id

WHERE

    o.accept\_time BETWEEN '2024-05-20 10:00:00'::timestamp

                     AND '2024-05-22 16:45:00'::timestamp

    AND o.outlet\_id = 1;

1. Получить перечень фототоваров и фирм, их производящих, которые пользуются наибольшим спросом в целом по фотоцентру, в указанном филиале.

*-- Перечень фототоваров и фирм, производящих их,*

*-- которые пользуются наибольшим спросом по фотоцентру и филиалу с id = 1*

WITH orders\_in\_branch AS (

    SELECT o.id

    FROM orders o

    WHERE o.outlet\_id = 1

),

service\_orders\_in\_branch AS (

    SELECT so.\*

    FROM service\_orders so

    JOIN orders\_in\_branch ob ON so.order\_id = ob.id

),

items\_demand\_in\_branch AS (

    SELECT

        stni.item\_id,

        SUM(stni.count \* so.count) AS total\_quantity

    FROM service\_orders\_in\_branch so

    JOIN service\_types\_needed\_items stni ON so.service\_type\_id = stni.service\_type\_id

    GROUP BY stni.item\_id

),

items\_demand\_overall AS (

    SELECT

        stni.item\_id,

        SUM(stni.count \* so.count) AS total\_quantity

    FROM service\_orders so

    JOIN service\_types\_needed\_items stni ON so.service\_type\_id = stni.service\_type\_id

    GROUP BY stni.item\_id

)

SELECT

    i.id AS item\_id,

    i.name AS item\_name,

    f.name AS firm\_name,

    COALESCE(d\_branch.total\_quantity, 0) AS demand\_in\_branch,

    COALESCE(d\_overall.total\_quantity, 0) AS demand\_overall

FROM items i

LEFT JOIN firms f ON i.firm\_id = f.id

LEFT JOIN items\_demand\_in\_branch d\_branch ON i.id = d\_branch.item\_id

LEFT JOIN items\_demand\_overall d\_overall ON i.id = d\_overall.item\_id

WHERE COALESCE(d\_branch.total\_quantity, 0) > 0 OR COALESCE(d\_overall.total\_quantity, 0) > 0

ORDER BY demand\_overall DESC, demand\_in\_branch DESC;

1. Получить перечень реализованных фототоваров и объемы их реализации в целом по фотоцентру, по указанному филиалу, проданных в течение некоторого периода времени.

WITH filtered\_orders AS (

    SELECT o.id

    FROM orders o

    WHERE o.accept\_time BETWEEN '2024-05-20 10:00:00'::timestamp

                            AND '2024-05-22 16:45:00'::timestamp

    AND o.outlet\_id = 1

),

service\_orders\_filtered AS (

    SELECT so.\*

    FROM service\_orders so

    JOIN filtered\_orders fo ON so.order\_id = fo.id

),

items\_sold AS (

    SELECT

        stni.item\_id,

        SUM(stni.count \* so.count) AS total\_quantity

    FROM service\_orders\_filtered so

    JOIN service\_types\_needed\_items stni ON so.service\_type\_id = stni.service\_type\_id

    GROUP BY stni.item\_id

)

SELECT

    i.id AS item\_id,

    i.name AS item\_name,

    f.name AS firm\_name,

    COALESCE(items\_sold.total\_quantity, 0) AS quantity\_sold

FROM items i

LEFT JOIN firms f ON i.firm\_id = f.id

LEFT JOIN items\_sold ON i.id = items\_sold.item\_id

WHERE items\_sold.total\_quantity IS NOT NULL

ORDER BY quantity\_sold DESC;

1. Получить перечень рабочих мест фотоцентра в целом и указанного профиля.

*-- Все рабочие места (без фильтра по профилю)*

SELECT o.id AS outlet\_id, o.address, ot.name AS outlet\_type

FROM outlets o

JOIN outlet\_types ot ON o.type\_id = ot.id

ORDER BY o.id;

*-- Рабочие места с профилем 'kiosk'*

SELECT o.id AS outlet\_id, o.address, ot.name AS outlet\_type

FROM outlets o

JOIN outlet\_types ot ON o.type\_id = ot.id

WHERE ot.name = 'kiosk'

ORDER BY o.id;

## **КОНКУРЕТНЫЙ ДОСТУП К БД**

**Уровни изоляции в PostgreSQL (по умолчанию READ COMMITTED):**

* **READ COMMITTED** (по умолчанию): Каждая команда (оператор SQL) в транзакции видит только те изменения, которые были закоммичены до начала этой команды.
* **REPEATABLE READ**: Транзакция видит снимок данных, который был актуален на момент ее *первой* команды. Все последующие команды в этой транзакции будут видеть тот же снимок. Это предотвращает неповторяемое чтение и фантомное чтение, но может привести к ошибкам сериализации (could not serialize access due to concurrent update).
* **SERIALIZABLE**: Самый строгий уровень изоляции. Транзакция выполняется так, как если бы она была единственной транзакцией в системе. Это предотвращает все аномалии чтения/записи, но имеет самую высокую вероятность ошибок сериализации и самые большие накладные расходы.
* **SELECT ... FOR UPDATE**: Блокировка определенных строк.

## **ПРИНЯТЫЕ МЕРЫ**

* 1. Триггер **trg\_recalculate\_order\_overall\_price\_for\_order,** обновляющий общую стоимость заказа имеет **FOR UPDATE OF o**; для блокировки строки в таблице orders.
  2. Конкурентный доступ к складским остаткам. Для обеспечения корректной работы с количеством товаров на складе при одновременном доступе нескольких транзакций, в функции **trg\_update\_storage\_quantity** было сделано:

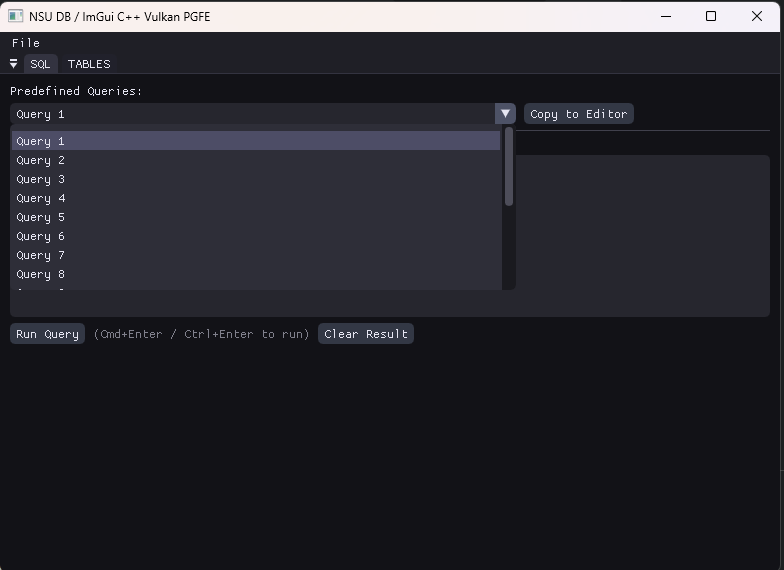
**ON CONFLICT (item\_id, storage\_id) DO UPDATE** при добавлении товаров:

При поступлении новых товаров через **delivery\_items** используется конструкция **INSERT ... ON CONFLICT DO UPDATE** - это позволяет избежать ошибок уникального ключа, если две транзакции одновременно пытаются добавить или обновить количество одного и того же товара на одном складе. Вместо того чтобы одна транзакция завершилась ошибкой, она атомарно обновляет существующую запись, гарантируя, что количество будет правильно увеличено.

## **АРХИТЕКТУРА ПРИЛОЖЕНИЯ**

Язык программирования C++, библиотека для отрисовки UI – ImGui, библиотека для взаимодействия с PostgreSQL – PGFE, библиотека для логирования – spdlog.

Подключаемся к БД, отправляем запросы, т.е. пока есть соединение можем говорить с БД, вместо переподключения каждый раз как хотим отправить запрос(такое тоже возможно из приложения если делать самому).

Пример окошка с возможностью выбрать и исполнить любой из 12 запросов(в приложении их чуть больше потому что некоторые запросы требуют получения информации по разным типам торговых точек, а это отдельные полноценные запросы).  


Пример выполненного запроса, который мы выбрали, он автоматически копируется в строку ввода SQL скриптов. 