ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

**"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по курсу

“Программирование систем с серверами баз данных”

Тема работы: «Комиссионные магазины города»

Руководители: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Щедрин С.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ногтев Е.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Филипишин Д.А.

(подпись) (дата)

Разработал: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Моргунов А.Г.

ст. гр. ПИ-18б (подпись) (дата)

РЕФЕРАТ

Пояснительная записка содержит 100 страниц, 98 рисунков, 6 источников, 7 приложения.

Объектом исследования данной курсовой работы являются комиссионные магазины города.

Цель работы – проектирование и реализация системы учета деятельности комиссионных магазинов, включающей в себя серверную и клиентскую части.

Результатом работы является многопользовательская база данных и клиентское приложение для взаимодействия с ней. Система может выполнять такие функции как: добавление, удаление, поиск записей в таблицах и справочниках БД, составление однотабличного отчета в Microsoft Excel, создание диаграммы, основанной на статистических данных, выполнение запросов различного уровня сложности. Система имеет интуитивно понятный интерактивный интерфейс. Различные роли имеют разграничение возможностей.

БАЗА ДАННЫХ, СУБД, С++, QT, SQL, КОМИССИОННЫЙ МАГАЗИН, РАБОТНИК, ПРОЦЕДУРА, ТАБЛИЦА, POSTGRESQL, БЕЗОПАСТНОСТЬ

СОДЕРЖАНИЕ

[Введение 5](#_Toc73987806)

[1 Описание предметной области, постановка задачи 6](#_Toc73987807)

[2 Система управления базой данных 7](#_Toc73987808)

[3 Обоснование выбора инструментальные средств для написания клиентской части, проектирование структуры ПО 8](#_Toc73987809)

[3.1 Не визуальные компоненты для работы с данными 9](#_Toc73987810)

[3.2 Визуальные компоненты отображения данных 10](#_Toc73987811)

[4 Проектирование базы данных в выбранной СУБД 12](#_Toc73987812)

[4.1 Проектирование концептуальной модели БД 12](#_Toc73987813)

[4.2 Создание таблиц, доменов, индексов 14](#_Toc73987814)

[4.3 Разработка триггеров 18](#_Toc73987815)

[4.4 Разработка ограничений доступа 21](#_Toc73987816)

[4.4.1 Права доступа работника (роль worker) 22](#_Toc73987817)

[4.4.2 Права доступа администратора (роль administrator) 23](#_Toc73987818)

[4.4.3 Права доступа для владельца (роль holder) 24](#_Toc73987819)

[4.5 Проектирование запросов к базе данных 24](#_Toc73987820)

[5 Разработка клиентского приложения 51](#_Toc73987821)

[5.1 Работник(worker) 53](#_Toc73987822)

[5.2 Администратор (administrator) 55](#_Toc73987823)

[5.3 Владелец (holder) 56](#_Toc73987824)

[6 Тестирование разработанной информационной системы (в т.ч. включая защиту от несанкционированного доступа, каскадное удаление) 60](#_Toc73987825)

[6.1 Ошибки ввода 60](#_Toc73987826)

[6.2 Каскадное удаление 61](#_Toc73987827)

[Заключение 63](#_Toc73987828)

[Список литературы 64](#_Toc73987829)

[Приложение А. Техническое задание 65](#_Toc73987830)

[Приложение Б. Листинг клиентского приложения 70](#_Toc73987831)

[Приложение В. Листинг серверного приложения 76](#_Toc73987832)

[Приложение Г. Антиплагиат 95](#_Toc73987833)

[Приложение Д. Руководство работника 96](#_Toc73987834)

[Приложение Е. Руководство Администратора 98](#_Toc73987835)

[Приложение Ё. Руководство владельца 100](#_Toc73987836)

# Введение

В современном мире технологии все больше и больше вливаются в жизнь людей. В том числе и в бизнес – сферу. Многие предприниматели и организации используют современные методы обработки информации. Основным средством для быстрой обработки информации являются базы данных. Это средство позволяет обрабатывать, выводить, хранить и манипулировать огромными объемами данных. Поэтому такое средство востребовано и используется повсеместно.

Система управления базами данных(СУБД) – это программный комплекс, обеспечивающий централизованное хранение данных и предоставляющий приложениям услуги по обработке данных.

Совокупность данных, хранимых под управлением СУБД, называется базой данных. В оригинальном английском варианте словосочетание data base означает «основание, состоящее из данных». Этот смысл несколько искажается в русском словосочетании «база данных». На самом деле это – фундамент, на котором строятся приложения и который состоит из данных. Действительно, данные (а, следовательно, база данных) являются очень существенной частью практически любой информационной системы. [1]

Целью разработки является создание базы данных комиссионных магазинов города с помощью СУБД PostgreSQL и создание клиентского приложение для взаимодействия с этой базой данных на языке программирования C++.

Разработанная система может применятся для учета деятельности комиссионных магазинов города.

# 1 Описание предметной области, постановка задачи

Система учета работы комиссионного магазина должна содержать данные о комитентах (ФИО, район проживания, социальное положение (рабочий, служащий, пенсионер, домохозяйка, предприниматель…), место работы, дата рождения, телефон), принимаемых товарах (номер квитанции, группа товара (посуда, одежда, обувь, игрушки,…), наименование товара, дата приема, количество единиц, цена за единицу) и о реализации товаров (товар, дата, количество проданных единиц).

Задача: разработать программный комплекс для автоматизации управления предметной областью. Разработать роли: работник, администратор, владелец сети магазинов.

Работник может просматривать товары в своем магазине, добавлять новые товары, реализовывать товары, добавлять новых клиентов.

Администратор – это управляющая роль. Он имеет такие возможности как добавление и удаление ролей (работников и владельцев), перевод работников из одного магазина в другой, увольнение работников, редактирование справочников.

Владелец может просматривать данные по всем товарам, реализациям работникам и клиентам. Также может просматривать заранее скомпонованные выборки данных (запросы), а также имеет доступ к просмотру статистики в виде диаграммы и экспорт статистики в формат MicrosoftExcel.

# 2 Система управления базой данных

В качестве СУБД был выбран PostgreSQL. PostgreSQL идеально подходит для решения задачи, т.к. имеет функциональность:

создание многопользовательской системы, в которой есть возможность разграничения прав различных пользователей;

поддерживает БД неограниченного размера;

возможность описания всей логики на сервере позволяет перенести все вычисления и обработки данных на сервер, что существенно упрощает поддержку системы и внесение в нее изменений;

существует поддержка большого количество встроенных типов, а также возможность создавать свои типы с поддержкой встраивания ограничения в тип (домены);

существует возможность оптимизировать запросы при помощи индексов в следствии анализа предметной области и выявления данных, к которым имеется большое количество запросов и по каким данным происходит сортировка и фильтрация, при помощи индексов;

партицирование позволяет задумываться о возможности масштабирования системы без потери эффективности.

Открытый исходный код позволяет не беспокоится о неожиданных исходах операций и проблем с безопасностью, поскольку все процессы, проходящие внутри системы, полностью открыты.

# 3 Обоснование выбора инструментальные средств для написания клиентской части, проектирование структуры ПО

Для разработки клиентской части приложения был выбран язык С++. Он обладает множеством преимуществ, которые выгодно выделяют его на фоне остальных языков программирования.

Во-первых, он очень быстрый. Программа, которая грамотно написана на С++ в подавляющем большинстве случаев окажется быстрее программы с тем же функционалом, написанной на другом языке программирования.

Во-вторых, он обладает завидной универсальностью. Поскольку компилятор С++ есть на каждой операционной системе, появляется возможность написать одну программу, которая сможет скомпилироваться на любой платформе.

В-третьих, до сих пор идет активная поддержка и модификация С++. Разрабатывают и выпускают новые стандарты, что позволяет языку не отставать от своих конкурентов, и даже опережать их как в функциональности, так и в удобстве написания кода.

В-четвертых, он до сих пор остается одним из самых востребованных языков программирования, потому что имеет широкие возможности, позволяющие оптимизировать программируемые системы до такого состояния, до которого невозможно добраться, используя более высокоуровневые языки программирования.

Также С++ обладает невероятно мощной библиотекой Qt. Эта библиотека позволяет создавать удобные и эффективные пользовательские интерфейсы, а также позволяет удобно связывать приложение с сервером базы данных.

3.1 Не визуальные компоненты для работы с данными

Для взаимодействия с базой данных библиотека Qt предоставляет класс QSqlQuery, который позволяет формировать запросы, принимать ответы сервера, экранировать переменные (рис 3.1). Для начала необходимо создать запрос, и ввести команду на языке SQL, затем при необходимости задаются значения. После этого запрос выполняется.

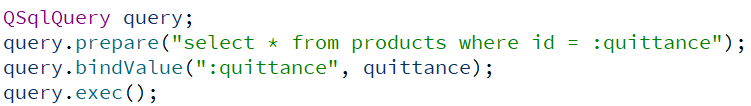


Рисунок 3.1 – Шаблон работы с запросом

Для взаимодействия с графическим интерфейсом используется класс QSqlQueryModel. Он позволяет сохранять результаты запроса, чтобы потом отобразить их пользователю (рис. 3.2).



Рисунок 3.2 – Создание новой модели и сохранение запроса, результаты которого она обдет отображать

При разработке программы было принято решение реализовать вспомогательные функции для ускорения и облегчения разработки. Одной из таких функций является sendQuery. Она позволяет в удобном виде отправлять запрос и дает на выходе курсор, который позволяет работать с результатом запроса (рис. 3.3).

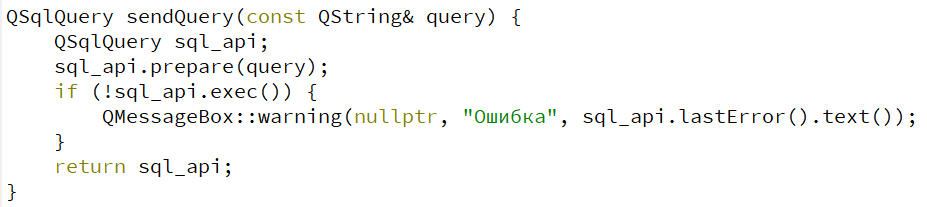


Рисунок 3.3 – Функция sendQuery

3.2 Визуальные компоненты отображения данных

Для визуализации результатов запросов используется класс QTableView. Он имеет возможность принимать модель и отображать ее содержимое. Эта особенность позволяет нам выводить данные, которые мы получили ранее, при выполнении запроса.



Рисунок 3.4 – Задать модель, которую будет отображать таблица

Помимо отображения данных в классе QTableView имеется возможность изменения отображения, что позволяет более гибко настраивать выводимую пользователю информацию.

Например, если мы не хотим показывать пользователю некоторые столбцы таблицы, однако данные в них нам еще нужны для дальнейшей работы, то мы можем просто скрыть их от пользователя, при этом использовать их в программе дальше (рис. 3.5).

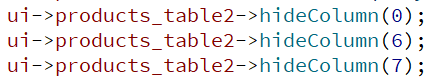


Рисунок 3.5 – Скрытие столбцов из таблицы

Для отображения неизменяемого текста используется класс QLabel. Для ввода данных пользователем используется класс QLineEdit.

Для отображения сообщений об ошибках используется класс QMessageBox (рис. 3.6)



Рисунок 3.6 – Использование всплывающего сообщения

# 4 Проектирование базы данных в выбранной СУБД

4.1 Проектирование концептуальной модели БД

Концептуальное проектирование базы данных – процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты [2].

Как правило, основной целью концептуального проектирования базы данных является описание способа физической реализации логического проекта базы данных [3].

Приступая к физическому проектированию, прежде всего, необходимо проанализировать и хорошо усвоить информацию об отношениях, собранную на этапе анализа предметной области [5].

В модели базы данных присутствуют 4 справочника:

* «shops» - справочник, содержащий идентификационный номер (id) и имя магазина (name);
* «product\_types» - справочник, содержащий идентификационный номер (id) и тип продукта (посуда, одежда и т.д.) (product\_type);
* «statuses» справочник, содержащий идентификационный номер (id) и имя социальный статус клиента (студент, пенсионер и т.д.) (status);
* «districts» справочник, содержащий идентификационный номер (id) и название района города (district).

В модели присутствуют 4 таблицы:

* «customers» - таблица, содержащая идентификационный номер клиента (id), имя (first\_name), фамилию клиента (second\_name), идентификатор района, в котором проживает клиент (id\_district), идентификатор социального статуса клиента (id\_status), место работы клиента (work), дату рождения клиента (birthday), номер телефона клиента (phone);
* «workers» - таблица, содержащая идентификационный номер клиента (id), логин работника (login), идентификатор магазина, в котором работает работник (id\_shop), его состояние (активен или уволен) (is\_available);
* «products» - таблица, содержащая идентификатор продукта (id), номер квитанции, которая была выписана при получении товара (quittance), идентификатор типа продукта (id\_product\_type), название продукта (product\_name), дату приема продукта на склад (reception), количество продукта при приеме (count), оставшееся количество продукта на складе (reception), цена за единицу продукта (price), идентификатор клиента, сдавшего этот продукт (id\_customer), идентификатор магазина, в котором хранится продукт (id\_shop), идентификатор работника, принявшего продукт на склад (id\_worker);
* «realization» - идентификатор продажи товара (id), номер чека, который был выписан при продаже товара (ticket), идентификационный номер продукта, который был куплен (id\_product), дата продажи (realization\_date), количество проданного товара (realization\_count), идентификатор работника, который продал товар (id\_worker).

После проведенного анализа и предварительного проектирования было решено остановится на варианте концептуальной модели базы данных представленной на рисунке 4.1.

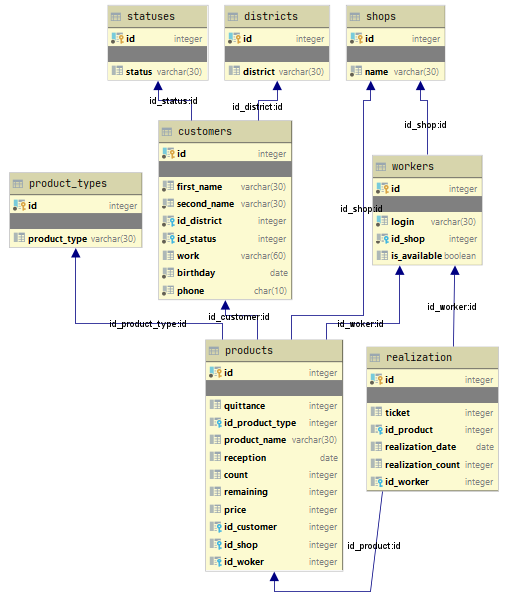


Рисунок 4.1 – Концептуальная модель базы данных

4.2 Создание таблиц, доменов, индексов

Таблицы создаются по уже готовой концептуальной модели базы данных. При создании необходимо указывать название полей, их типы, ограничения, а также можно указать внешние ключи, которые связывают разные таблицы.

При создании таблиц были использованы следующие SQL запросы. (см. рис. 4.2 – 4.9)

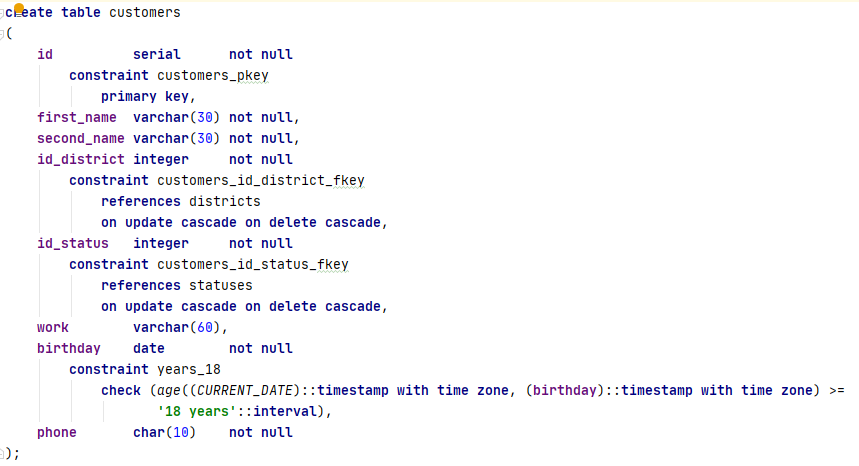


Рисунок 4.2 – Создание таблицы пользователей

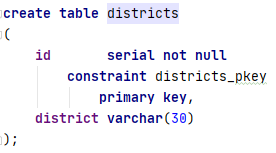


Рисунок 4.3 – Создание справочника районы

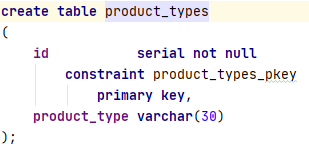


Рисунок 4.4 – Создание справочника типы продуктов

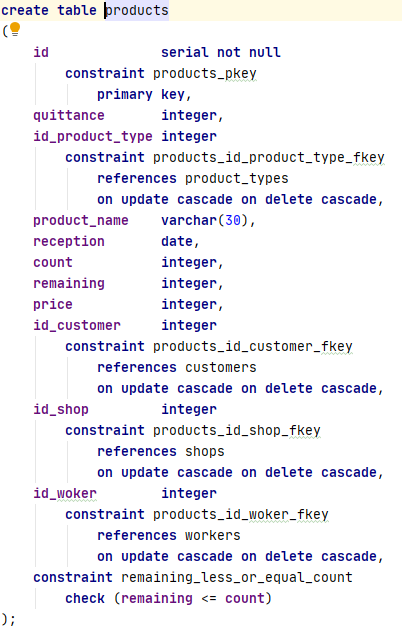


Рисунок 4.5 – Создание таблицы продуктов

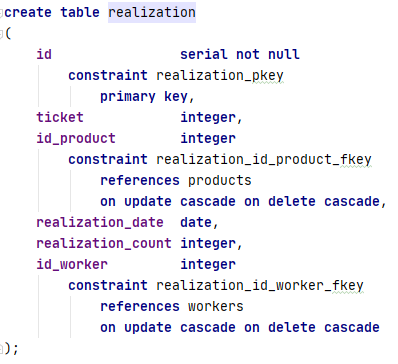


Рисунок 4.6 – Создание таблицы реализаций

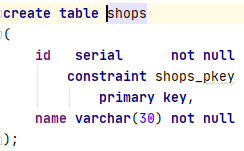


Рисунок 4.7 – Создание справочника магазинов

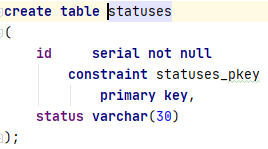


Рисунок 4.8 – Создание справочника социальных статусов

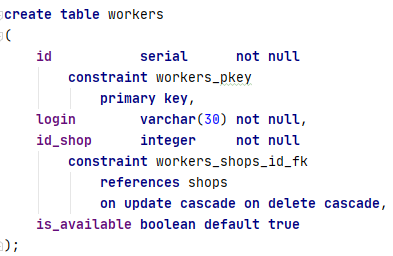


Рисунок 4.9 – Создание таблицы работников



Рисунок 4.10 – Создание домена для ввода номера телефона



Рисунок 4.11 – Создание индекса в таблице реализаций



Рисунок 4.12 – Создание индекса в таблице продуктов

4.3 Разработка триггеров

Триггеры необходимы в проекте для добавления большей логики при стандартных действиях (удаления, изменения, вставки), а также для того чтобы указать базе данных как обновлять таблицы при обновлении представлений [6]. Также триггеры позволяют реализовать ограничения, которые используют подзапросы.

В программе при помощи триггеров выполнялись различные действия с данными. Первое их применение – это проверка вводимого значения в таблицу (рис. 4.13 – 4.14)

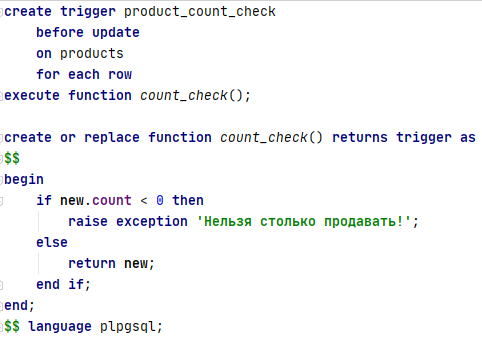


Рисунок 4.13 – Триггер проверяющие количество продаваемого товара

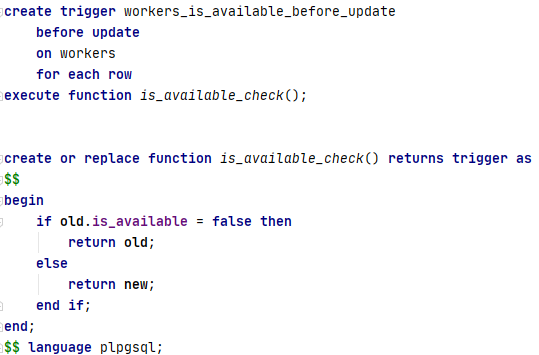


Рисунок 4.14 – Триггер, проверяющий, чтобы не изменялись данные уже уволенных сотрудников

Также триггеры применялись для обеспечения целостности данных при проведении операций, которые осуществлялись над связанными по смыслу таблицами (рис 4.15).

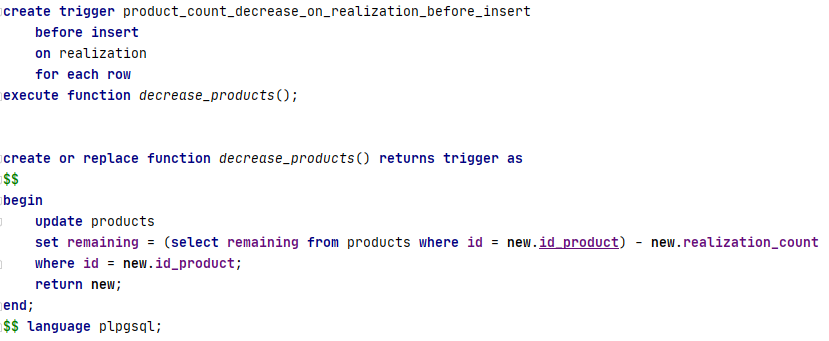


Рисунок 4.15 – Триггер, который уменьшает количество товара при его продаже

Еще одним применением триггеров стала реализация модифицируемого представления. Исходя из анализа предметной области было решено позволить работникам только вставлять данные в таблицу. Возможность изменения и удаления данных может стать инструментом в руках злоумышленников. Важным условием при создании модифицируемого представления (рис. 3.16) является указание в триггерах параметра instead of, который и позволяет реализовывать такие удобные инструменты. Создание триггера показано на рисунке 3.17.

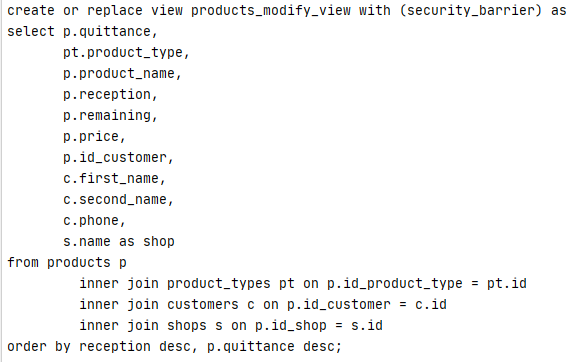


Рисунок 4.16– Создание модифицируемого представления

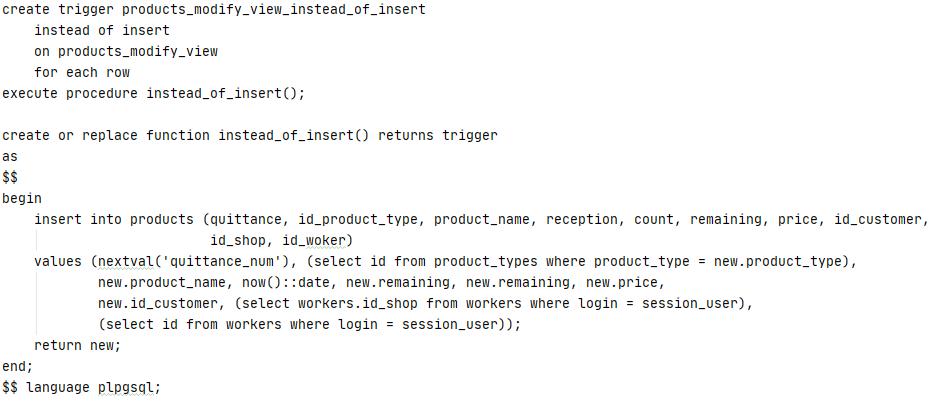


Рисунок 4.17 – Триггер, обеспечивающий взаимодействие с можифицируемым представлением

4.4 Разработка ограничений доступа

При разработке базы данных необходимо разграничить права доступа для таких ролей, как администратор, рабочий, владелец.

Для разграничения прав использовался следующий подход: предоставить grant на таблицу для роли к которой будет осуществляться доступ, далее определить политику, то есть создать row level security (RLS), если данный подход не практичен, например из-за того, что в условии RLS необходим доступ к таблицам к которым нет доступа, то создается представления view с условием where и далее дается право grant select для данной таблицы, если и этот метод не подходит, то создается функция с параметром security definer и дается grant execute на данную функцию роли.

Для того, чтобы включить защиту строк у таблцы необходимо прописть команду: alter table «имя таблицы» enable row level security.

4.4.1 Права доступа работника (роль worker)

Работник – это роль, которой необходим доступ к просмотру товаров, реализаций, заказчиков, для того, чтобы отвечать на вопросы покупателей. Также ему нужны права на последовательности и вставку в таблицы из-за того, что он должен добавлять новые товары, новых клиентов, а также реализовывать товар (рис. 4.18).

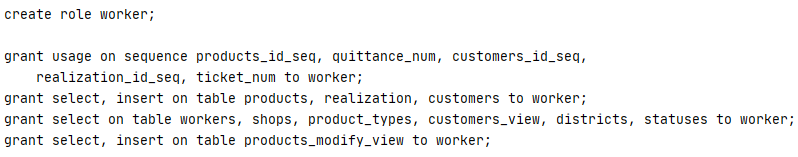


Рисунок 4.18 – Права для работника

Политики рабочего обеспечивают ему доступ к продуктам и реализациям только в своем магазине. Также в талице работников он может видеть только себя. В таблице магазинов работник видит только свой магазин (рис. 4.19).

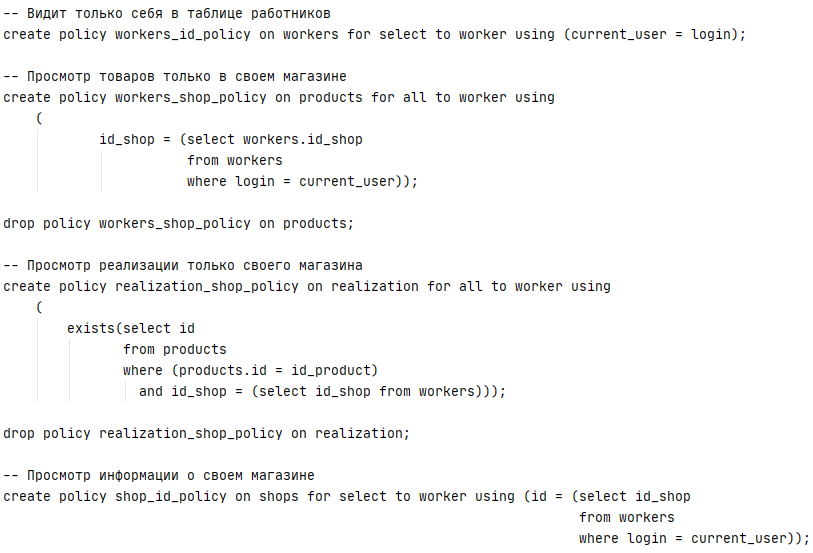


Рисунок 4.19 – Политики для работника

4.4.2 Права доступа администратора (роль administrator)

Администратор имеет доступ к добавлению и удалению в справочники, а также полный доступ к таблице работников потому, что он через него проходят все изменения среди персонала (рис.4.20). Администратор имеет доступ ко всем записям в таблицах работников и магазинов (рис. 4.21).

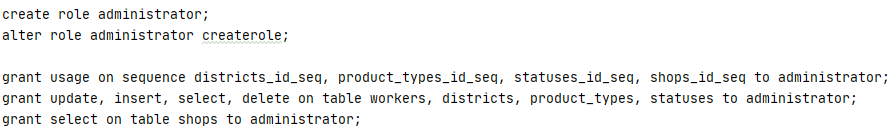


Рисунок 4.20 – Права для администратора



Рисунок 4.21 – Политики для администратора

4.4.3 Права доступа для владельца (роль holder)

Владелец имеет право смотреть на все таблицы, а также имеет доступ к запросам и статистике (рис. 4.22).

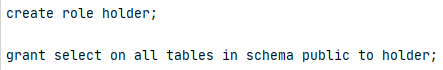


Рисунок 4.22 – Права для владельца

4.5 Проектирование запросов к базе данных

Запросы с симметричным внутренним соединением предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON.

Запрос, показывающий информацию об идентификаторе продукта, названии продукта, дате према продукта на склад, количестве оставшихся единиц товара, цене за единицу товара, имя, фамилия заказчика, название магазина, типе продукта, у продуктов, которые относятся к указанному типу продуктов. Тип продукта задает пользователь. Код предоставлен на рисунке 4.23, а результат – 4.24.

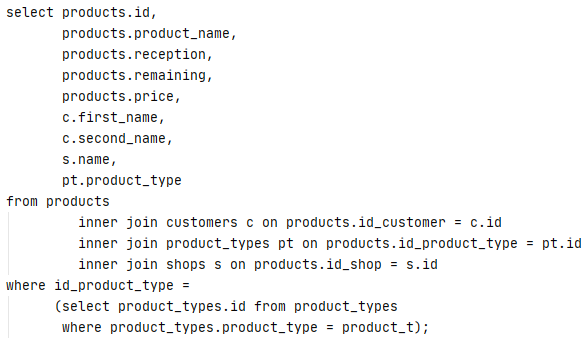


Рисунок 4.23 – Симметричное внутреннее соединение с условием по внешнему ключу

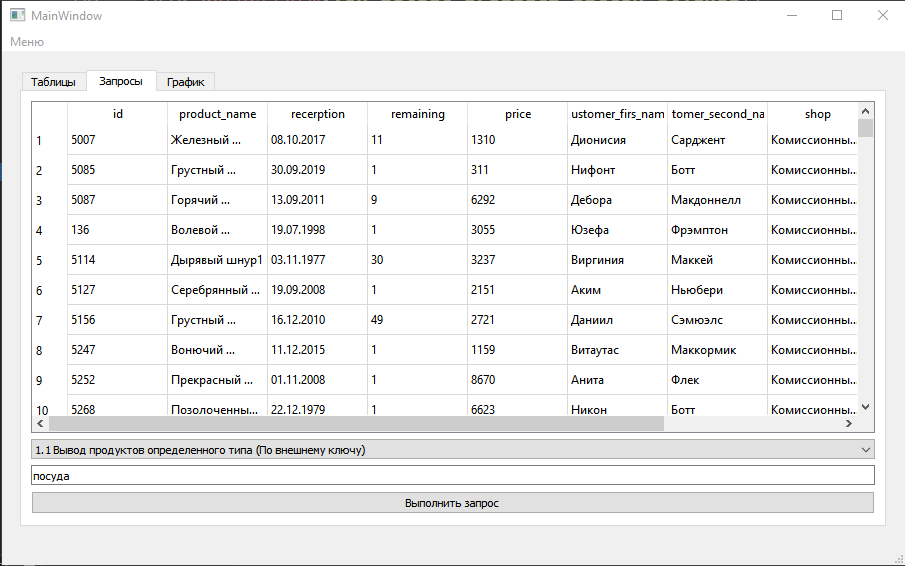


Рисунок 4.24– Продукты типа посуда

Запрос, показывающий информацию об идентификаторе пользователя, имени, фамилии заказчика, названии района его проживания, месте его работы, дате его рождения, номере его телефона, проживающего в указанном районе. Район задает пользователь. Код предоставлен на рисунке 4.25, а результат – 4.26.

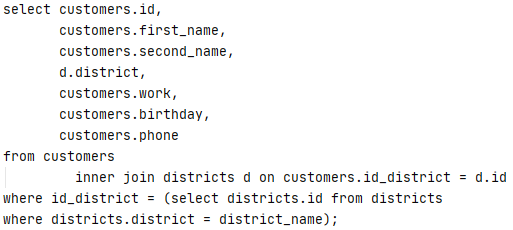


Рисунок 4.25– Симметричное внутреннее соединение с услвоием по внешнему ключу

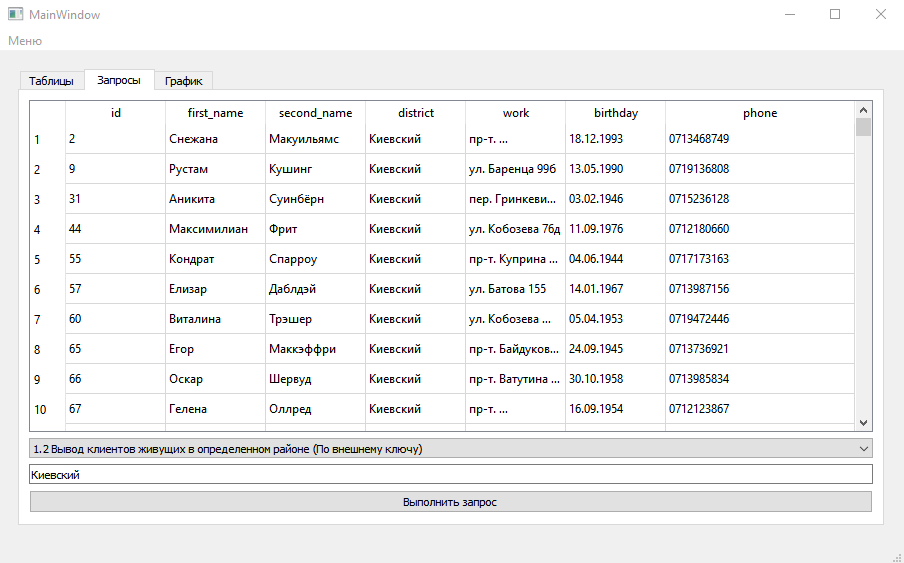


Рисунок 4.26 – Клиенты, живущие в киевском районе

Запрос, показывающий информацию об идентификаторе пользователя, имени, фамилии заказчика, названии района его проживания, месте его работы, дате его рождения, номере его телефона, у пользователей, которые родились позже указанной даты. Дату задаёт пользователь. Код предоставлен на рисунке 4.27, а результат – 4.28.

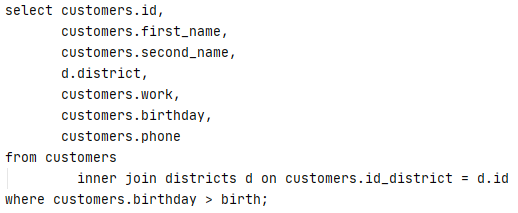


Рисунок 4.27– Симметричное внутреннее соединение с условием на дату

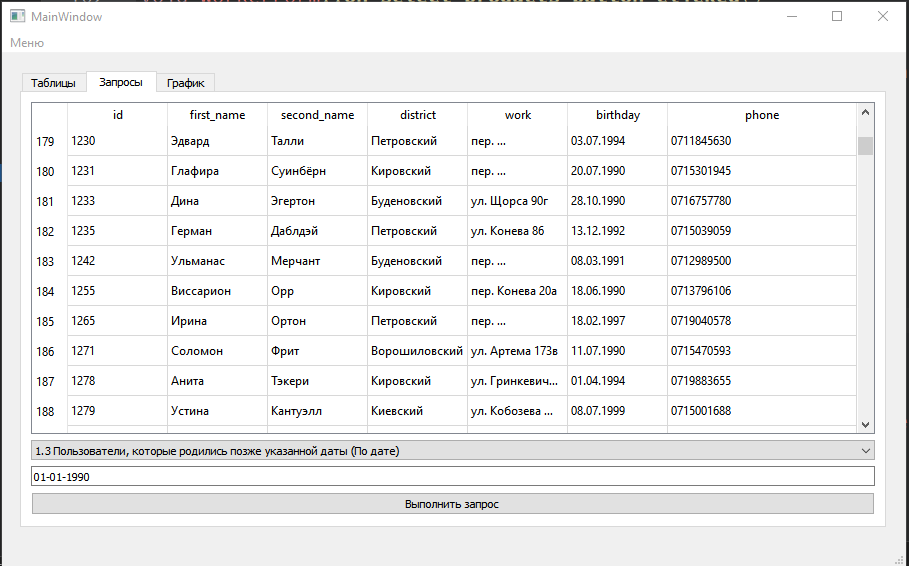


Рисунок 4.28 – Клиенты, родившиеся в 1990 году и позже

Запрос, показывающий информацию об идентификаторе продукта, названии продукта, дате према продукта на склад, количестве оставшихся единиц товара, цене за единицу товара, имя, фамилия заказчика, название магазина, типе продукта, у продуктов, которые были приняты на склад псле указанной даты. Дату задаёт пользователь. Код предоставлен на рисунке 4.29, а результат – 4.30.

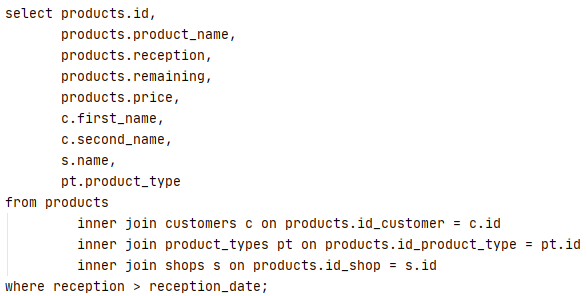


Рисунок 4.29– Симметричное внутреннее соединение с условиием на дату

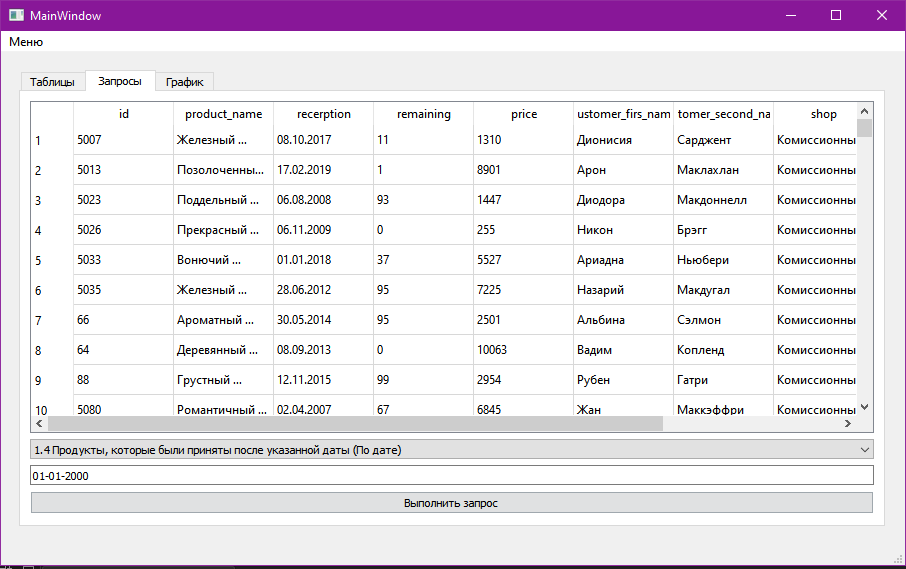


Рисунок 4.30 –Продукты, которые были приняты в 2000 году и позже

Рассмотрим примеры запросов аналогичного типа, но без условий. Запрос, показывающий информацию об имени, фамилии заказчика, месте его работы, номере квитанции сданного им товара, количестве единиц товара оставшихся на складе, названии магазина, в котором находится товар. Код предоставлен на рисунке 4.31, а результат – 4.32.

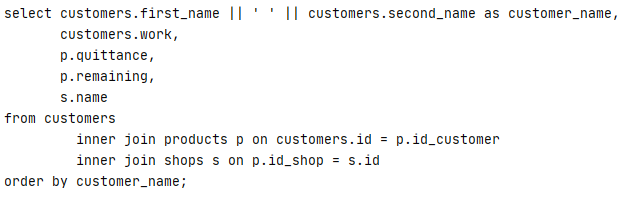


Рисунок 4.31 – Симметричное внутреннее соединение без условия

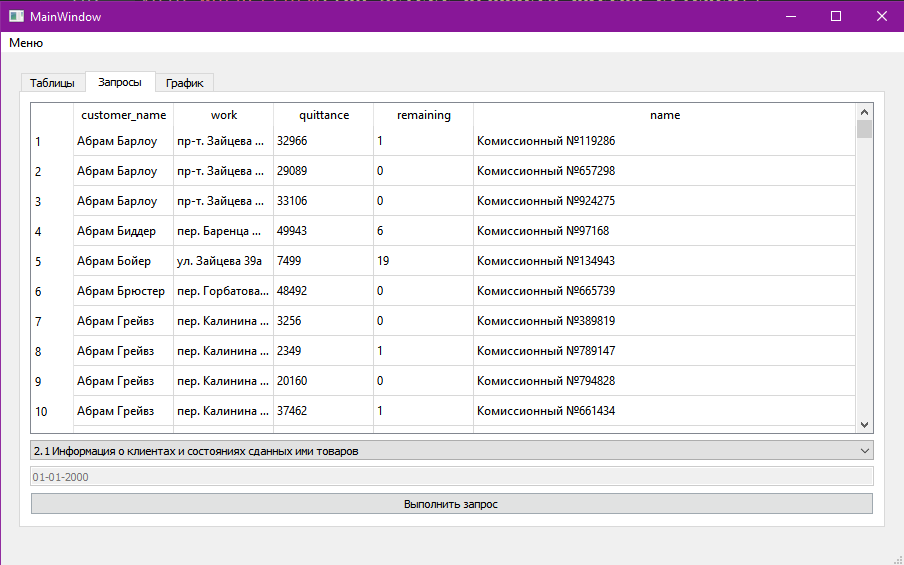


Рисунок 4.32 – Информация о клиентах и состоянии сданных ими товаров

Запрос, показывающий информацию о номере квитанции, с которой товар был принят, имени продукта, дате приема, количестве проданных экземпляров, дате каждой покупки, имени магазина, в котором находится. Код предоставлен на рисунке 4.33, а результат – 4.34.

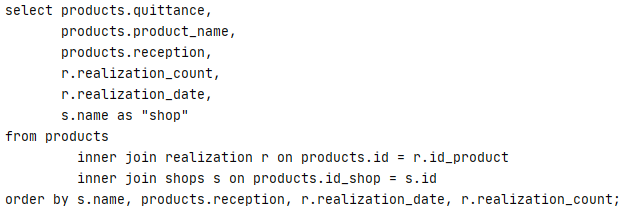


Рисунок 4.33 – Симметричное внутреннее соединение без условия

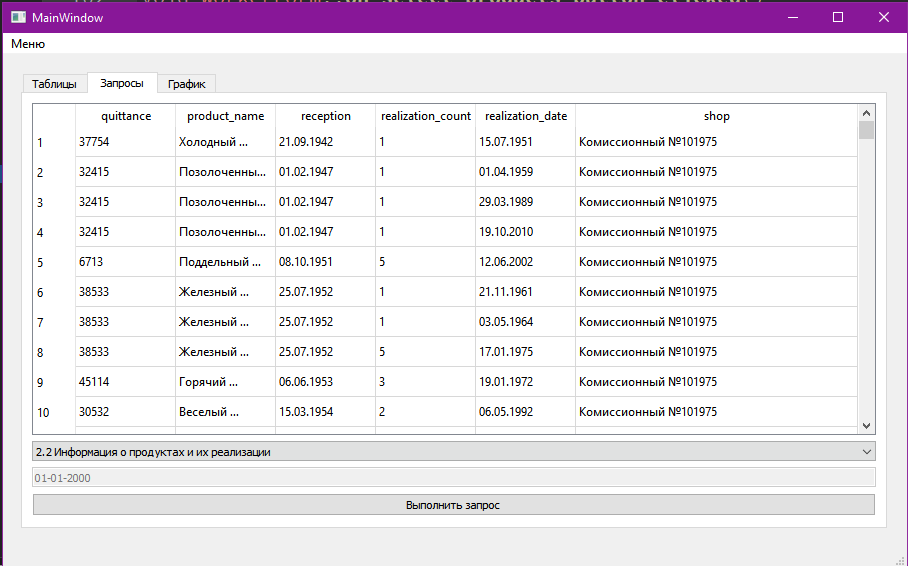


Рисунок 4.34 – Информация о продуктах и их реализации

Запрос, показывающий информацию о номере чека, выданного при продаже, дате продажи, количества проданного товара, логине работника, который принял товар, названии магазина, в котором находится товар. Код предоставлен на рисунке 4.35, а результат – 4.36.

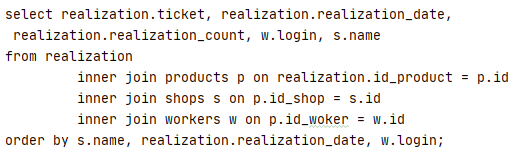


Рисунок 4.35 – Симметричное внутреннее соединение без условия

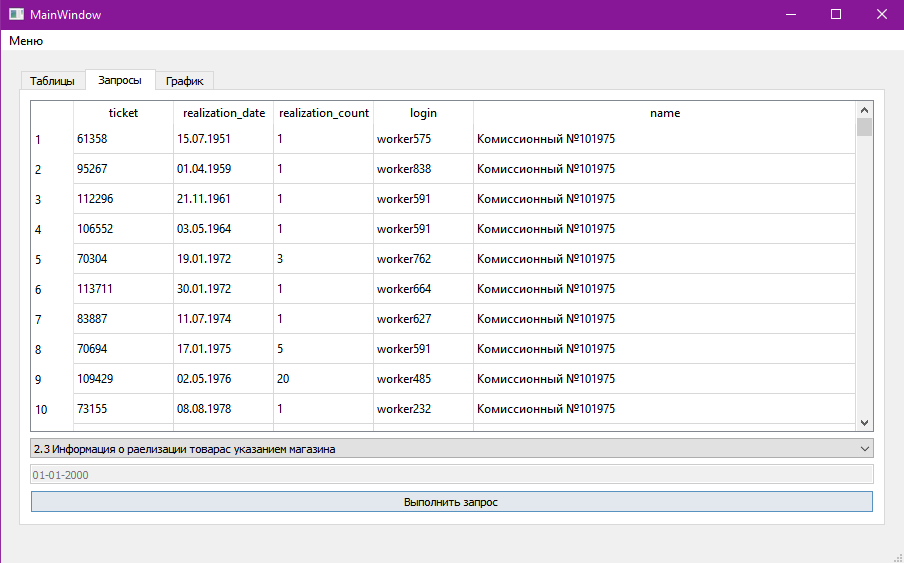


Рисунок 4.36 – Информация о реализации товаров с указанием магазина

Запрос с оператором LEFT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из первой по порядку (левой) таблицы, даже если они не соответствуют условию. У записей левой таблицы, которые не соответствуют условию, значение столбца из правой таблицы будет NULL (неопределённым).

Запрос, показывающий информацию о номере квитанции товара, с которой он был принят на склад, количестве товара, которое было при преме, оставшееся количество товара, номер чека, который был выписан при продаже товара. Код запроса представлен на рисунке 4.37, а результат – 4.38.

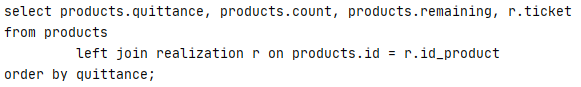


Рисунок 4.37 – Левое внешнее соединение

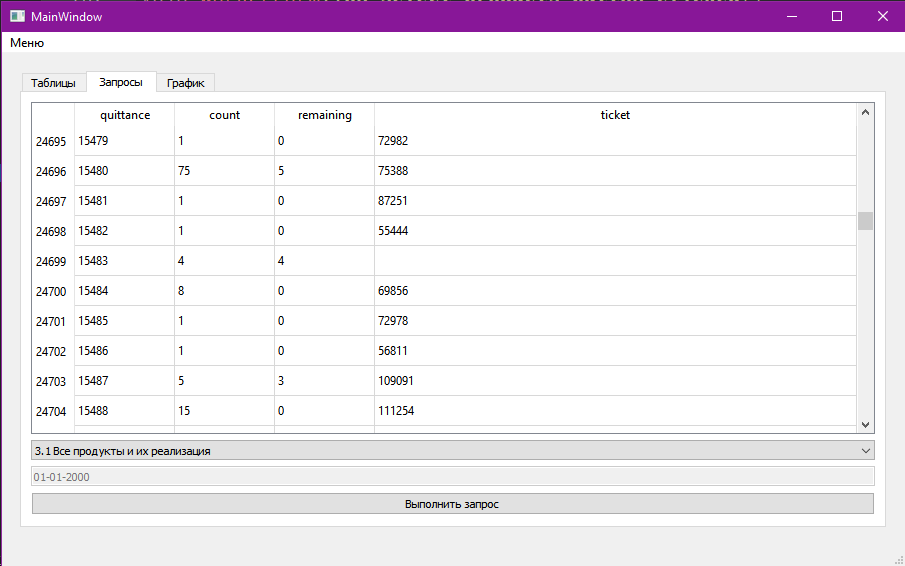


Рисунок 4.38 – Все продукты и их реализация (даже те, что еще не были реализованы)

Запрос с оператором RIGHT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из второй по порядку (правой) таблицы, даже если они не соответствуют условию. У записей правой таблицы, которые не соответствуют условию, значение столбца из левой таблицы будет NULL (неопределённым).

Запрос, показывающий информацию о номере чека, который был продан товар, дате продажи, количестве проданного товара, логине работника, который продал товар. Код запроса представлен на рисунке 4.39, а результат – 4.40.

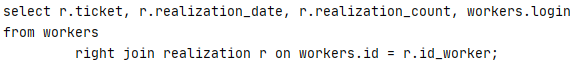


Рисунок 4.39 – правое внешнее соединение

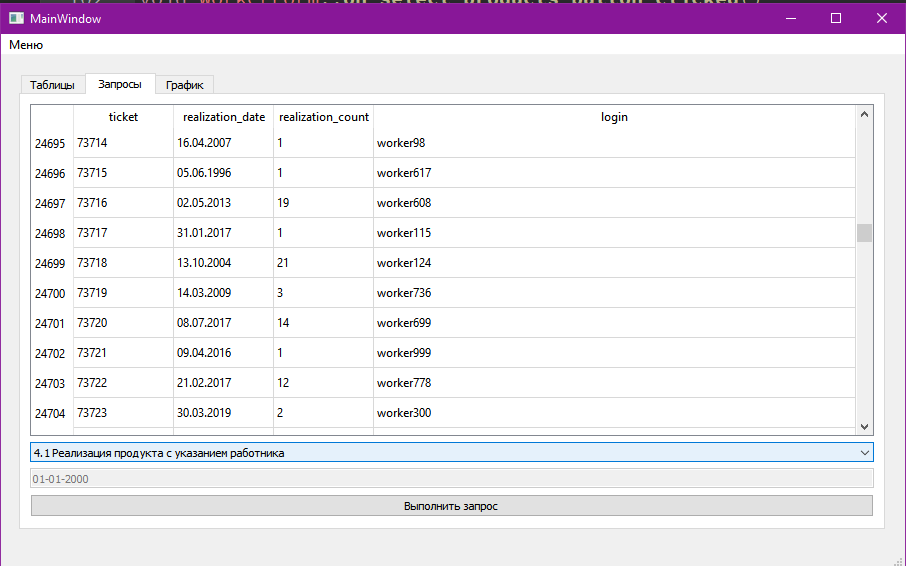


Рисунок 4.40– Реализация продукта с указанием работника

Рассмотрим запрос на запросе по принципу левого. С помощью него мы находим информацию о квитанции, которая была выписана при приеме товара, имени продукта, дате приема продукта, логине работника, который принял товар на склад. Код предоставлен на рисунке 4.41, а результат – 4.42.

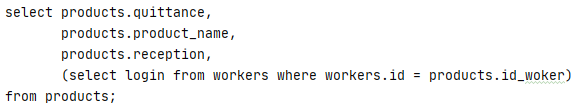


Рисунок 4.41 – Запрос на запросе по принципу левого соединения

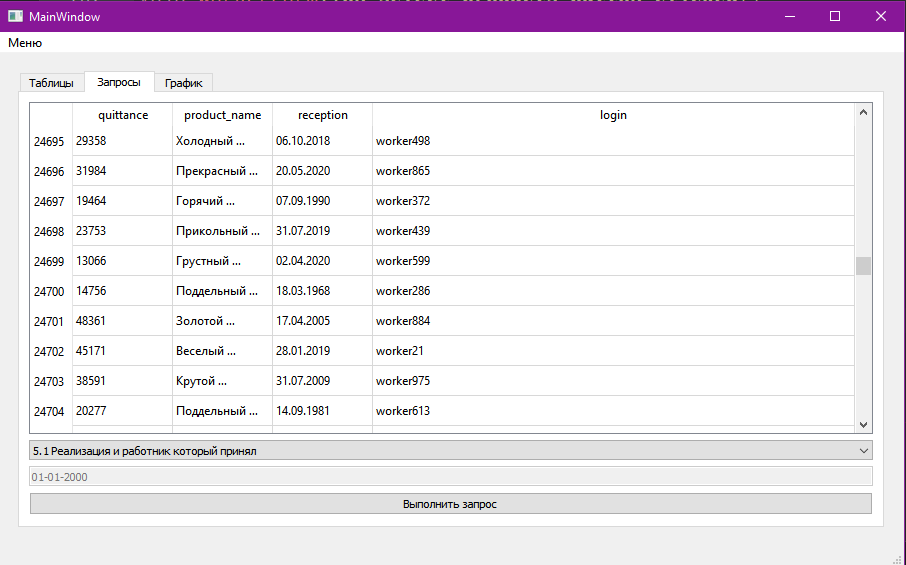


Рисунок 4.42 – Реализация и работник, который принял товар

С помощью итогового запроса без условия пользователь может получить информацию о квитанциях товаров, количестве чеков, выписанных на этот товар, дате приема товара на склад, имени магазина, в котором товар был принят (см. рис. 4.43 – 4.44).

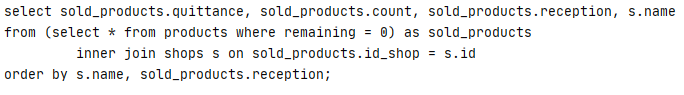


Рисунок 4.43 – Итоговый запрос без условия

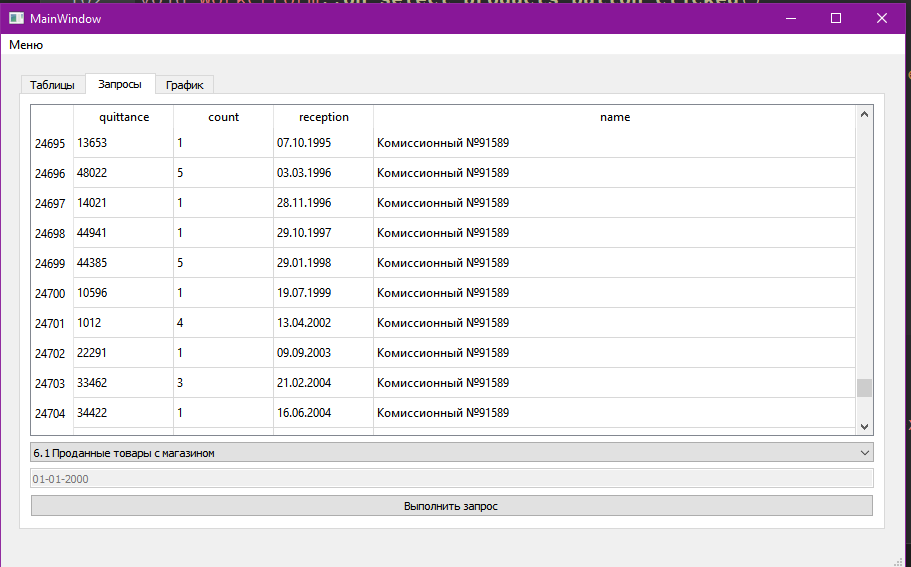


Рисунок 4.44 – Проданные товары с указанием магазина

Итоговый запрос без условия с итоговыми данными вида «всего», «в том числе» используются для того, чтобы получить информацию об общем количестве принятого товара и количество полностью проданного товара (больше не осталось на складе) (см. рис. 4.45 – 4.46).



Рисунок 4.45 - Итоговый запрос без условия с итоговыми данными выда: "всего", "в том числе"

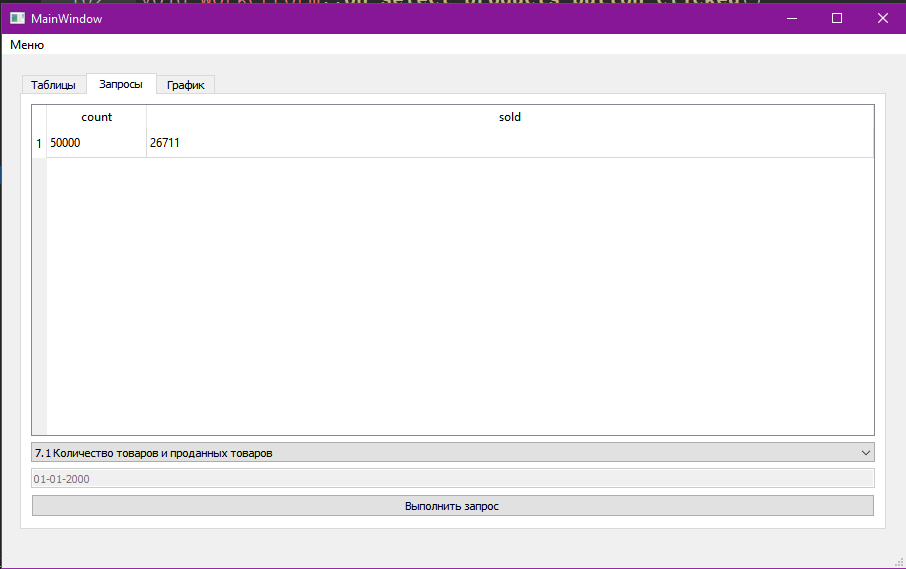


Рисунок 4.46 – Количество товаров и проданных товаров

Итоговый запрос с условием на данные используем, чтобы получить информацию об имени товара, квитанции, с которой был принят товар, оставшееся количество товара, количестве чеков, оформленных на товар, дате приема, имени магазина, в котором находится товар у товаров, количество которых на складе больше указанного. Количество задаёт пользователь. Код представлен на рисунке 4.47, результат – 4.48.

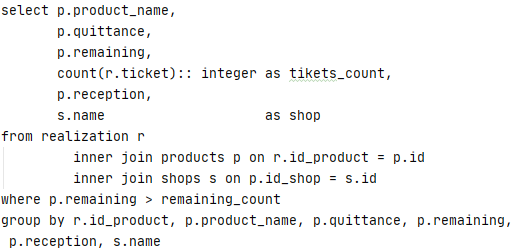


Рисунок 4.47 - Итоговые запросы с условием на данные по значению

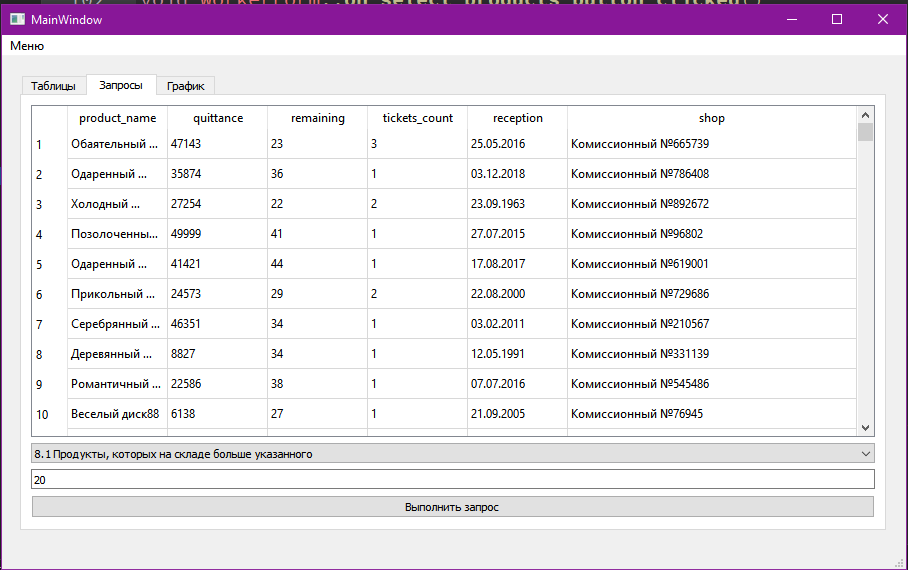


Рисунок 4.48 – Товары, которых на складе больше 20

Итоговый запрос с условием на данные, но в качестве проверки используется маска. Такой запрос используется дляполучения информации об имени и фамилии клиента, его телефоне, количестве товаров, сданных им, у клиентов телефонный номер которых содержит определенный набор цифр. Набор цифр задает пользователь. (см. рис. 4.49 – 4.50).

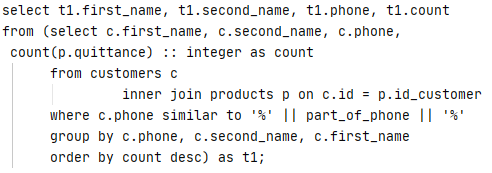


Рисунок 4.49- Итоговый запрос с условием на данные по маске

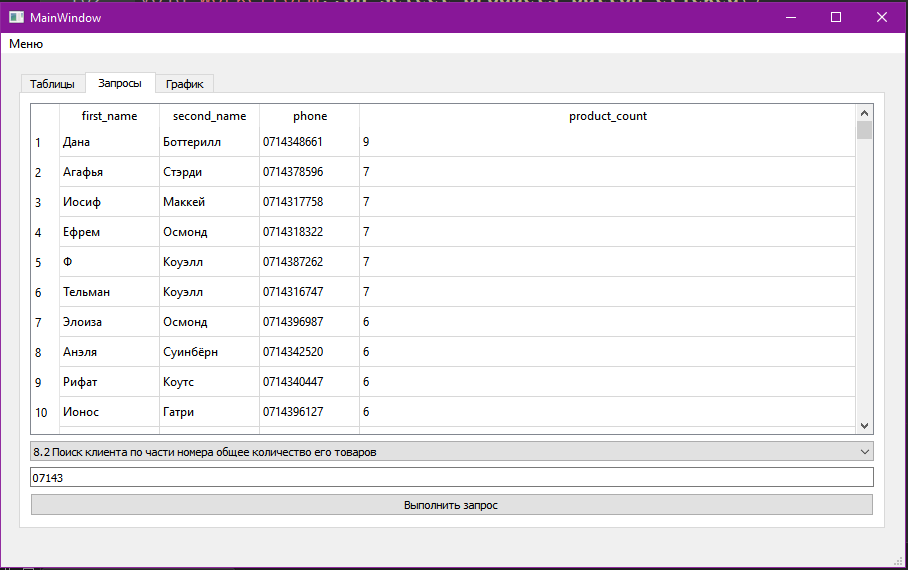


Рисунок 4.50– Клиент, номер которого содержит «07143» и общее количество его товаров

Итоговый запрос с условием на данные с использованием индекса используется для получения информации об имени продукта, номере квитанции, с которой был принят товар, имени магазина, в котором находится товар, количестве чеков, которые были выписаны на этот товар, у товара с указанным индексом. Индекс товара задает пользователь. (рис. 4.51 – 4.53)



Рисунок 4.51– Создание индекса для таблицы реализации

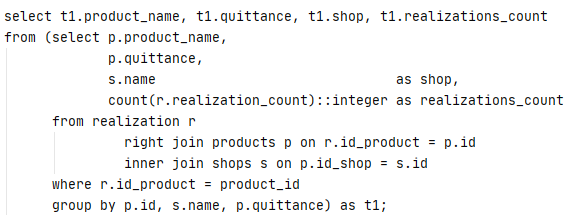


Рисунок 4.52– Итоговый запрос с условием на данные с использованием индекса

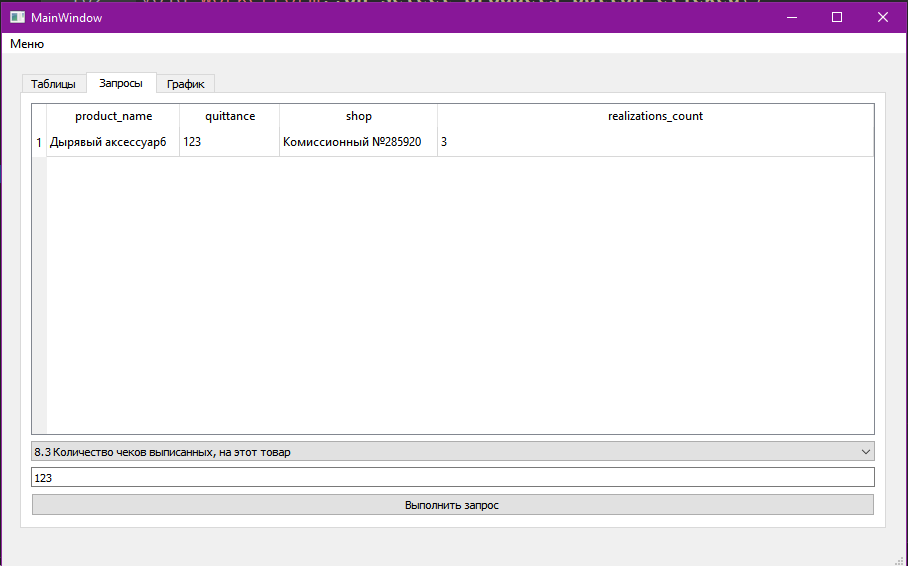


Рисунок 4.53 – Количество чеков, выписанных на товар, имеющий номер квитанции 123

Итоговый запрос с условием на данные используется для получения информации об имени магазина, количестве работников в магазине, у магазина с указанным индексом. Индекс магазина указывается пользователем. (рис. 4.54 – 4.55)

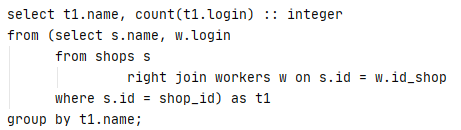


Рисунок 4.54- Итоговый запрос с условием на данные

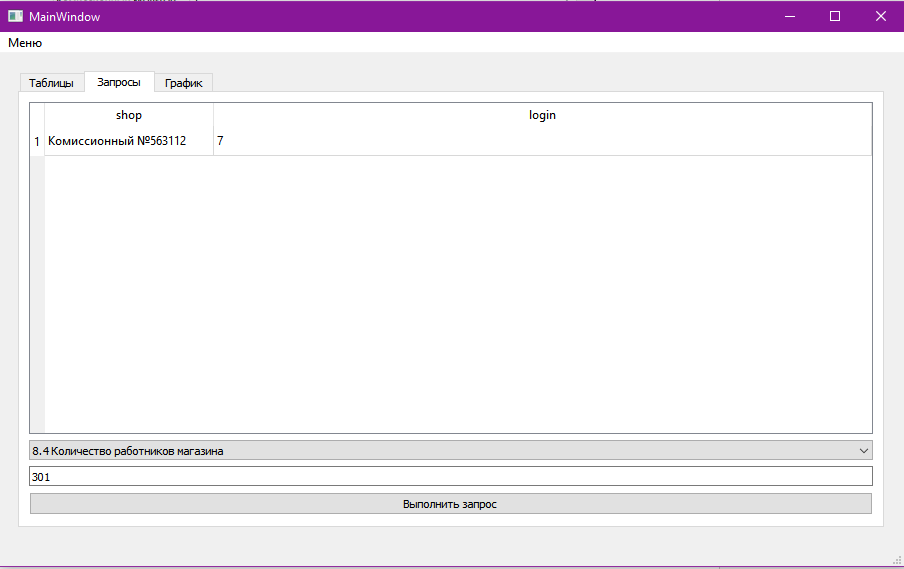


Рисунок 4.55 – Количество работников магазина с идентификационным номером 301

Итоговый запрос с условием на группы показывает пользователю информацию об имени продукта, названии магазина, в котором находится продукт, общем количестве реализаций продукта, для продуктов, общее количество реализаций которых больще укащанного. Количество реализаций указывает пользователь. Код представлен на рисунке 4.56, а результат – 4.57.

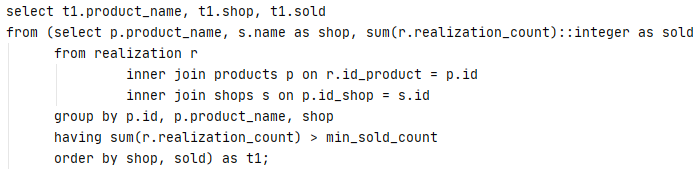


Рисунок 4.56– Итоговый запрос с условием на группы

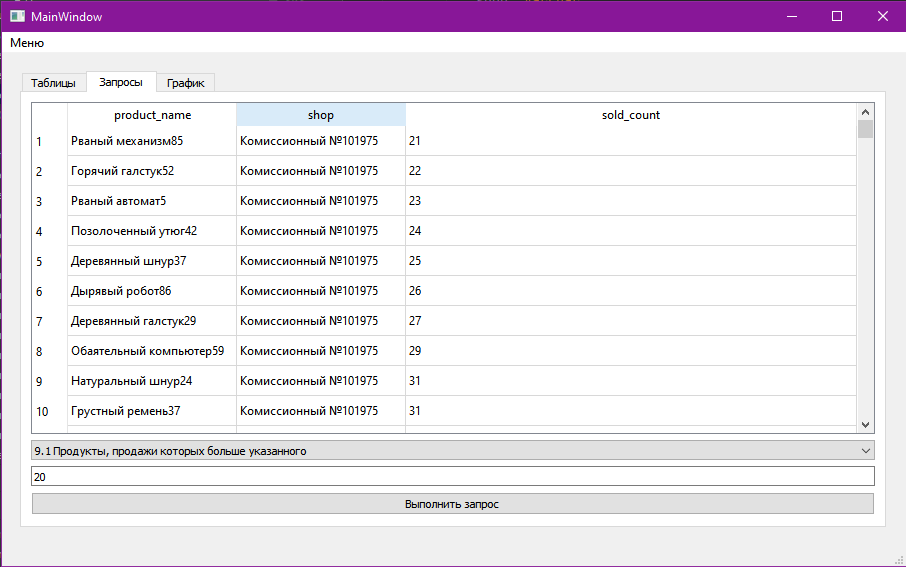


Рисунок 4.57 – Продукты, продажи которых больше 20

Итоговый запрос с условием на данные и группы используем для вывода информации об имени и фамилии пользователя, имени магазина, социальном статусе, количестве продуктов, которые пользователь сдал в магазин, для пользователей с указанным социальным статусом и количеством сданных товаров больше указанного (см. рис. 4.58 – 4.59). Социальный статус и количество на вход подаёт пользователь.

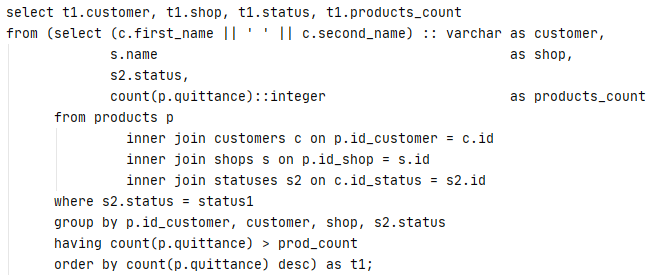


Рисунок 4.58 – Итоговый запрос с условием на данные и на группы

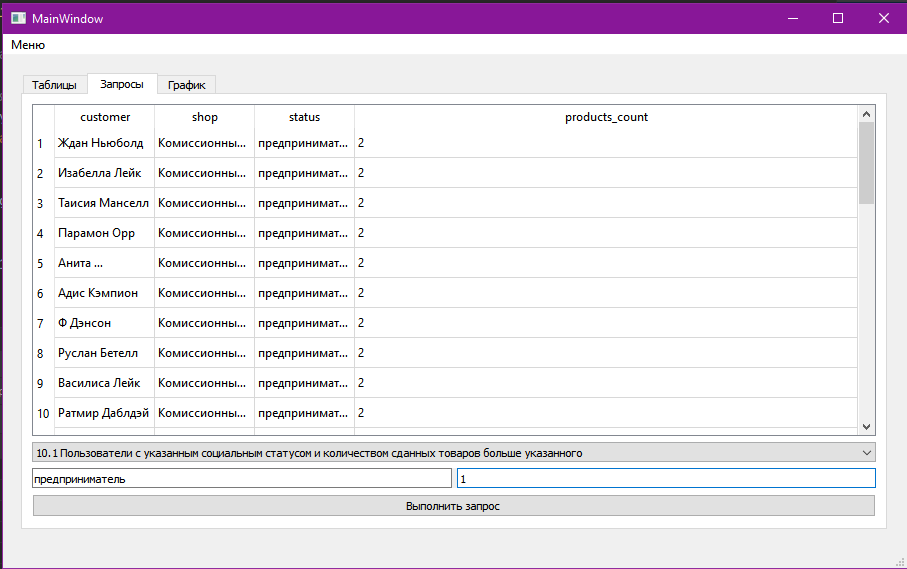


Рисунок 4.59 – Предприниматели, у которых сданных товаров больше 1

Запрос на запросе по принципу итогового запроса в данной системе создаётся для демонстрации средней цены товара в магазине и мени магазина. (см. рис. 4.60 – 4.61).

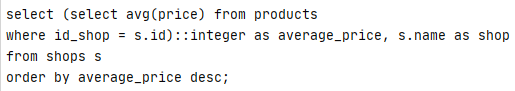


Рисунок 4.60– Запрос на запросе по принципу итогового запроса

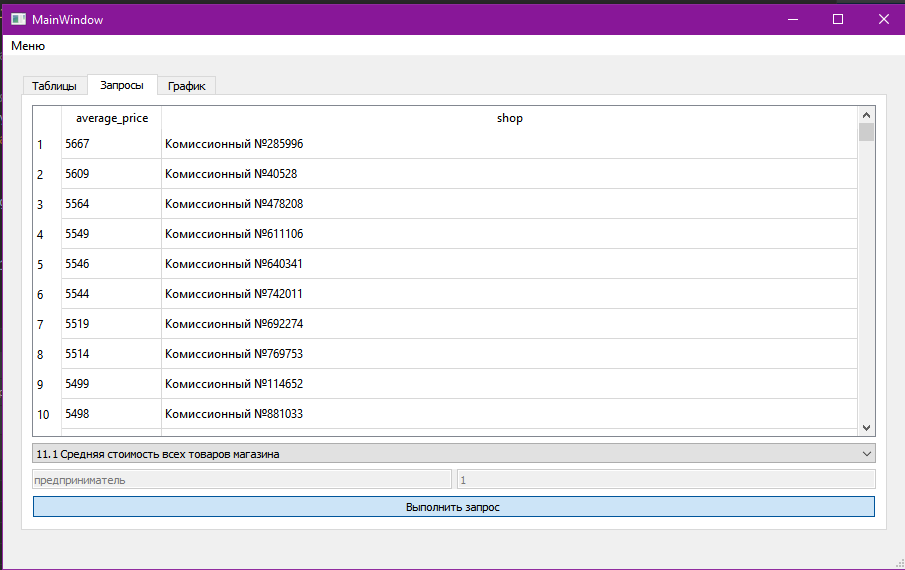


Рисунок 4.61 – Средняя стоимость всех товаров магазинов

Оператор языка UNION предназначен для объединения результирующих таблиц базы данных, полученных с применением слова SELECT. Условие объединения результирующих таблиц: совпадение числа, порядка следования и типа данных столбцов.

Используем данный тип запроса для того, чтобы получить полную информацию о продуктах, которые проданы полностью и продуктах, которые не проданы ни разу. Код представлен на рисунке 4.62, результат работы – 4.63.

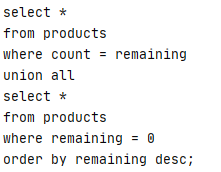


Рисунок 4.62 – Запрос с использованием объединения

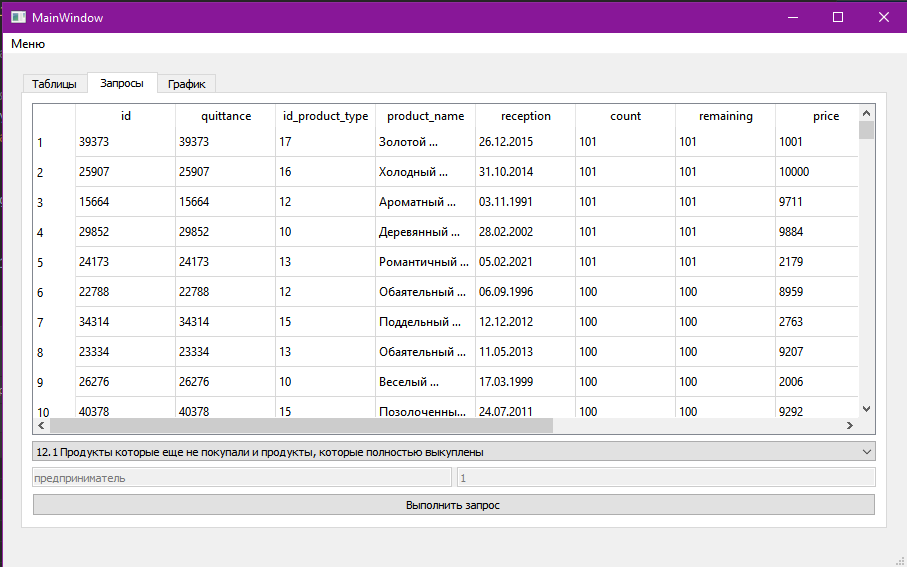


Рисунок 4.63 – Товары, которые полностью скупили и товары, которые еще не покупали

Запрос с подзапросом с использованием in. Запрос позволяет получить информацию об имени и фамилии клиента, номере телефона, для клиентов, товары которых еще остались на складе. (рис. 4.64 – 4.65)

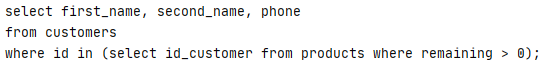


Рисунок 4.64 – Запрос с подзапросом с использованием in

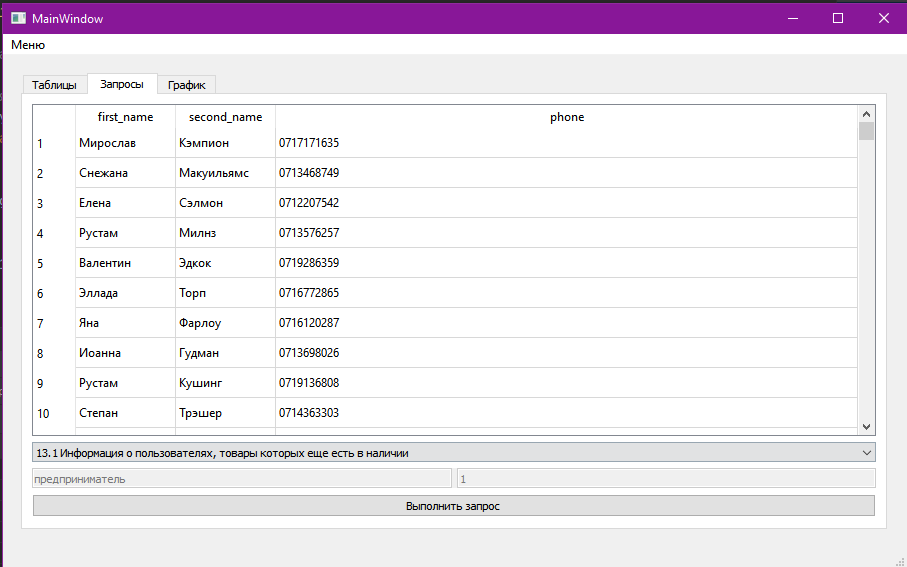
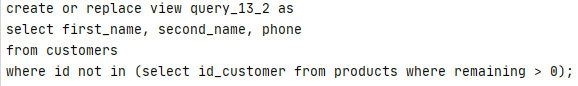


Рисунок 4.65 – Пользователи, товары которых еще есть в наличии

Запрос с подзапросом с использованием in. Запрос позволяет получить информацию об имени и фамилии клиента, его номере телефона, для клиентов, товаров которых уже не осталось на складе. (рис. 4.66 – 4.67)



Риснуок 4.66 - Запрос с подзапросом с использованием not in

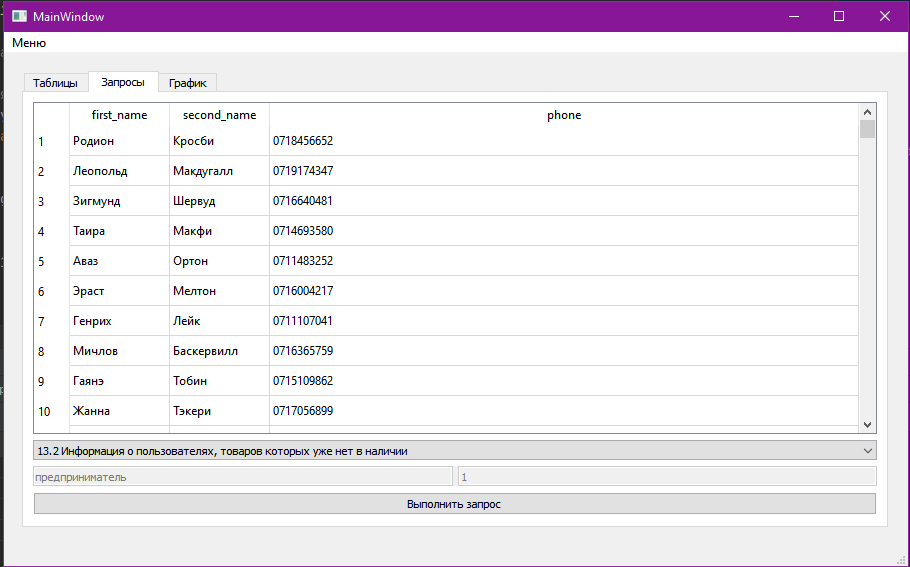


Рисунок 4.67 – Пользователи, товаров которых уже нет в наличии

Запрос с подзапросом с использованием case. Запрос позволяет получить информацию о номере квитанции, с которой был принят товар, имени товара, цене товара, оставшемся количество товара на складе, количестве товара при поступлении, популярности продукта у покупателей. Популярность определяется по отношению оставшегося количества товара к исходному количеству товара (0% - раскупили,1-20% - очень хорошо покупается, 21-40% - хорошо покупается, 41-60% - покупается, 61-80% - плохо покупается, 80-99% - очень плохо покупается, 100% - не покупается). (рис. 4.68 – 4.69)

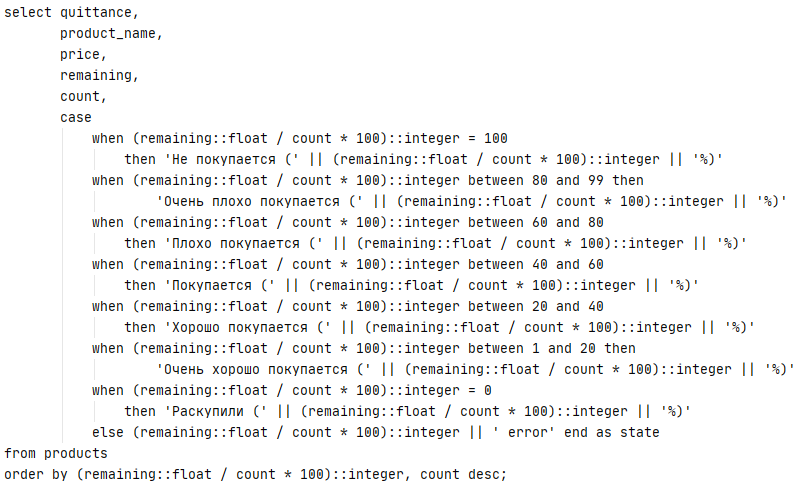


Рисунок 4.68 - Запрос с подзапросом с использованием case

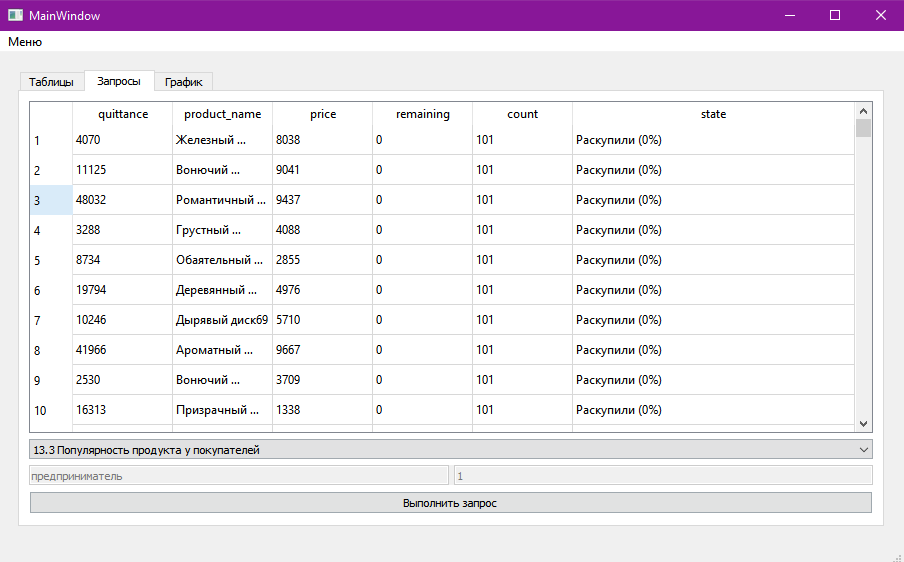


Рисунок 4.69 – Популярность продуктов у покупателей

Запрос с подзапросом с использованием операций над итоговыми данными. Запрос позволяет получить информацию о номере квитанции, с которой был принят товар, имени товара, цене товара, имени магазина, в котором находится . (рис. 4.70 – 4.71)

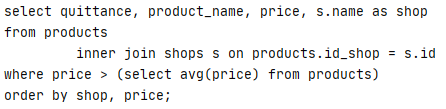


Рисунок 4.70 - Запрос с подзапросом с использованием операций над итоговыми данными

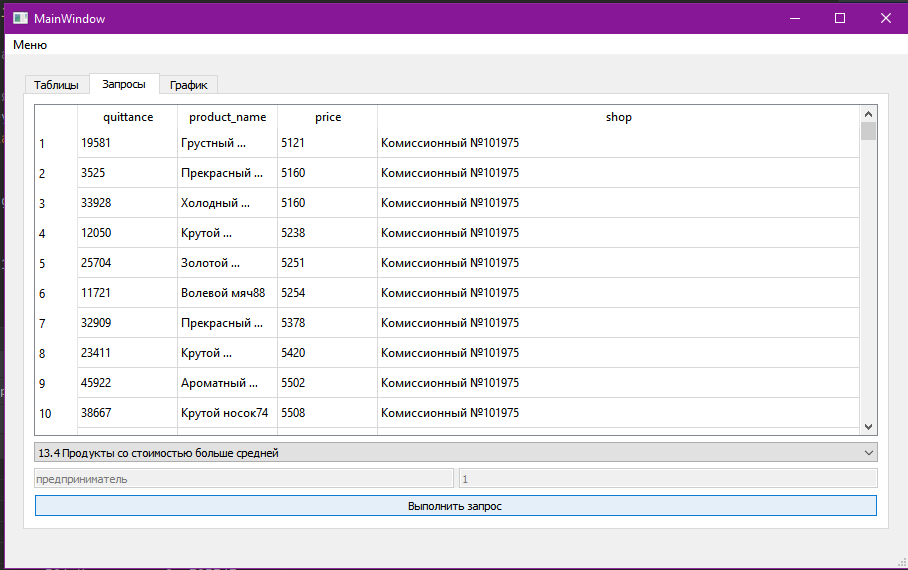


Рисунок 4.71 – Продукты со стоимостью больше средней

4.5 Создание представлений и хранимых функций

Представления – это таблицы чье содержание выбирается или получается из других таблиц, при этом они не содержат никаких собственных данных.

Представление – это фактически запрос, который выполняется всякий раз, когда представление становится темой команды. Вывод запроса при этом в каждый момент становится содержанием представления [6].

CREATE VIEW создаёт представление запроса.

Все запросы, которые выполняются без параметров, реализованы с помощью представлений. Рассмотрим пример создания представления (см. рис. 4.72).

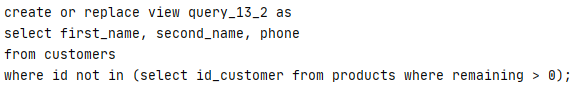


Рисунок 4.72 – Представление для запроса с левым соединением

Остальные запросы реализованы аналогичным способом.

Хранимая процедура – это объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Postgres позволяет вызывать функции с именованными параметрами. Команда CREATE FUNCTION определяет новую функцию.

Рассмотрим пример создания функции для запроса (см. рис. 4.73).

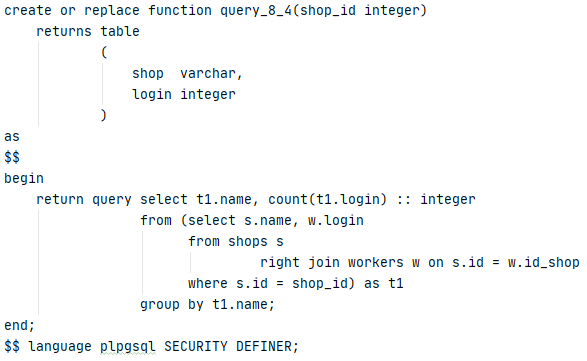


Рисунок 4.73 – Пример создания функции для итогового запроса с условием на данные и группы

В данном случае отличие от представлений в том, что мы можем передать параметры; указывает тип возвращаемых данных; также нужно указывать язык, на котором, пишется функция.

Остальные параметризованные запросы аналогичным способом помещаются в такую «оболочку».

# 5 Разработка клиентского приложения

При разработке клиентского приложения была необходимость сделать разные интерфейсы для разных видов пользователей. Это было достигнуто благодаря определению роли пользователя после авторизации, после чего производится выбор интерфейса в зависимости от определенной при авторизации роли.

Первой формой всегда является форма авторизации (см. рис. 5.1).

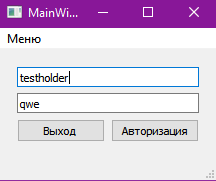


Рисунок 5.1 – Форма авторизации

В этой форме пользователь может авторизоваться либо выйти. Меню содержит один пункт – выйти из профиля, и сохраняется для любой формы, но в форме авторизации при нажатии на нее ничего не произойдет.

Для определения роли входящего используется следующий механизм: сначала форма авторизации испускает сигнал, который запрашивает проверку роли в базе данных по введенному логину и паролю (рис. ).



Рисунок 5.2 – Запрос на проверку роли по логину и паролю

Главная форма обрабатывает этот сагнал и выолняет проверку роли по присланному логину и паролю (рис. 5.3). В зависимости от результата выполняется либо подключение к базе данных (рис. 5.4) и переход на нужную форму, либо, если введенный логин или пароль не верен, то пользователю выводится предупреждение об ошибке аутентификации.

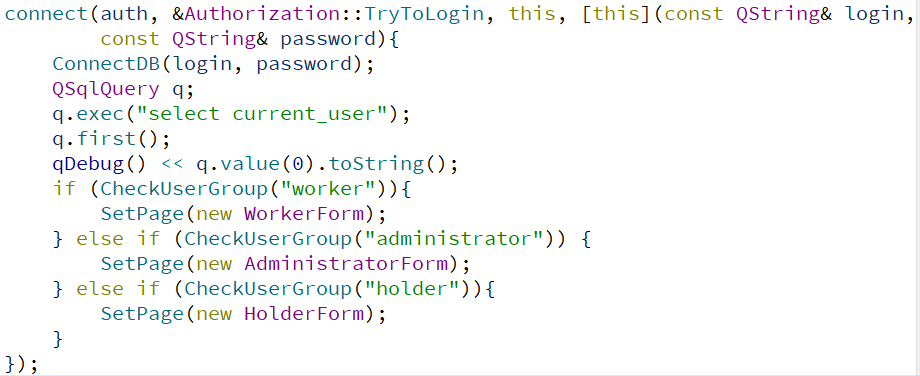


Рисунок 5.3 – Проверка роли пользователя

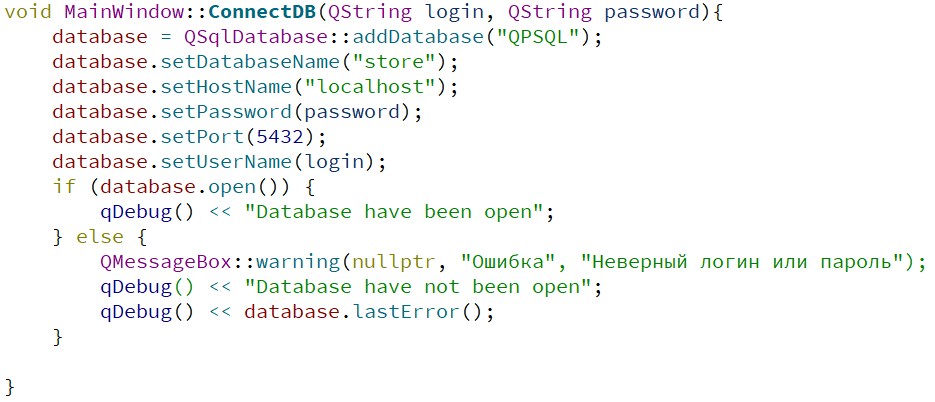


Рисунок 5.4– Подключение к базе данных

5.1 Работник(worker)

Для работников был разработан интерфейс, который разделен на 3 части: «Продукты» (рис. 5.5), «Реализация» (рис. 5.6) и «Клиенты» (рис. 5.7). Подробнее про работу с формой работника можно прочитать в руководстве работника (приложение Д).

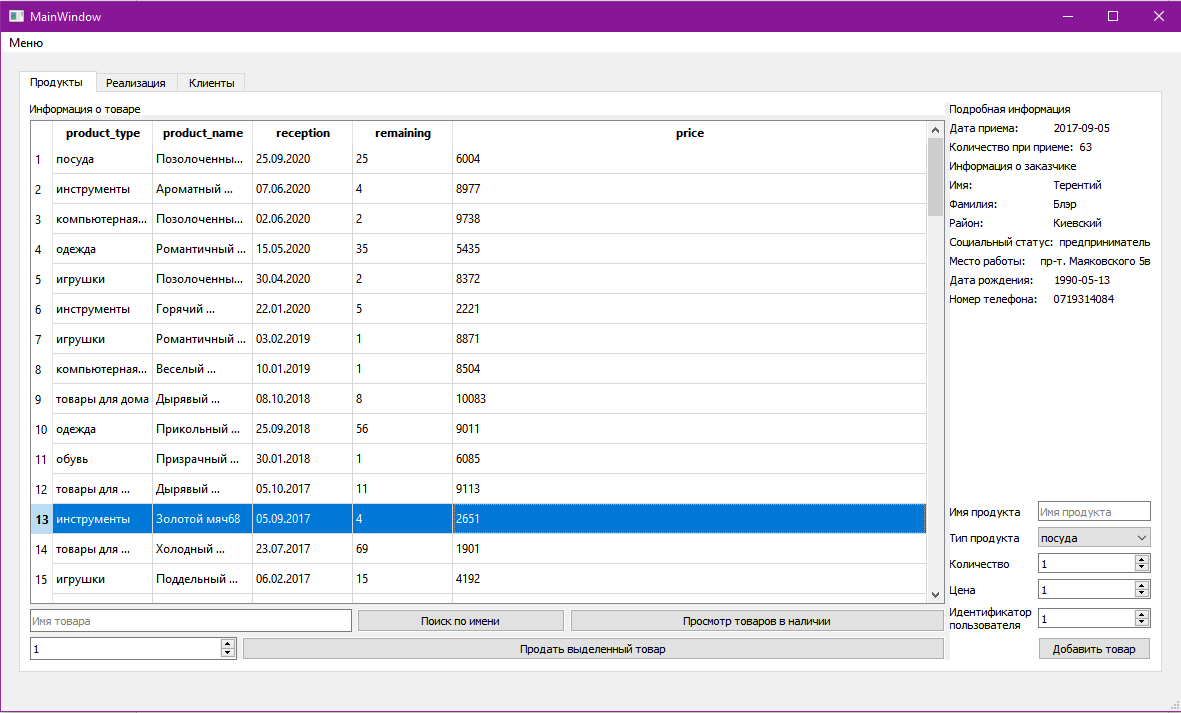


Рисунок 5.5 – Форма работника для взаимодействия с продуктами

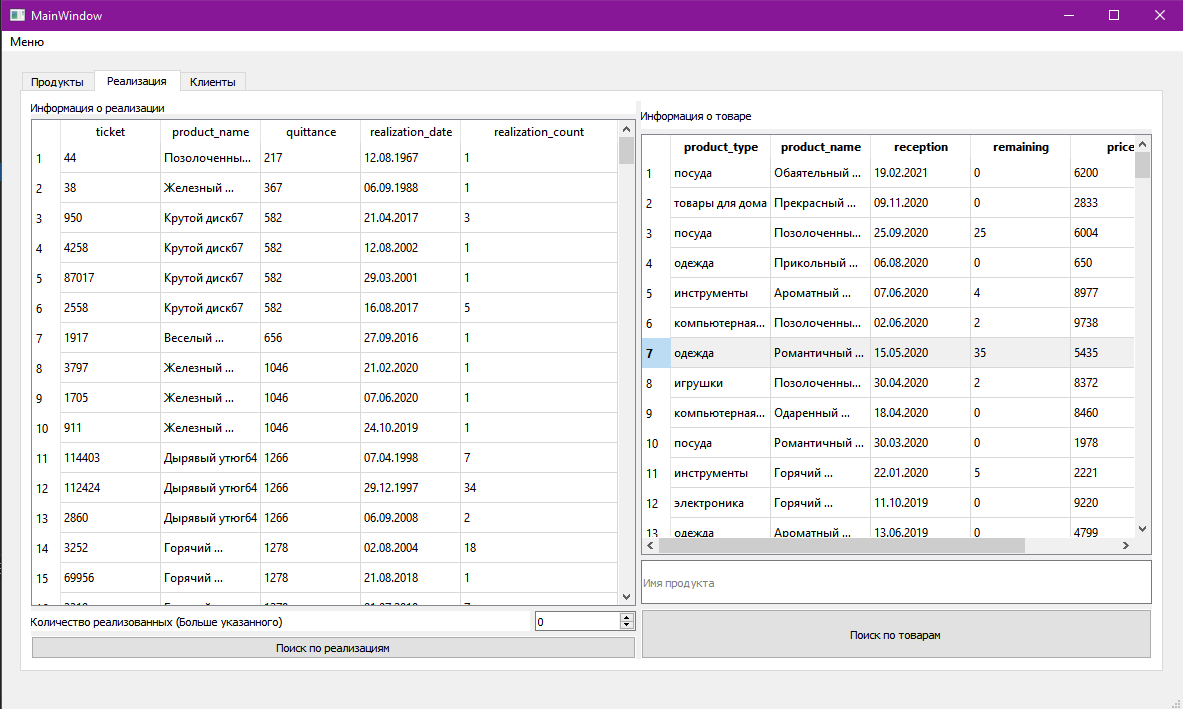


Рисунок 5.6 – Форма работника для взаимодействия между реализациями и товарами

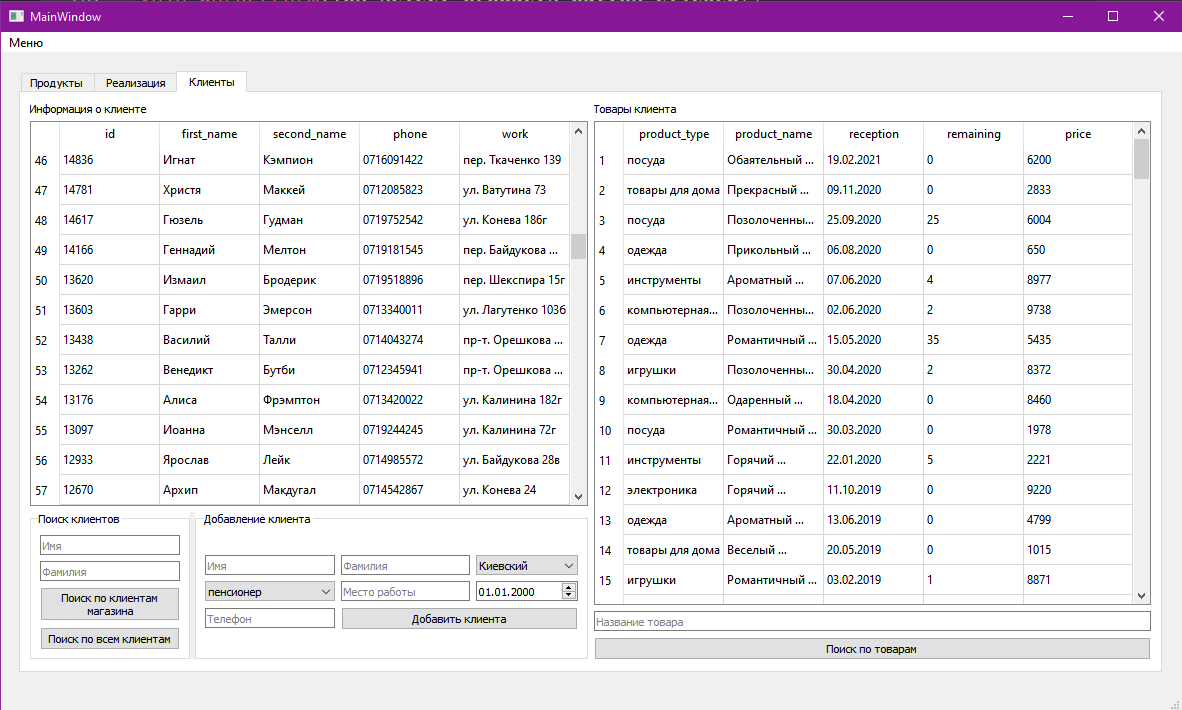


Рисунок 5.7 – Форма работника для взаимодействия с клиентами и их товарами

5.2 Администратор (administrator)

Для администратора был разработан интерфейс, который разделен на 2 части: «Роли» (рис. 5.8) и «Справочники» (рис. 5.9). Подробнее про работу с формой администратора можно прочитать в руководстве администратора (приложение Е).

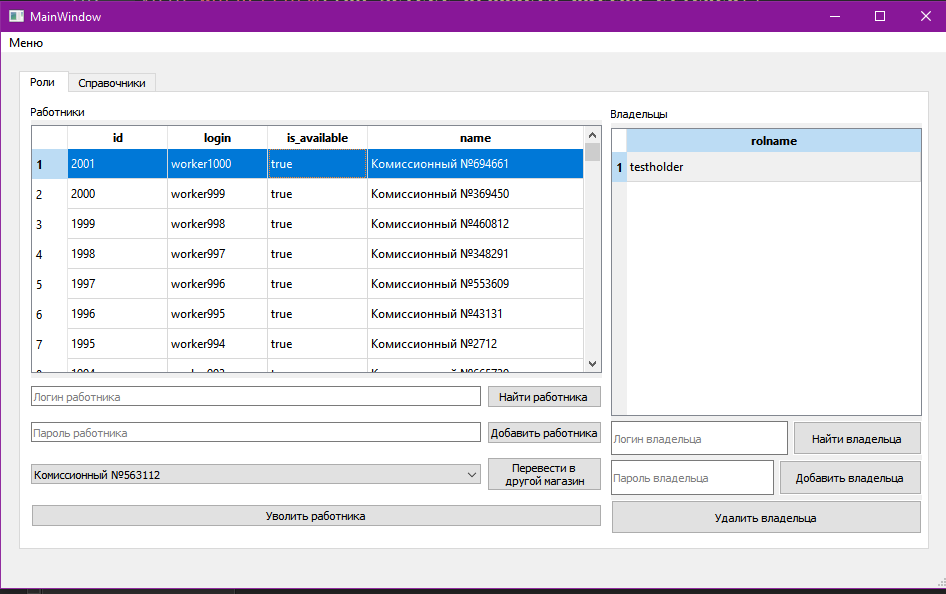


Рисунок 5.8 – Форма для работы с ролями

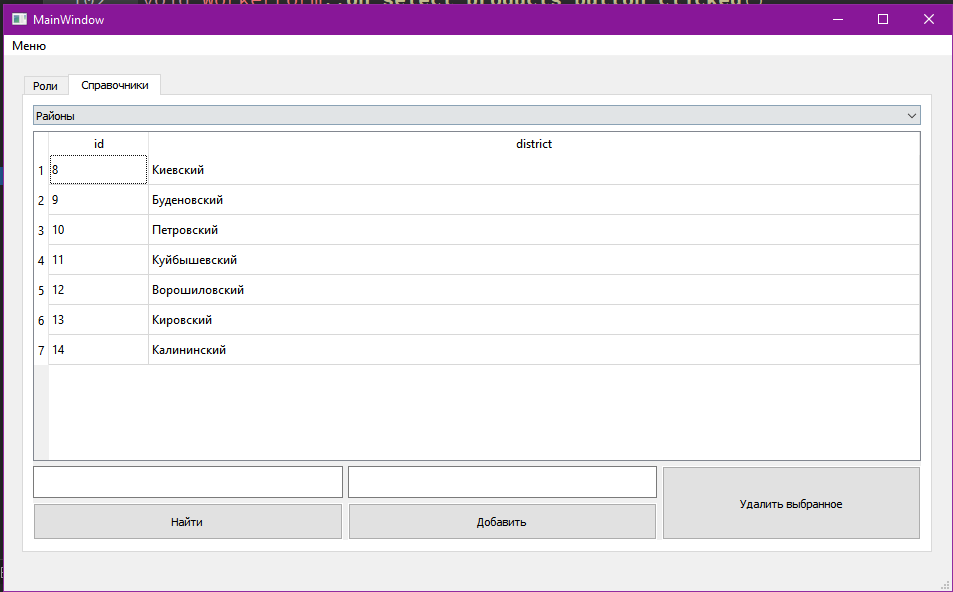


Рисунок 5.9 – Форма для работы со справочниками

5.3 Владелец (holder)

Для владельца был разработан интерфейс, который разделен на 3 части: «Таблицы» (рис 5.10 – 5.11), «Запросы» (рис. 5.12) и «График» (рис. 5.13 – 5.15). Подробнее про работу с формой владельца можно прочитать в руководстве владельца (приложение Ё).

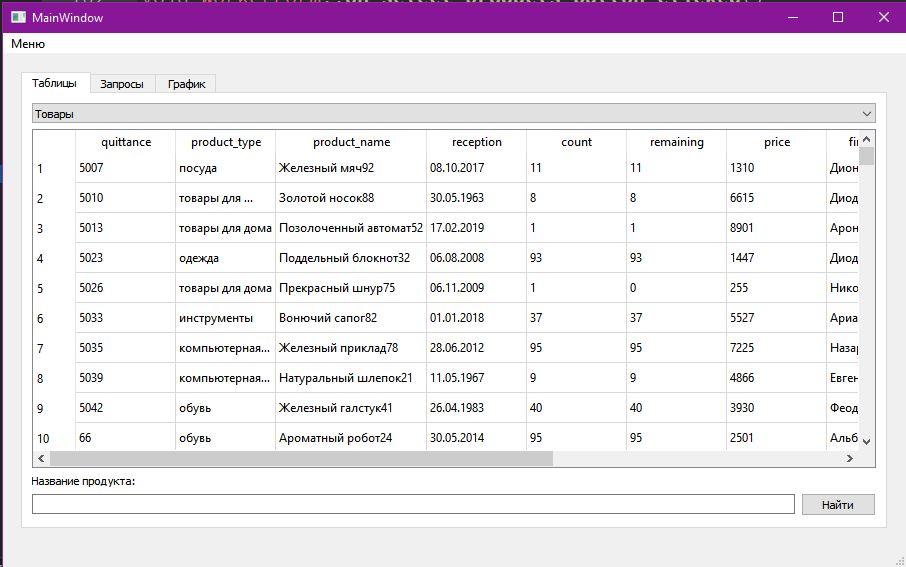


Рисунок 5.10 – Форма для работы с таблицами

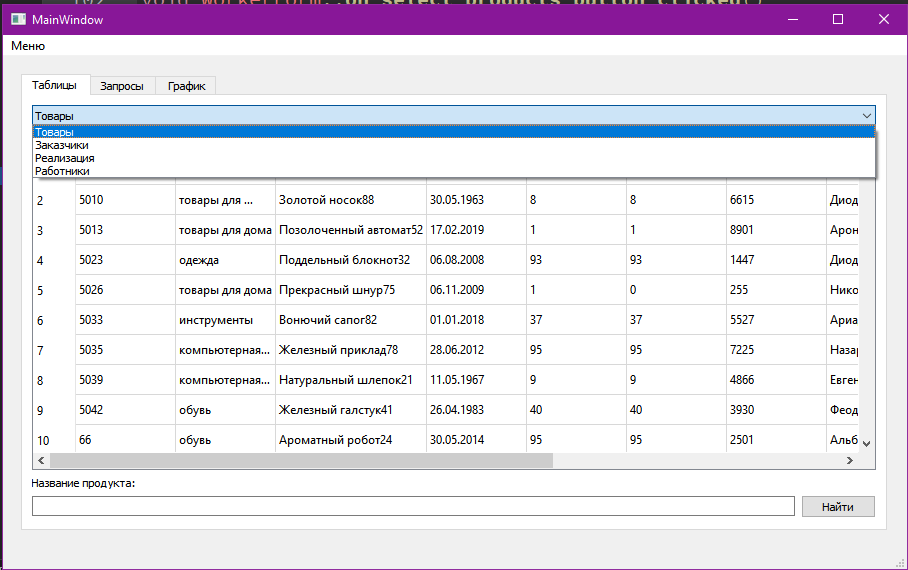


Рисунок 5.11– Выбор таблицы

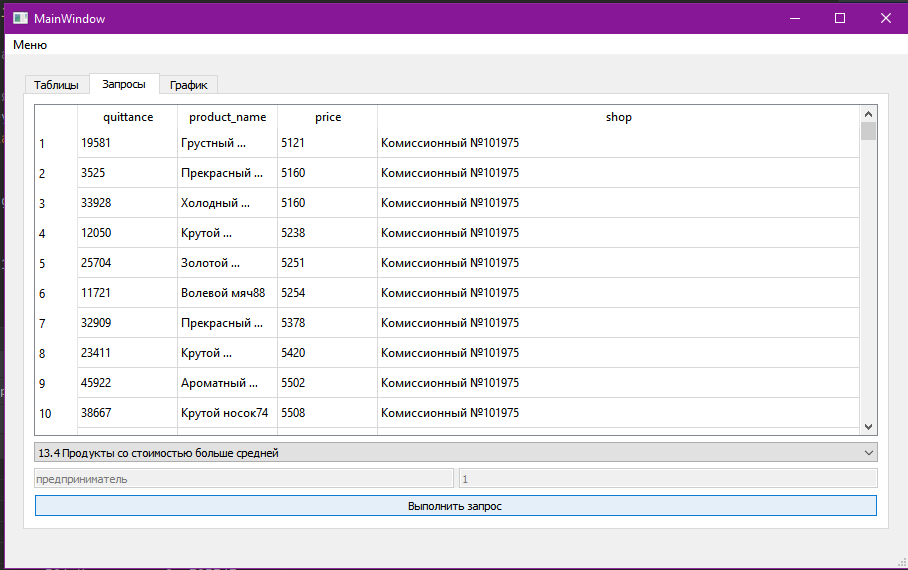


Рисунок 5.12 – Просмотр результатов запроса

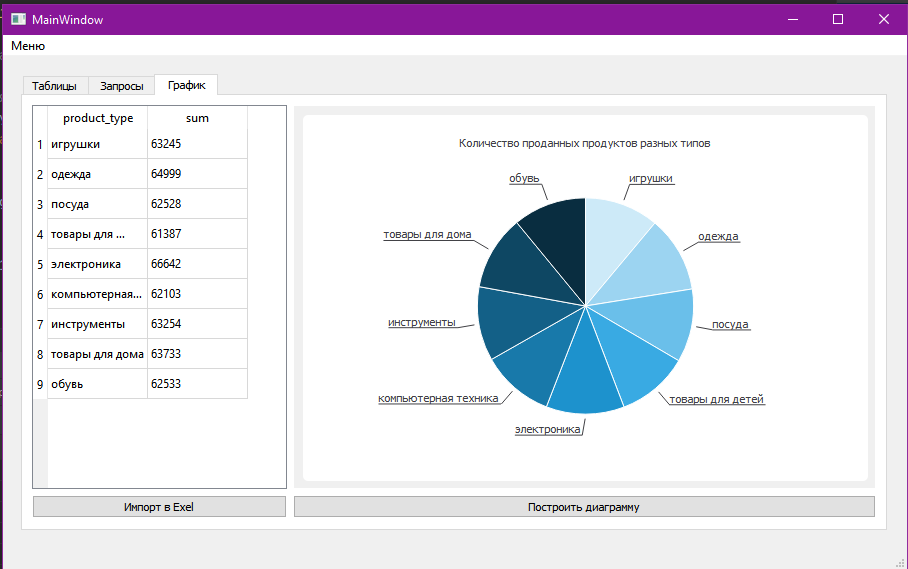


Рисунок 5.13 – Количество проданных продуктов разных типов

При нажатии на кнопку «Импорт в Exel» открывается форма выбора пути сохранения файла (рис. 5.10), после чего будет создан файл Exel содержащий записанную статистику (рис. 5.11).

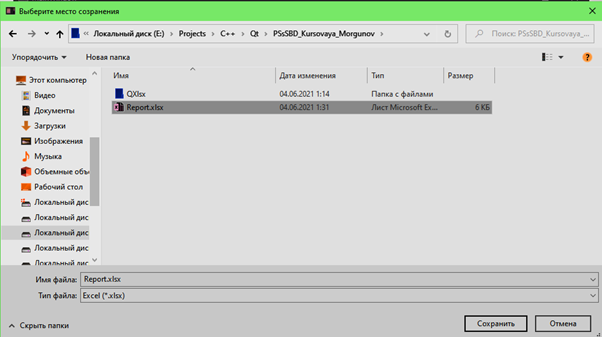


Рисунок 5.14 – Выбор места сохранения Excel

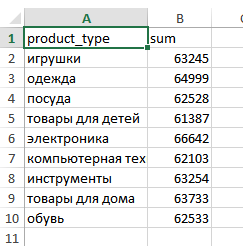


Рисунок 5.15– Результат экспорта в Excel

# Тестирование разработанной информационной системы (в т.ч. включая защиту от несанкционированного доступа, каскадное удаление)

6.1 Ошибки ввода

При разработке учитывались такие ошибки как: несанкционированный доступ (рис. 6.1).

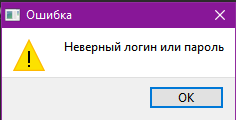


Рисунок 6.1 – Несанкционированный доступ

Клиентское приложение спроектировано так, чтобы предугадать ошибки пользователя и не дать ему их совершить. Например, пока пользователь не выберет товар, то кнопка «Продать товар» будет неактивна.

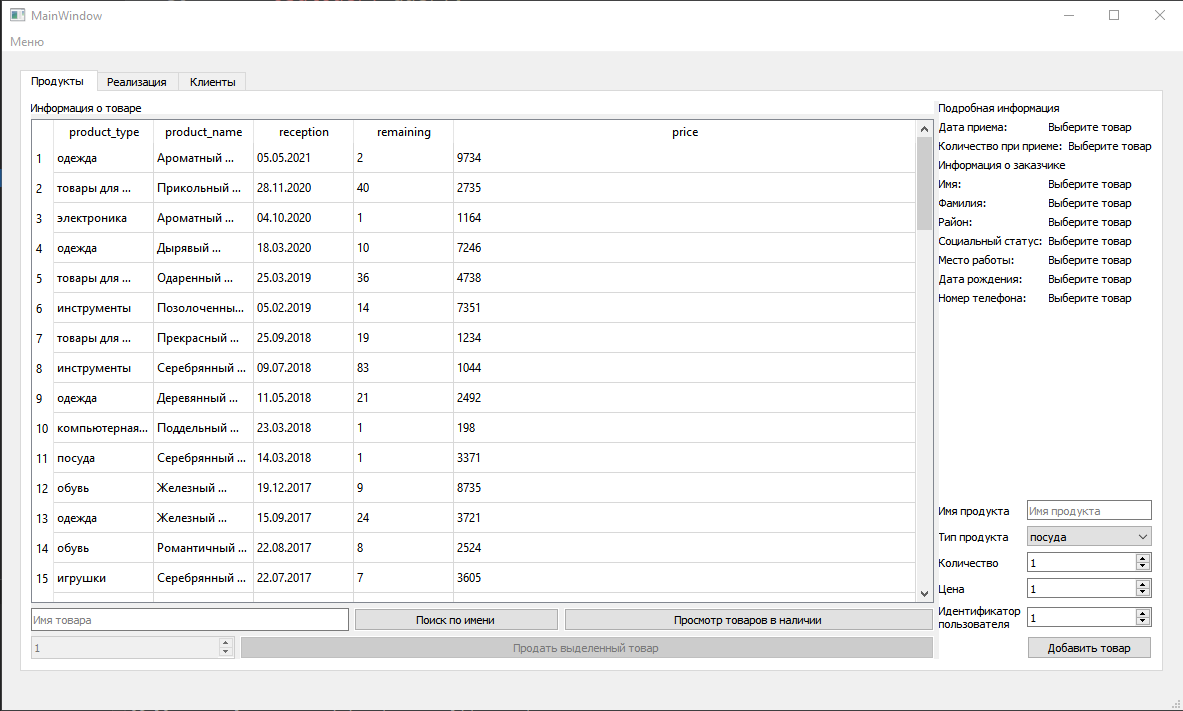


Рисунок 6.2 – Пока не выбран товар кнопка продажи неактивна

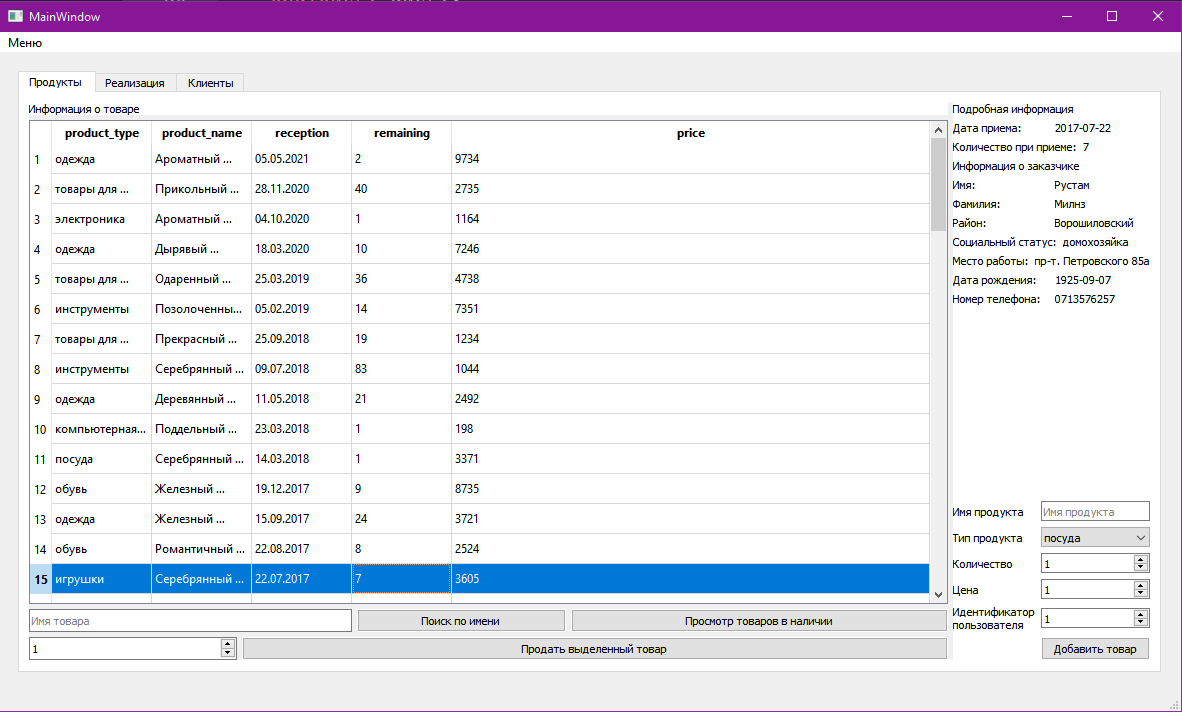


Рисунок 6.3 – При выборе товара она становится активна

Также обрабатывается возможная ошибка пользователя (рис. 6.4).

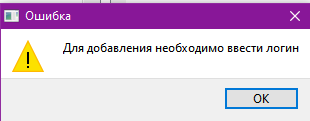


Рисунок 6.4 – Администратор пытается добавить пользователя с пустым логином

6.2 Каскадное удаление

При проектировании приложения был сделан упор на то, что никто не должен уметь затирать или изменять операции, которые уже были выполнены. Это делается для того, чтобы минимизировать риск мошенничества.

Каскадное удаление реализовано при создании таблиц (п. 4.2). Но продемонстрировать его не представляется возможным по вышеописанной причине.

# Заключение

Результат работы – система, реализующая систему для учета деятельности комиссионных магазинов города, реализованная при помощи языка программирования C++ и системы управления базами данных PostgreSQL.

К преимуществам программы относятся: удобный пользовательский интерфейс, реализация и управление базой данных с помощью клиентского интерфейса, возможность добавлять, удалять, редактировать, выбирать данные из таблиц. Возможен экспорт данных в excel. Безопасность для пользователя от случайного удаления данных. Возможен вывод диаграмм со статистическими данными. Просмотр результатов запросов различной сложности. Разграничение возможностей различных ролей. Возможность просматривать диаграммы.

При дальнейшей работе над системой можно смасштабировать ее до более крупных размеров, а также оптимизировать внутреннюю логику сервера для более эффективной работы с данными.

# Список литературы

1. Основы технологий баз данных: учеб. пособие/ Б.А. Новиков, Е.А.Горшкова; под ред. Е.В. Рогова. – М.:ДМК Пресс, 2019. -240с
2. Малыхина, М. П. / Базы данных. Основы, проектирование, использование. — СПб.: БХВ-Петербург, 2004. — 512 с.
3. Сибилев В. Д. / Моделирование и проектирование баз данных. В 2 ч.: Учеб. пособие для вузов. — Томск: Изд-во ТУСУР, 2002. — 144 с.
4. Петров В. Н. / Информационные системы: Учеб. для вузов. — СПб.: Питер, 2003. — 688 с.
5. Хомоненко А. Д., Цыганков В. М., Мальцев М. Г. / Базы данных. Учебник для вузов — 4-е издание, доп. и перераб. — СПб.: КОРОНА принт, 2004. — 736 с.
6. Черенков А. П. / Информационные системы для экономистов: Учеб. пособие для вузов. — М.: Экзамен, 2004. — 192 с.

# Приложение А. Техническое задание

# Приложение Б. Листинг клиентского приложения

administratorform.cpp

#include "administratorform.h"

#include "ui\_administratorform.h"

#include "QSql"

#include "QSqlQueryModel"

#include "helper.hpp"

#include <QtDebug>

#include <QSqlError>

#include <QMessageBox>

*struct* **dictionary\_data**{

QString name;

QString colunmn\_name;

};

Q\_DECLARE\_METATYPE(dictionary\_data);

AdministratorForm::**AdministratorForm**(QWidget \*parent) :

QWidget(*parent*),

ui(*new* Ui::AdministratorForm)

{

ui->setupUi(*this*);

helper::setupTableView(*ui->owners\_table1*);

helper::setupTableView(*ui->workers\_table1*);

helper::setupTableView(*ui->dictionary\_table2*);

QSqlQuery role\_query("set role administrator");

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(QString("select r.rolname from pg\_roles r,"

" pg\_group g where g.groname = \'holder\' and ARRAY[oid] && g.grolist "

"and r.rolname like \'%%1%\';").arg(ui->owner\_login\_le1->text()));

ui->owners\_table1->*setModel*(*query\_model*);

QSqlQueryModel\* query\_model2 = *new* QSqlQueryModel(*this*);

query\_model2->setQuery("select id, login, is\_available, "

"(select name from shops where id = id\_shop) from workers "

"order by id desc");

ui->workers\_table1->*setModel*(*query\_model2*);

*auto* query = helper::sendQuery("select \* from shops");

*while* (query.next()) {

ui->shop\_cb1->addItem(query.value(1).toString(), query.value(0));

}

QVariant v;

v.setValue(dictionary\_data{"districts", "district"});

ui->current\_dictionary\_cb2->addItem("Районы", v);

v.setValue(dictionary\_data{"statuses", "status"});

ui->current\_dictionary\_cb2->addItem("Социальное положение", v);

v.setValue(dictionary\_data{"product\_types", "product\_type"});

ui->current\_dictionary\_cb2->addItem("Типы товаров", v);

v.setValue(dictionary\_data{"shops", "name"});

ui->current\_dictionary\_cb2->addItem("Магазины", v);

int index = ui->current\_dictionary\_cb2->currentIndex();

QString dictionary\_name = qvariant\_cast<dictionary\_data>

(ui->current\_dictionary\_cb2->itemData(index)).name;

QString colunmn\_name = qvariant\_cast<dictionary\_data>

(ui->current\_dictionary\_cb2->itemData(index)).colunmn\_name;

QSqlQueryModel\* query\_model3 = *new* QSqlQueryModel(*this*);

query\_model3->setQuery(QString("select \* from %1 where %2 like \'%%3%\'").arg(

*dictionary\_name*,

*colunmn\_name*,

ui->name\_le2->text()));

ui->dictionary\_table2->*setModel*(*query\_model3*);

}

AdministratorForm::~***AdministratorForm***()

{

*delete* ui;

}

void AdministratorForm::**on\_find\_owner\_b1\_clicked**()

{

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(QString("select r.rolname from pg\_roles r,"

" pg\_group g where g.groname = \'holder\' and ARRAY[oid] && g.grolist "

"and r.rolname like \'%%1%\';").arg(ui->owner\_login\_le1->text()));

ui->owners\_table1->*setModel*(*query\_model*);

}

void AdministratorForm::**on\_add\_owner\_b1\_clicked**()

{

QSqlQuery query(QString("create role %1 login password \'%2\' in role holder").arg(

ui->owner\_login\_le1->text(), ui->owner\_password\_le1->text()));

helper::reloadModel(*ui->owners\_table1*);

}

void AdministratorForm::**on\_delete\_owner\_b1\_clicked**()

{

*//* *Как* *выбрать* *нужную* *запись* *в* *выбранной* *строке?*

*//* *auto* *model* *=* *index.model()->index(index.row(),* *0);*

*auto* selected\_owner = helper::data(ui->owners\_table1->currentIndex(),0);

QSqlQuery query(QString("drop role %1").arg(selected\_owner.toString()));

helper::reloadModel(*ui->owners\_table1*);

*//* *qDebug()* *<<* *selected\_owner;*

}

void AdministratorForm::**on\_owners\_table1\_clicked**(*const* QModelIndex &index)

{

}

void AdministratorForm::**on\_find\_worker\_b1\_clicked**()

{

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery("select id, login, is\_available, "

"(select name from shops where id = id\_shop) from workers "

"order by id desc");

ui->workers\_table1->*setModel*(*query\_model*);

}

void AdministratorForm::**on\_add\_worker\_b1\_clicked**()

{

*if* (ui->worker\_login\_le1->text() == ""){

QMessageBox::warning(*nullptr*, "Ошибка", "Для добавления необходимо ввести логин");

}*else*{

QSqlQuery query(QString("select create\_worker(\'%1\', \'%2\', %3)").arg(

ui->worker\_login\_le1->text(),

ui->worker\_password\_le1->text(),

ui->shop\_cb1->currentData().toString()));

qDebug() << query.lastError();

helper::reloadModel(*ui->workers\_table1*);

}

}

void AdministratorForm::**on\_delete\_worker\_b1\_clicked**()

{

QSqlQuery query(QString("select fire\_worker(\'%1\')").arg(

helper::data(ui->workers\_table1->currentIndex(),1).toString()));

qDebug() << query.lastError();

helper::reloadModel(*ui->workers\_table1*);

}

void AdministratorForm::**on\_add\_worker\_b1\_2\_clicked**()

{

QSqlQuery query(QString("update workers set id\_shop = %1 "

"where id = %2").arg(ui->shop\_cb1->currentData().toString(),

helper::data(ui->workers\_table1->currentIndex(),0).toString()));

qDebug() << query.lastError();

helper::reloadModel(*ui->workers\_table1*);

}

void AdministratorForm::**on\_find\_b2\_clicked**()

{

int index = ui->current\_dictionary\_cb2->currentIndex();

QString dictionary\_name = qvariant\_cast<dictionary\_data>

(ui->current\_dictionary\_cb2->itemData(index)).name;

QString colunmn\_name = qvariant\_cast<dictionary\_data>

(ui->current\_dictionary\_cb2->itemData(index)).colunmn\_name;

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(QString("select \* from %1 where %2 like \'%%3%\'").arg(

*dictionary\_name*,

*colunmn\_name*,

ui->name\_le2->text()));

ui->dictionary\_table2->*setModel*(*query\_model*);

}

void AdministratorForm::**on\_add\_b2\_clicked**()

{

int index = ui->current\_dictionary\_cb2->currentIndex();

QString dictionary\_name = qvariant\_cast<dictionary\_data>

(ui->current\_dictionary\_cb2->itemData(index)).name;

QString colunmn\_name = qvariant\_cast<dictionary\_data>

(ui->current\_dictionary\_cb2->itemData(index)).colunmn\_name;

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(QString("insert into %1 (%2) values(\'%3\')").arg(

*dictionary\_name*,

*colunmn\_name*,

ui->new\_name\_le2->text()));

helper::reloadModel(*ui->dictionary\_table2->model()*);

qDebug() << query\_model->lastError();

*//* *ui->dictionary\_table2->setModel(query\_model);*

}

void AdministratorForm::**on\_current\_dictionary\_cb2\_activated**(int index)

{

QString dictionary\_name = qvariant\_cast<dictionary\_data>

(ui->current\_dictionary\_cb2->itemData(index)).name;

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(QString("select \* from %1").arg(

dictionary\_name));

ui->dictionary\_table2->*setModel*(*query\_model*);

}

void AdministratorForm::**on\_delete\_b2\_clicked**()

{

{

int index = ui->current\_dictionary\_cb2->currentIndex();

QString dictionary\_name = qvariant\_cast<dictionary\_data>

(ui->current\_dictionary\_cb2->itemData(index)).name;

QString colunmn\_name = qvariant\_cast<dictionary\_data>

(ui->current\_dictionary\_cb2->itemData(index)).colunmn\_name;

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(QString("delete from %1 where id = \'%2\'").arg(

*dictionary\_name*,

helper::data(ui->dictionary\_table2->currentIndex(),0).toString()));

helper::reloadModel(*ui->dictionary\_table2->model()*);

qDebug() << query\_model->lastError();

*//* *ui->dictionary\_table2->setModel(query\_model);*

}

}

administratorform.h

#ifndef ADMINISTRATORFORM\_H

#define ADMINISTRATORFORM\_H

#include <QWidget>

*namespace* **Ui** {

*class* **AdministratorForm**;

}

*class* **AdministratorForm** : *public* QWidget

{

Q\_OBJECT

*public*:

*explicit* **AdministratorForm**(QWidget \*parent = *nullptr*);

~***AdministratorForm***();

*private* slots:

void **on\_find\_owner\_b1\_clicked**();

void **on\_add\_owner\_b1\_clicked**();

void **on\_delete\_owner\_b1\_clicked**();

void **on\_owners\_table1\_clicked**(*const* QModelIndex &index);

void **on\_find\_worker\_b1\_clicked**();

void **on\_add\_worker\_b1\_clicked**();

void **on\_delete\_worker\_b1\_clicked**();

void **on\_add\_worker\_b1\_2\_clicked**();

void **on\_find\_b2\_clicked**();

void **on\_add\_b2\_clicked**();

void **on\_current\_dictionary\_cb2\_activated**(int index);

void **on\_delete\_b2\_clicked**();

*private*:

Ui::AdministratorForm \*ui;

};

#endif *//* *ADMINISTRATORFORM\_H*

authorization.cpp

#include "authorization.h"

#include "ui\_authorization.h"

Authorization::**Authorization**(QWidget \*parent) :

QWidget(*parent*),

ui(*new* Ui::Authorization)

{

ui->setupUi(*this*);

}

Authorization::~***Authorization***()

{

*delete* ui;

}

void Authorization::**ClearForm**(){

ui->login\_le->clear();

ui->password\_le->clear();

}

void Authorization::**on\_login\_button\_clicked**()

{

emit TryToLogin(ui->login\_le->text(), ui->password\_le->text());

}

void Authorization::**on\_exit\_button\_clicked**()

{

exit(0);

}

authorization.h

#ifndef AUTHORIZATION\_H

#define AUTHORIZATION\_H

#include <QWidget>

*namespace* **Ui** {

*class* **Authorization**;

}

*class* **Authorization** : *public* QWidget

{

Q\_OBJECT

*public*:

*explicit* **Authorization**(QWidget \*parent = *nullptr*);

~***Authorization***();

void **ClearForm**();

signals:

void **TryToLogin**(QString login,QString password);

*private* slots:

void **on\_login\_button\_clicked**();

void **on\_exit\_button\_clicked**();

*private*:

Ui::Authorization \*ui;

};

#endif *//* *AUTHORIZATION\_H*

helper.cpp

#pragma once

#include "helper.hpp"

#include <QtSql>

#include <QHeaderView>

#include <QMessageBox>

*namespace* **helper** {

void **debugQuery**(QSqlQuery& query) {

*if* (!query.exec()) {

qDebug() << "Query error:";

qDebug() << "Last query: " << query.lastQuery();

qDebug() << "Error: " << query.lastError().text();

*return*;

}

*auto* record = query.record();

*auto* record\_size = record.count();

*while*(query.next()) {

*for* (int i = 0; i < record\_size; ++i) {

qDebug().nospace() << record.field(i).name() << ": " << query.value(i);

}

qDebug().noquote() << "\n";

}

qDebug() << "Last query: " << query.lastQuery();

}

void **setupTableView**(QTableView\* table\_view) {

table\_view->horizontalHeader()->setStretchLastSection(*true*);

table\_view->setSelectionMode(QTableView::*SingleSelection*);

table\_view->setSelectionBehavior(QTableView::*SelectRows*);

*//* *table\_view->verticalHeader()->hide();*

}

QVariant **data**(*const* QModelIndex& index, int column\_number) {

*auto* result\_index = index.model()->*index*(index.row(), column\_number);

*auto* result\_data = result\_index.data();

*return* result\_data;

}

void **reloadModel**(QAbstractItemModel\* model) {

*auto* query\_model = *dynamic\_cast*<QSqlQueryModel\*>(model);

assert(query\_model);

query\_model->setQuery(query\_model->query().lastQuery());

}

void **reloadModel**(QTableView\* view) {

reloadModel(*view->model()*);

}

QSqlQuery **sendQuery**(*const* QString& query) {

QSqlQuery sql\_api;

sql\_api.prepare(query);

*if* (!sql\_api.exec()) {

QMessageBox::warning(*nullptr*, "Ошибка", sql\_api.lastError().text());

}

*return* sql\_api;

}

}

helper.hpp

#ifndef HELPER\_HPP

#define HELPER\_HPP

#include <QSqlQuery>

#include <QTableView>

*namespace* **helper** {

void **debugQuery**(QSqlQuery& query);

void **setupTableView**(QTableView\* table\_view);

QVariant **data**(*const* QModelIndex& index, int column\_number);

void **reloadModel**(QAbstractItemModel\* model);

void **reloadModel**(QTableView\* view);

QSqlQuery **sendQuery**(*const* QString& query);

}

#endif *//* *HELPER\_HPP*

holderform.cpp

#include "holderform.h"

#include "ui\_holderform.h"

#include "helper.hpp"

#include <QtDebug>

#include <QtSql>

#include <QVariant>

#include <QMessageBox>

#include <QtCharts>

#include "xlsxdocument.h"

#include "xlsxchartsheet.h"

#include "xlsxcellrange.h"

#include "xlsxchart.h"

#include "xlsxrichstring.h"

#include "xlsxworkbook.h"

void **setupQuery\_cb2**(Ui::HolderForm\* ui);

*struct* **query\_data**{

bool have\_arguments;

QString query;

};

*struct* **table\_data**{

QString name;

QString colunmn\_name;

QString label\_text;

};

Q\_DECLARE\_METATYPE(query\_data);

Q\_DECLARE\_METATYPE(table\_data);

HolderForm::**HolderForm**(QWidget \*parent) :

QWidget(*parent*),

ui(*new* Ui::HolderForm)

{

ui->setupUi(*this*);

helper::setupTableView(*ui->query\_table2*);

helper::setupTableView(*ui->tables\_table2*);

setupQuery\_cb2(*ui*);

QVariant v;

v.setValue(table\_data{"products\_view", "product\_name", "Название продукта:"});

ui->current\_table\_cb2->addItem("Товары", v);

v.setValue(table\_data{"customers\_view", "second\_name", "Фамилия:"});

ui->current\_table\_cb2->addItem("Заказчики", v);

v.setValue(table\_data{"realization\_view", "product\_name",

"Название товара"});

ui->current\_table\_cb2->addItem("Реализация", v);

v.setValue(table\_data{"workers\_view", "login", "Логин:"});

ui->current\_table\_cb2->addItem("Работники", v);

int index = ui->current\_table\_cb2->currentIndex();

QString table\_name = qvariant\_cast<table\_data>

(ui->current\_table\_cb2->itemData(index)).name;

QString colunmn\_name = qvariant\_cast<table\_data>

(ui->current\_table\_cb2->itemData(index)).colunmn\_name;

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(QString("select \* from %1").arg(

table\_name));

ui->tables\_table2->*setModel*(*query\_model*);

}

HolderForm::~***HolderForm***()

{

*delete* ui;

}

void **setupQuery\_cb2**(Ui::HolderForm\* ui){

QVariant v;

v.setValue(query\_data{*true*, "query\_1\_1"});

ui->query\_cb2->addItem("1.1 Вывод продуктов определенного типа (По внешнему ключу)", v);

v.setValue(query\_data{*true*, "query\_1\_2"});

ui->query\_cb2->addItem("1.2 Вывод клиентов живущих в определенном районе (По внешнему ключу)", v);

v.setValue(query\_data{*true*, "query\_1\_3"});

ui->query\_cb2->addItem("1.3 Пользователи, которые родились позже указанной даты (По дате)", v);

v.setValue(query\_data{*true*, "query\_1\_4"});

ui->query\_cb2->addItem("1.4 Продукты, которые были приняты после указанной даты (По дате)", v);

v.setValue(query\_data{*false*, "query\_2\_1"});

ui->query\_cb2->addItem("2.1 Информация о клиентах и состояниях сданных ими товаров", v);

v.setValue(query\_data{*false*, "query\_2\_2"});

ui->query\_cb2->addItem("2.2 Информация о продуктах и их реализации", v);

v.setValue(query\_data{*false*, "query\_2\_3"});

ui->query\_cb2->addItem("2.3 Информация о раелизации товарас указанием магазина", v);

v.setValue(query\_data{*false*, "query\_3\_1"});

ui->query\_cb2->addItem("3.1 Все продукты и их реализация", v);

v.setValue(query\_data{*false*, "query\_4\_1"});

ui->query\_cb2->addItem("4.1 Реализация продукта с указанием работника", v);

v.setValue(query\_data{*false*, "query\_5\_1"});

ui->query\_cb2->addItem("5.1 Реализация и работник который принял", v);

v.setValue(query\_data{*false*, "query\_6\_1"});

ui->query\_cb2->addItem("6.1 Проданные товары с магазином", v);

v.setValue(query\_data{*false*, "query\_7\_1"});

ui->query\_cb2->addItem("7.1 Количество товаров и проданных товаров", v);

v.setValue(query\_data{*true*, "query\_8\_1"});

ui->query\_cb2->addItem("8.1 Продукты, которых на складе больше указанного", v);

v.setValue(query\_data{*true*, "query\_8\_2"});

ui->query\_cb2->addItem("8.2 Поиск клиента по части номера общее количество его товаров", v);

v.setValue(query\_data{*true*, "query\_8\_3"});

ui->query\_cb2->addItem("8.3 Количество чеков выписанных, на этот товар", v);

v.setValue(query\_data{*true*, "query\_8\_4"});

ui->query\_cb2->addItem("8.4 Количество работников магазина", v);

v.setValue(query\_data{*true*, "query\_9\_1"});

ui->query\_cb2->addItem("9.1 Продукты, продажи которых больше указанного", v);

v.setValue(query\_data{*true*, "query\_10\_1"});

ui->query\_cb2->addItem("10.1 Пользователи с указанным социальным статусом и количеством сданных товаров больше указанного", v);

v.setValue(query\_data{*false*, "query\_11\_1"});

ui->query\_cb2->addItem("11.1 Средняя стоимость всех товаров магазина", v);

v.setValue(query\_data{*false*, "query\_12\_1"});

ui->query\_cb2->addItem("12.1 Продукты которые еще не покупали и продукты, которые полностью выкуплены", v);

v.setValue(query\_data{*false*, "query\_13\_1"});

ui->query\_cb2->addItem("13.1 Информация о пользователях, товары которых еще есть в наличии", v);

v.setValue(query\_data{*false*, "query\_13\_2"});

ui->query\_cb2->addItem("13.2 Информация о пользователях, товаров которых уже нет в наличии", v);

v.setValue(query\_data{*false*, "query\_13\_3"});

ui->query\_cb2->addItem("13.3 Популярность продукта у покупателей", v);

v.setValue(query\_data{*false*, "query\_13\_4"});

ui->query\_cb2->addItem("13.4 Продукты со стоимостью больше средней", v);

}

void HolderForm::**on\_query\_cb2\_currentIndexChanged**(int index)

{

bool have\_arguments = qvariant\_cast<query\_data>

(ui->query\_cb2->itemData(index)).have\_arguments;

QString name = qvariant\_cast<query\_data>

(ui->query\_cb2->itemData(index)).query;

ui->argument\_le2->setEnabled(have\_arguments);

*if* (name == "query\_10\_1"){

ui->argument2\_le2->setEnabled(*true*);

} *else* {

ui->argument2\_le2->setEnabled(*false*);

}

}

void HolderForm::**on\_query\_b2\_clicked**()

{

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

int index = ui->query\_cb2->currentIndex();

bool have\_arguments = qvariant\_cast<query\_data>

(ui->query\_cb2->itemData(index)).have\_arguments;

QString name = qvariant\_cast<query\_data>

(ui->query\_cb2->itemData(index)).query;

*if*(have\_arguments){

QSqlQuery query;

*if* (name == "query\_10\_1"){

query.prepare(QString("select \* from %1(\'%2\', %3)").arg(

*name*,

ui->argument\_le2->text(),

ui->argument2\_le2->text()

));

} *else* {

query.prepare(QString("select \* from %1(\'%2\')").arg(

*name*,

ui->argument\_le2->text()

));

}

*if*(!query.exec()){

QMessageBox::warning(*nullptr*, "Ошибка", "Введите параметр, подходящий под запрос");

}

query\_model->setQuery(query);

ui->query\_table2->*setModel*(*query\_model*);

*//* *qDebug()* *<<* *"Function* *"* *<<* *name;*

}*else* {

query\_model->setQuery(QString("select \* from %1").arg(name));

*//* *qDebug()* *<<* *"View"* *<<* *name;*

ui->query\_table2->*setModel*(*query\_model*);

}

}

void HolderForm::**on\_find\_b2\_clicked**()

{

int index = ui->current\_table\_cb2->currentIndex();

QString table\_name = qvariant\_cast<table\_data>

(ui->current\_table\_cb2->itemData(index)).name;

QString colunmn\_name = qvariant\_cast<table\_data>

(ui->current\_table\_cb2->itemData(index)).colunmn\_name;

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(QString("select \* from %1 where %2 like \'%%3%\'").arg(

*table\_name*,

*colunmn\_name*,

ui->name\_le2->text()));

ui->tables\_table2->*setModel*(*query\_model*);

}

void HolderForm::**on\_current\_table\_cb2\_activated**(int index)

{

QString table\_name = qvariant\_cast<table\_data>

(ui->current\_table\_cb2->itemData(index)).name;

QString label\_text = qvariant\_cast<table\_data>

(ui->current\_table\_cb2->itemData(index)).label\_text;

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

ui->label->setText(label\_text);

query\_model->setQuery(QString("select \* from %1").arg(

table\_name));

ui->tables\_table2->*setModel*(*query\_model*);

}

void HolderForm::**on\_chart\_button\_clicked**()

{

*auto* series = *new* QPieSeries();

*auto* cursor = helper::sendQuery("select \* from test\_view");

*while* (cursor.next()) {

series->append(cursor.value(0).toString(), cursor.value(1).toInt());

}

series->setLabelsVisible(*true*);

*auto* chart = *new* QChart;

chart->addSeries(*series*);

chart->setAnimationOptions(QChart::*AllAnimations*);

chart->setTitle("Количество проданных продуктов разных типов");

// chart->legend()->setAlignment(*Qt::AlignRight);*

chart->legend()->hide();

ui->chart->setChart(*chart*);

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery("select \* from test\_view");

ui->view\_table3->*setModel*(*query\_model*);

ui->chart->setRenderHint(QPainter::*Antialiasing*);

}

void HolderForm::**on\_import\_button\_clicked**()

{

QXlsx::Document xlsxW;

*auto* query = helper::sendQuery("select \* from test\_view");

*auto* record = query.record();

*auto* record\_size = record.count();

*for* (int i = 1; i <= record\_size; ++i) {

xlsxW.write(1, i, record.fieldName(i - 1));

}

*for* (int i = 2; query.next(); ++i) {

*for* (int j = 1; j <= record\_size; ++j) {

xlsxW.write(i, j, query.value(j - 1));

}

}

*const* *auto* path = QFileDialog::getSaveFileName(*nullptr*, "Выберите место созранения", "", "Excel (\*.xlsx)");

xlsxW.saveAs(path);

}

holderform.h

#ifndef HOLDERFORM\_H

#define HOLDERFORM\_H

#include <QWidget>

*namespace* **Ui** {

*class* **HolderForm**;

}

*class* **HolderForm** : *public* QWidget

{

Q\_OBJECT

*public*:

*explicit* **HolderForm**(QWidget \*parent = *nullptr*);

~***HolderForm***();

*private* slots:

void **on\_query\_cb2\_currentIndexChanged**(int index);

void **on\_query\_b2\_clicked**();

void **on\_find\_b2\_clicked**();

void **on\_current\_table\_cb2\_activated**(int index);

void **on\_chart\_button\_clicked**();

void on\_import\_button\_clicked();

*private*:

Ui::HolderForm \*ui;

};

#endif *//* *HOLDERFORM\_H*

main.cpp

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(*argc*, *argv*);

MainWindow w;

w.show();

*return* a.exec();

}

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "authorization.h"

#include "workerform.h"

#include "administratorform.h"

#include "holderform.h"

#include <QDebug>

#include <QMessageBox>

#include <QtSql>

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(*new* Ui::MainWindow) {

ui->setupUi(*this*);

*auto* auth = *new* Authorization(*this*);

SetPage(*auth*);

setGeometry(300, 200, 218, 99+40);

connect(auth, &Authorization::TryToLogin, *this*, [*this*](*const* QString& login,

*const* QString& password){

ConnectDB(login, password);

QSqlQuery q;

q.exec("select current\_user");

q.first();

qDebug() << q.value(0).toString();

*if* (CheckUserGroup("worker")){

SetPage(*new* WorkerForm);

} *else* *if* (CheckUserGroup("administrator")) {

SetPage(*new* AdministratorForm);

} *else* *if* (CheckUserGroup("holder")){

SetPage(*new* HolderForm);

}

*//* *SetPage(new* *Formtest);*

});

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

void MainWindow::**SetPage**(QWidget \*w){

resize(w->width()+18,w->height()+60);

ui->stackedWidget->addWidget(*w*);

ui->stackedWidget->setCurrentWidget(*w*);

qDebug() << ui->stackedWidget->currentWidget()->objectName();

}

bool MainWindow::**PreviousPage**(){

int last = ui->stackedWidget->count() - 1;

*auto*\* last\_widget = ui->stackedWidget->widget(last);

ui->stackedWidget->removeWidget(*last\_widget*);

*return* last;

}

void MainWindow::**ConnectDB**(QString login, QString password){

database = QSqlDatabase::addDatabase("QPSQL");

database.setDatabaseName("store");

database.setHostName("localhost");

database.setPassword(password);

database.setPort(5432);

database.setUserName(login);

*if* (database.open()) {

qDebug() << "Database have been open";

} *else* {

QMessageBox::warning(*nullptr*, "Ошибка", "Неверный логин или пароль");

qDebug() << "Database have not been open";

qDebug() << database.lastError();

}

}

bool MainWindow::**CheckUserGroup**(*const* QString& role){

QSqlQuery query;

query.prepare("select pg\_has\_role(current\_user, :role, \'USAGE\')");

query.bindValue(":role", role);

query.exec();

query.first();

*return* query.value(0).toBool();

}

void MainWindow::**on\_stackedWidget\_currentChanged**(int)

{

}

void MainWindow::**on\_logout\_triggered**()

{

database.close();

QSize size1(218, 99+40);

*for* (int i = 1; i < ui->stackedWidget->count(); i++) {

*auto*\* widget = ui->stackedWidget->widget(i);

ui->stackedWidget->removeWidget(*widget*);

*delete* widget;

}

*auto*\* authorization = ui->stackedWidget->widget(0);

*auto*\* authorization\_widget = *dynamic\_cast*<Authorization\*>(authorization);

assert(authorization\_widget);

authorization\_widget->ClearForm();

resize(size1);

}

mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QStackedWidget>

#include <QSqlDatabase>

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **on\_stackedWidget\_currentChanged**(int arg1);

void **on\_logout\_triggered**();

*private*:

Ui::MainWindow \*ui;

QSqlDatabase database;

void **SetPage**(QWidget \*w);

bool **PreviousPage**();

void **ConnectDB**(QString login, QString passwodr);

bool **CheckUserGroup**(*const* QString& role);

};

#endif *//* *MAINWINDOW\_H*

workerform.cpp

#include "workerform.h"

#include "ui\_workerform.h"

#include <QSqlQuery>

#include <QSql>

#include <QSqlQueryModel>

#include <QDebug>

#include <QSqlError>

#include "helper.hpp"

#include <QMessageBox>

WorkerForm::**WorkerForm**(QWidget \*parent) :

QWidget(*parent*),

ui(*new* Ui::WorkerForm)

{

ui->setupUi(*this*);

helper::setupTableView(*ui->realizations\_table2*);

helper::setupTableView(*ui->products\_table\_view*);

helper::setupTableView(*ui->products\_table2*);

helper::setupTableView(*ui->products\_table3*);

helper::setupTableView(*ui->clients\_table3*);

*auto* query = helper::sendQuery("select \* from districts");

*while* (query.next()) {

ui->district\_cb3->addItem(query.value(1).toString(), query.value(0));

}

query = helper::sendQuery("select \* from statuses");

*while* (query.next()) {

ui->status\_cb3->addItem(query.value(1).toString(), query.value(0));

}

query = helper::sendQuery("select \* from product\_types");

*while* (query.next()) {

ui->product\_type\_cb1->addItem(query.value(1).toString(), query.value(0));

}

*//* *Setup* *products\_view\_table*

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery("select \* from products\_modify\_view where remaining > 0");

ui->products\_table\_view->*setModel*(*query\_model*);

ui->sell\_product\_button->setEnabled(*false*);

ui->realization\_count\_spin1->setEnabled(*false*);

ui->products\_table\_view->hideColumn(0);

ui->products\_table\_view->hideColumn(6);

ui->products\_table\_view->hideColumn(7);

ui->products\_table\_view->hideColumn(8);

ui->products\_table\_view->hideColumn(9);

ui->products\_table\_view->hideColumn(10);

*//* *Setup* *client\_table3*

*auto* client\_q = QString("select c.id, c.first\_name, c.second\_name, c.phone, c.work "

"from customers c "

"where first\_name like \'%%1%\' "

"and second\_name like \'%%2%\' "

"group by c.id order by c.id desc").arg(ui->first\_name\_search\_le3->text(), ui->second\_name\_search\_le3->text());

QSqlQueryModel\* client\_qm = *new* QSqlQueryModel(*this*);

client\_qm->setQuery(client\_q);

ui->clients\_table3->*setModel*(*client\_qm*);

ui->phone\_add\_le3->setValidator(*new* QRegExpValidator(QRegExp("[0-9]{1,10}"), *this*));

}

WorkerForm::~***WorkerForm***()

{

*delete* ui;

}

void WorkerForm::**SelectCustomer**(int id\_customer)

{

QSqlQuery query;

query.prepare("select \* from customers\_view where id = :id\_customer");

query.bindValue(":id\_customer", id\_customer);

query.exec();

query.next();

qDebug() << query.lastError();

ui->first\_name\_label->setText(query.value("first\_name").toString());

ui->second\_name\_label->setText(query.value("second\_name").toString());

ui->district\_label->setText(query.value("district").toString());

ui->status\_label->setText(query.value("status").toString());

ui->work\_label->setText(query.value("work").toString());

ui->birthday\_label->setText(query.value("birthday").toString());

ui->phone\_label->setText(query.value("phone").toString());

}

int WorkerForm::**SelectProduct**(int quittance)

{

QSqlQuery query;

query.prepare("select \* from products where id = :quittance");

query.bindValue(":quittance", quittance);

query.exec();

query.next();

ui->reception\_label->setText(query.value("reception").toString());

ui->count\_label->setText(query.value("count").toString());

*return* query.value("id\_customer").toInt();

}

void WorkerForm::**on\_select\_products\_button\_clicked**()

{

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery("select \* from products\_modify\_view where remaining > 0");

ui->products\_table\_view->*setModel*(*query\_model*);

ui->sell\_product\_button->setEnabled(*false*);

ui->realization\_count\_spin1->setEnabled(*false*);

ui->products\_table\_view->hideColumn(0);

ui->products\_table\_view->hideColumn(6);

ui->products\_table\_view->hideColumn(7);

ui->products\_table\_view->hideColumn(8);

ui->products\_table\_view->hideColumn(9);

ui->products\_table\_view->hideColumn(10);

qDebug() << query\_model->lastError();

}

void WorkerForm::**on\_products\_table\_view\_clicked**(*const* QModelIndex &index)

{

*auto* model = index.model()->*index*(index.row(), 0);

int customer\_id = SelectProduct(model.data().toInt());

SelectCustomer(customer\_id);

model = index.model()->*index*(index.row(), 4);

ui->realization\_count\_spin1->setMaximum(model.data().toInt());

ui->sell\_product\_button->setEnabled(*true*);

ui->realization\_count\_spin1->setEnabled(*true*);

}

void WorkerForm::**on\_search\_button\_clicked**()

{

*auto* query = QString("select \* from products\_modify\_view "

"where product\_name like \'%%1%\' and remaining > 0").arg(ui->product\_name\_le->text());

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->products\_table\_view->*setModel*(*query\_model*);

ui->sell\_product\_button->setEnabled(*false*);

ui->realization\_count\_spin1->setEnabled(*false*);

ui->products\_table\_view->hideColumn(0);

ui->products\_table\_view->hideColumn(6);

ui->products\_table\_view->hideColumn(7);

ui->products\_table\_view->hideColumn(8);

ui->products\_table\_view->hideColumn(9);

ui->products\_table\_view->hideColumn(10);

}

void WorkerForm::**on\_sell\_product\_button\_clicked**()

{

QSqlQuery query;

QString query\_str = QString{"select sell\_product(%1, %2)"}.arg(

helper::data(ui->products\_table\_view->currentIndex(), 0).toString(),

ui->realization\_count\_spin1->text());

query.exec(query\_str);

helper::reloadModel(*ui->products\_table\_view->model()*);

ui->realization\_count\_spin1->setEnabled(*false*);

ui->sell\_product\_button->setEnabled(*false*);

*//* *qDebug()* *<<* *query.lastError();*

*//* *qDebug()* *<<* *helper::data(ui->products\_table\_view->currentIndex(),* *0);*

}

void WorkerForm::**on\_products\_find\_button2\_clicked**()

{

*auto* query = QString("select \* from products\_modify\_view "

"where product\_name like \'%%1%\'").arg(ui->product\_name\_le2->text());

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->products\_table2->*setModel*(*query\_model*);

ui->products\_table2->hideColumn(0);

ui->products\_table2->hideColumn(6);

ui->products\_table2->hideColumn(7);

ui->products\_table2->hideColumn(8);

ui->products\_table2->hideColumn(9);

ui->products\_table2->hideColumn(10);

}

void WorkerForm::**on\_products\_table2\_clicked**(*const* QModelIndex &index)

{

*auto* model = index.model()->*index*(index.row(), 0);

int quittance = model.data().toInt();

*auto* query = QString("select r.ticket, p.product\_name, p.quittance, r.realization\_date, r.realization\_count "

"from realization r "

"inner join products p on r.id\_product = p.id "

"where r.id\_product = %1 "

"order by quittance").arg(quittance);

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->realizations\_table2->*setModel*(*query\_model*);

}

void WorkerForm::**on\_realization\_find\_button2\_clicked**()

{

*auto* query = QString("select r.ticket, p.product\_name, p.quittance, r.realization\_date, r.realization\_count "

"from realization r "

"inner join products p on r.id\_product = p.id "

"where realization\_count > %1 "

"order by quittance").arg(ui->realization\_count\_sb2->text());

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->realizations\_table2->*setModel*(*query\_model*);

}

void WorkerForm::**on\_realizations\_table2\_clicked**(*const* QModelIndex &index)

{

*auto* model = index.model()->*index*(index.row(), 2);

int quittance = model.data().toInt();

*auto* query = QString("select \* from products\_modify\_view "

"where quittance = %1").arg(quittance);

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->products\_table2->*setModel*(*query\_model*);

ui->products\_table2->hideColumn(0);

ui->products\_table2->hideColumn(6);

ui->products\_table2->hideColumn(7);

ui->products\_table2->hideColumn(8);

ui->products\_table2->hideColumn(9);

ui->products\_table2->hideColumn(10);

}

void WorkerForm::**on\_find\_client\_button3\_clicked**()

{

*auto* query = QString("select c.id, c.first\_name, c.second\_name, c.phone, c.work "

"from customers c "

"inner join products p on c.id = p.id\_customer where "

"first\_name like \'%%1%\' "

"and second\_name like \'%%2%\' "

"order by c.id desc").arg(ui->first\_name\_search\_le3->text(), ui->second\_name\_search\_le3->text());

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->clients\_table3->*setModel*(*query\_model*);

*//* *ui->clients\_table3->hideColumn(0);*

*//* *auto* *query* *=* *QString("select* *c.id,* *c.first\_name,* *c.second\_name,* *c.phone,* *c.work* *"*

*//"from* *customers* *c* *"*

*//"where* *first\_name* *like* *\'%%1%\'* *"*

*//"and* *second\_name* *like* *\'%%2%\'* *"*

*//"group* *by* *c.id* *order* *by* *c.id* *desc").arg(ui->first\_name\_search\_le3->text(),* *ui->second\_name\_search\_le3->text());*

*//* *QSqlQueryModel\** *query\_model* *=* *new* *QSqlQueryModel(this);*

*//* *query\_model->setQuery(query);*

*//* *ui->clients\_table3->setModel(query\_model);*

}

void WorkerForm::**on\_clients\_table3\_clicked**(*const* QModelIndex &index)

{

*auto* model = index.model()->*index*(index.row(), 0);

int customer\_id = model.data().toInt();

*auto* query = QString("select \* from products\_modify\_view "

"where quittance in (select quittance from products where id\_customer = %1)").arg(customer\_id);

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->products\_table3->*setModel*(*query\_model*);

ui->products\_table3->hideColumn(0);

ui->products\_table3->hideColumn(6);

ui->products\_table3->hideColumn(7);

ui->products\_table3->hideColumn(8);

ui->products\_table3->hideColumn(9);

ui->products\_table3->hideColumn(10);

}

void WorkerForm::**on\_find\_products\_button3\_clicked**()

{

*auto* query = QString("select \* from products\_modify\_view "

"where product\_name like \'%%1%\'").arg(ui->product\_name\_le3->text());

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->products\_table3->*setModel*(*query\_model*);

ui->products\_table3->hideColumn(0);

ui->products\_table3->hideColumn(6);

ui->products\_table3->hideColumn(7);

ui->products\_table3->hideColumn(8);

ui->products\_table3->hideColumn(9);

ui->products\_table3->hideColumn(10);

}

void WorkerForm::**on\_products\_table3\_clicked**(*const* QModelIndex &index)

{

*auto* model = index.model()->*index*(index.row(), 0);

int quittance = model.data().toInt();

*auto* query = QString("select c.id, c.first\_name, c.second\_name, c.phone, c.work "

"from customers c "

"inner join products p on c.id = p.id\_customer "

"where p.id\_shop = (select id from shops) "

"and p.quittance = %1 "

"group by c.id").arg(quittance);

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->clients\_table3->*setModel*(*query\_model*);

*//* *ui->clients\_table3->hideColumn(0);*

}

void WorkerForm::**on\_add\_client\_button3\_clicked**()

{

*auto* query = QString("insert into customers (first\_name, second\_name, id\_district, id\_status, work, birthday, phone)"

"VALUES (\'%1\', \'%2\', %3, %4, \'%5\', \'%6\', \'%7\')").arg(ui->first\_name\_add\_le3->text(),

ui->second\_name\_add\_le3->text(),

ui->district\_cb3->currentData().toString(),

ui->status\_cb3->currentData().toString(),

ui->work\_add\_le3->text(),

ui->birthday\_de3->text(),

ui->phone\_add\_le3->text());

QSqlQuery q(query);

helper::reloadModel(*ui->clients\_table3*);

qDebug() << q.lastError();

*//* *q.exec();*

}

void WorkerForm::**on\_find\_all\_client\_button3\_clicked**()

{

*auto* query = QString("select c.id, c.first\_name, c.second\_name, c.phone, c.work "

"from customers c "

"where first\_name like \'%%1%\' "

"and second\_name like \'%%2%\' "

"group by c.id order by c.id desc").arg(ui->first\_name\_search\_le3->text(), ui->second\_name\_search\_le3->text());

QSqlQueryModel\* query\_model = *new* QSqlQueryModel(*this*);

query\_model->setQuery(query);

ui->clients\_table3->*setModel*(*query\_model*);

*//* *ui->clients\_table3->hideColumn(0);*

}

void WorkerForm::**on\_add\_product\_button\_clicked**()

{

QSqlQuery query;

QString query\_str{

"insert into products\_modify\_view(product\_type, product\_name, "

"remaining, price, id\_customer) "

"values (\'%1\', \'%2\', %3, %4, %5)"};

query.prepare(query\_str.arg(ui->product\_type\_cb1->currentText(),

ui->product\_name\_le1->text(),

ui->count\_sb1->text(),

ui->price\_sb1->text(),

ui->id\_customer\_sb1->text()

));

*if* (!query.exec()) {

QMessageBox::warning(*nullptr*, "Ошибка", query.lastError().text());

qDebug() << query.lastQuery();

*return*;

}

ui->sell\_product\_button->setEnabled(*false*);

ui->realization\_count\_spin1->setEnabled(*false*);

helper::reloadModel(*ui->products\_table\_view->model()*);

}

workerform.h

#ifndef WORKERFORM\_H

#define WORKERFORM\_H

#include <QWidget>

*namespace* **Ui** {

*class* **WorkerForm**;

}

*class* **WorkerForm** : *public* QWidget

{

Q\_OBJECT

*public*:

*explicit* **WorkerForm**(QWidget \*parent = *nullptr*);

~***WorkerForm***();

void **SelectCustomer**(int id\_customer);

int **SelectProduct**(int quittance);

*private* slots:

void **on\_select\_products\_button\_clicked**();

void **on\_products\_table\_view\_clicked**(*const* QModelIndex &index);

void **on\_search\_button\_clicked**();

void **on\_sell\_product\_button\_clicked**();

void **on\_products\_find\_button2\_clicked**();

void **on\_products\_table2\_clicked**(*const* QModelIndex &index);

void **on\_realization\_find\_button2\_clicked**();

void **on\_realizations\_table2\_clicked**(*const* QModelIndex &index);

void **on\_find\_client\_button3\_clicked**();

void **on\_clients\_table3\_clicked**(*const* QModelIndex &index);

void **on\_find\_products\_button3\_clicked**();

void **on\_products\_table3\_clicked**(*const* QModelIndex &index);

void **on\_add\_client\_button3\_clicked**();

void **on\_find\_all\_client\_button3\_clicked**();

void **on\_add\_product\_button\_clicked**();

*private*:

Ui::WorkerForm \*ui;

};

#endif *//* *WORKERFORM\_H*

# Приложение В. Листинг серверного приложения

--

-- PostgreSQL database dump

--

-- Dumped from database version 13.2

-- Dumped by pg\_dump version 13.2

-- Started on 2021-06-04 08:56:05

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET idle\_in\_transaction\_session\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SELECT pg\_catalog.set\_config('search\_path', '', false);

SET check\_function\_bodies = false;

SET xmloption = content;

SET client\_min\_messages = warning;

SET row\_security = off;

--

-- TOC entry 763 (class 1247 OID 25129)

-- Name: phone\_number; Type: DOMAIN; Schema: public; Owner: postgres

--

CREATE DOMAIN public.phone\_number AS character(10)

CONSTRAINT phone\_number\_check CHECK ((length(VALUE) = 10));

ALTER DOMAIN public.phone\_number OWNER TO postgres;

--

-- TOC entry 788 (class 1247 OID 25257)

-- Name: positive\_num; Type: DOMAIN; Schema: public; Owner: postgres

--

CREATE DOMAIN public.positive\_num AS integer

CONSTRAINT positive CHECK ((VALUE > 0));

ALTER DOMAIN public.positive\_num OWNER TO postgres;

--

-- TOC entry 259 (class 1255 OID 25259)

-- Name: count\_check(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.count\_check() RETURNS trigger

LANGUAGE plpgsql

AS $$

begin

if new.count < 0 then

raise exception 'Нельзя столько продавать!';

else

return new;

end if;

end;

$$;

ALTER FUNCTION public.count\_check() OWNER TO postgres;

--

-- TOC entry 265 (class 1255 OID 16972)

-- Name: create\_worker(text, text, integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.create\_worker(username text, password text, shop\_id integer) RETURNS void

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

insert into workers values (default, username, shop\_id, true);

execute format('create role %s login password ''%s'' in role worker', username, password);

end;

$$;

ALTER FUNCTION public.create\_worker(username text, password text, shop\_id integer) OWNER TO postgres;

--

-- TOC entry 238 (class 1255 OID 16841)

-- Name: decrease\_products(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.decrease\_products() RETURNS trigger

LANGUAGE plpgsql

AS $$

begin

update products set remaining = (select remaining from products where id = new.id\_product)-new.realization\_count

where id = new.id\_product;

return new;

end;

$$;

ALTER FUNCTION public.decrease\_products() OWNER TO postgres;

--

-- TOC entry 267 (class 1255 OID 16931)

-- Name: delete\_from\_all\_tables(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.delete\_from\_all\_tables() RETURNS void

LANGUAGE plpgsql

AS $$

begin

delete

from customers

where true;

alter sequence customers\_id\_seq restart 1;

delete

from districts

where true;

alter sequence districts\_id\_seq restart 1;

delete

from statuses

where true;

alter sequence statuses\_id\_seq restart 1;

-- delete

-- from pg\_roles

-- where rolcanlogin = true

-- and ARRAY [oid] && (select grolist from pg\_group where groname = 'worker');

--

-- delete

-- from workers

-- where true;

-- alter sequence workers\_id\_seq restart 1;

delete

from product\_types

where true;

alter sequence product\_types\_id\_seq restart 1;

delete

from shops

where true;

alter sequence shops\_id\_seq restart 1;

delete

from products

where true;

alter sequence products\_id\_seq restart 1;

alter sequence quittance\_num restart 1;

delete

from realization

where true;

alter sequence ticket\_num restart 1;

alter sequence realization\_id\_seq restart 1;

end;

$$;

ALTER FUNCTION public.delete\_from\_all\_tables() OWNER TO postgres;

--

-- TOC entry 262 (class 1255 OID 26303)

-- Name: delete\_workers(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.delete\_workers() RETURNS void

LANGUAGE plpgsql

AS $$

declare

count integer;

begin

count = (select count(\*) from workers);

for i in 1..count

loop

execute format('drop role ' || (select login from workers limit 1));

end loop;

end;

$$;

ALTER FUNCTION public.delete\_workers() OWNER TO postgres;

--

-- TOC entry 242 (class 1255 OID 25229)

-- Name: fire\_worker(character varying); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.fire\_worker(login\_ character varying) RETURNS void

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

update workers set is\_available = false where login = login\_;

execute format('drop role %s ', login\_);

end;

$$;

ALTER FUNCTION public.fire\_worker(login\_ character varying) OWNER TO postgres;

--

-- TOC entry 268 (class 1255 OID 16932)

-- Name: generate\_all\_tables(integer, integer, integer, integer, integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.generate\_all\_tables(shops\_count integer, workers\_count integer, products\_count integer, customers\_count integer, realization\_count integer) RETURNS void

LANGUAGE plpgsql

AS $$

begin

perform generate\_statuses();

perform generate\_districts();

perform generate\_product\_types();

perform generate\_shops(shops\_count);

perform generate\_workers(workers\_count);

perform generate\_customers(customers\_count);

perform generate\_products(products\_count);

perform generate\_realization(realization\_count);

end;

$$;

ALTER FUNCTION public.generate\_all\_tables(shops\_count integer, workers\_count integer, products\_count integer, customers\_count integer, realization\_count integer) OWNER TO postgres;

--

-- TOC entry 239 (class 1255 OID 16787)

-- Name: generate\_customers(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.generate\_customers(count integer) RETURNS void

LANGUAGE plpgsql

AS $$

declare

first\_names varchar[];

second\_names varchar[];

prefixes varchar[];

streets varchar[];

letters varchar[];

begin

first\_names = array [ 'Августина', 'Аврора', 'Агафья', 'Агнесса', 'Агния', 'Ада', 'Аделия', 'Аза',

'Азиза', 'Аида', 'Алана', 'Алевтина', 'Александра', 'Алико', 'Алина', 'Алиса', 'Алла', 'Альберта', 'Альбина', 'Альжбета',

'Амелия', 'Амина', 'Анастасия', 'Ангелина', 'Анжела, Анжелика', 'Анисья', 'Анита', 'Анна', 'Антонина', 'Анфиса',

'Анэля', 'Ариадна', 'Арина', 'Архелия', 'Астра', 'Ася', 'Аурелия', 'Беатриса', 'Белла',

'Береслава', 'Берта', 'Биргит', 'Богдана', 'Божена', 'Борислава', 'Бронислава', 'Валентина', 'Валерия', 'Ванда',

'Варвара', 'Василиса', 'Венера', 'Вера', 'Вероника', 'Веселина', 'Веста', 'Вета, Иветта', 'Вида, Видана', 'Виктория',

'Вилора', 'Виолетта', 'Виргиния', 'Виталина', 'Владислава', 'Галина', 'Гаянэ', 'Гелена', 'Гелла', 'Генриетта',

'Георгина', 'Гера', 'Гертруда', 'Глафира', 'Глория', 'Гражина', 'Грета', 'Гюзель', 'Дайна', 'Дана', 'Даниэла', 'Данута',

'Дарина', 'Дарья', 'Дебора', 'Джемма', 'Джулия', 'Джульетта', 'Диана', 'Дина', 'Динара', 'Диодора', 'Дионисия', 'Доля',

'Доминика', 'Ева', 'Евгения', 'Евдокия', 'Екатерина', 'Елена', 'Елизавета', 'Жанна', 'Зара', 'Земфира',

'Зинаида', 'Злата', 'Зоя', 'Иветта', 'Ивона', 'Изабелла', 'Изольда', 'Илзе', 'Инара', 'Инга', 'Инесса', 'Инна',

'Иоанна', 'Иоланта', 'Ираида', 'Ирина', 'Ольга', 'Сабина', 'Санта', 'Сарра', 'Светлана', 'Северина', 'Серафима',

'Сильва, Сильвия', 'Сима', 'Симона', 'Снежана', 'Софья', 'Станислава', 'Стелла', 'Стефания', 'Сусанна', 'Таира', 'Таисия',

'Тала', 'Тамара', 'Татьяна', 'Тереза', 'Томила', 'Ульяна','Юлиана', 'Устина', 'Фаиза', 'Фаина', 'Фаня', 'Фая',

'Фелиция', 'Флора', 'Франсуаза', 'Фрида', 'Хильда', 'Христина', 'Христя', 'Цветана', 'Чеслава', 'Эдда',

'Эдита', 'Элеонора', 'Элина, Эллина', 'Элла', 'Эллада', 'Элоиза', 'Эльвира', 'Эльга', 'Эльза', 'Эльмира', 'Эмилия', 'Эмма',

'Эрика', 'Эсмеральда', 'Юзефа', 'Юлия', 'Юна', 'Юнона', 'Юстина', 'Ядвига', 'Яна', 'Янита', 'Янка', 'Ярослава',

'Абрам', 'Аваз', 'Август', 'Авдей', 'Автандил', 'Адам', 'Адис', 'Адольф', 'Адриан', 'Азарий', 'Аким', 'Алан',

'Александр', 'Алексей', 'Альберт', 'Альфред', 'Амадей', 'Амадеус', 'Амаяк', 'Анатолий', 'Ангел', 'Андоим', 'Андрей',

'Аникита', 'Антон', 'Ануфрий', 'Арам', 'Арий', 'Аристарх', 'Аркадий', 'Арно', 'Арнольд', 'Арон', 'Арсений',

'Артем', 'Артемий', 'Артур', 'Архип', 'Аскольд', 'Афанасий', 'Ахмет', 'Ашот', 'Бежен ', 'Бенедикт', 'Берек',

'Бернар', 'Богдан', 'Боголюб', 'Болеслав', 'Бонифаций', 'Борис', 'Борислав', 'Боян', 'Бронислав', 'Бруно', 'Вадим',

'Валентин', 'Валерий', 'Вальтер', 'Василий', 'Велизар', 'Венедикт', 'Вениамин', 'Виктор', 'Вилен', 'Вилли', 'Вильгельм',

'Виссарион', 'Виталий', 'Витаутас', 'Витольд', 'Владимир', 'Владислав', 'Владлен', 'Володар', 'Вольдемар', 'Всеволод',

'Вячеслав', 'Г', 'Гавриил', 'Гарри', 'Гастон', 'Геннадий', 'Генрих', 'Георгий', 'Геральд', 'Герасим', 'Герман', 'Глеб',

'Гордей', 'Гордон', 'Градимир', 'Григорий', 'Гурий', 'Давыд', 'Даниил', 'Демид', 'Демьян',

'Денис', 'Джордан', 'Дмитрий', 'Дональд', 'Донат', 'Донатос', 'Дорофей', 'Евгений', 'Евграф', 'Евдоким', 'Евстафий',

'Егор', 'Елизар', 'Елисей', 'Емельян', 'Ермолай', 'Ерофей', 'Ефим', 'Ефимий', 'Ефрем', 'Жан', 'Ждан', 'Жорж',

'Захар', 'Захария', 'Зигмунд', 'Зиновий', 'Ибрагим', 'Иван', 'Игнат', 'Игорь', 'Измаил', 'Израиль', 'Илиан',

'Илларион', 'Илья', 'Иннокентий', 'Ион', 'Ионос', 'Иосиф', 'Ираклий', 'Иржи', 'Исай','Казимир', 'Карен', 'Карл',

'Ким', 'Кирилл', 'Клавдий', 'Клемент', 'Клим', 'Клод', 'Кондрат', 'Конкордий', 'Константин', 'Кузьма', 'Лазарь',

'Лев', 'Леван', 'Леонард', 'Леонид', 'Леонтий', 'Леопольд', 'Лука', 'Любомир', 'Людвиг', 'Люсьен', 'Мадлен', 'Май',

'Макар', 'Максим', 'Максимилиан', 'Мануил', 'Марат', 'Мариан', 'Марк', 'Мартин', 'Матвей', 'Мераб', 'Мечеслав',

'Милан', 'Мирон', 'Мирослав', 'Михаил', 'Мичлов', 'Модест', 'Моисей', 'Мурат', 'Муслим', 'Назар', 'Назарий', 'Натан',

'Наум', 'Никита', 'Никифор', 'Николай', 'Никон', 'Нисон', 'Нифонт', 'Олан', 'Олег', 'Олесь', 'Онисим', 'Орест', 'Осип',

'Оскар', 'Павел', 'Парамон', 'Петр', 'Платон', 'Порфирий', 'Прохор','Равиль', 'Радий', 'Радомир', 'Раис',

'Раймонд', 'Ратмир', 'Рафаил', 'Рафик', 'Рашид', 'Рем', 'Ренольд', 'Ринат', 'Рифат', 'Ричард', 'Роберт', 'Родион',

'Ролан', 'Роман', 'Ростислав', 'Рубен', 'Рудольф', 'Руслан', 'Рустам', 'Савва', 'Савел', 'Самсон', 'Святослав',

'Севастьян', 'Северин', 'Семен', 'Серафим', 'Сергей', 'Сократ', 'Соломон', 'Спартак', 'Стакрат', 'Станислав', 'Степан',

'Стивен', 'Стоян', 'Таис', 'Талик', 'Тамаз', 'Тарас', 'Тельман', 'Теодор', 'Терентий', 'Тибор', 'Тигран', 'Тигрий',

'Тимофей', 'Тимур', 'Тит', 'Тихон', 'Трифон', 'Трофим', 'Ульманас', 'Устин', 'Ф', 'Фаддей', 'Федор', 'Феликс', 'Феодосий',

'Фидель', 'Филимон', 'Филипп', 'Флорентий', 'Фома', 'Франц', 'Фридрих', 'Харитон', 'Христиан', 'Христос', 'Христофор',

'Эдвард', 'Эдуард', 'Эльдар', 'Эмиль', 'Эммануил', 'Эраст', 'Эрик', 'Эрнест', 'Юлиан', 'Юрий', 'Юхим','Яким',

'Яков', 'Ян', 'Яромир', 'Ярослав', 'Ясон'];

second\_names = array ['Барлоу' ,'Баскервилл' ,'Батчелор','Бетелл' ,'Биддер','Бичем' ,'Блэр','Бойер' ,'Ботт','Боттерилл','Бродерик',

'Бромфилд','Брэгг','Брюстер','Бутби','Гатри','Грейвз','Грейнджер','Гудман','Даблдэй','Даунинг','Додвелл','Дэнсон','Калвер',

'Кантуэлл','Кларксон','Клауд','Клиленд','Клоуз','Кокрил','Коллингвуд','Копленд','Косгроув','Коупленд','Коутс','Коуэлл','Кросби',

'Кушинг','Кэмпион','Лампкин','Лейк','Листон','Лэнгтон','Макгуайр','Макдоннелл','Макдугал','Макдугалл','Маккей','Маккинни','Макклейн',

'Маккормик','Маккэффри','Маклахлан','Маклохлин','Макмаллен','Макмюррей','Макуильямс','Макфи','Макьюэн','Манселл','Мелтон','Мерриман',

'Мерритт','Мерчант','Милнз','Мэнселл','Несбит','Несбитт','Ньюбери','Ньюберри','Ньюболд','Ньюолл','Ньюэлл','Оллред','Орр','Ортон',

'Осборн','Осмонд','Рэтлифф','Сарджент','Сеймур','Симм','Спарроу','Спунер','Стэндинг','Стэнтон','Стэрди','Суинбёрн','Сэлмон',

'Сэмюэлс','Таббс','Талли','Тейт','Тернбулл','Тобин','Толмен','Торнхилл','Торп','Трэверс','Трэшер','Турнер','Тэкери','Тэннер','Фарлоу',

'Фейтфулл','Флек','Флетчер','Фрит','Фрэмптон','Шервуд','Эбби','Эгертон','Эдди','Эдкок','Эйтчисон','Эмерсон','Эшби','Эшкрофт'];

prefixes = array ['ул. ','пр-т. ','пер. '];

streets = array ['Гринкевича','Гурова','Ильича','Артема','Байдукова','Батова','Баренца','Ватутина','Зайцева','Ермошенко',

'Горбатова','Ионова','Калинина','Кобозева','Коваля','Конева','Куприна','Лагутенко','Маяковского','Орешкова','Петровского','Прокофьева',

'Ткаченко','Флеровского','Шекспира','Щорса'];

letters = array['','а','б','в','г','д'];

for i in 1..count

loop

insert into customers (first\_name, second\_name, id\_district, id\_status, work, birthday, phone)

values (first\_names[trunc(random() \* array\_length(first\_names, 1)) + 1],

second\_names[trunc(random() \* array\_length(second\_names, 1)) + 1],

(select id from districts limit 1 offset trunc(random()\*(select count(\*) from districts))),

(select id from statuses limit 1 offset trunc(random()\*(select count(\*) from statuses))),

prefixes[trunc(random()\*array\_length(prefixes,1))+1] || streets[trunc(random()\*array\_length(streets,1))+1] || ' ' ||

trunc(random() \* 200) || letters[trunc(random()\*array\_length(letters,1))+1],

'01-01-2002'::date - trunc(random() \* 29220)::integer,

'071' || trunc(random() \* (9999999-1000000)+1000000));

end loop;

end ;

$$;

ALTER FUNCTION public.generate\_customers(count integer) OWNER TO postgres;

--

-- TOC entry 243 (class 1255 OID 16804)

-- Name: generate\_districts(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.generate\_districts() RETURNS void

LANGUAGE plpgsql

AS $$

begin

insert into districts (district)

values ('Киевский'),('Буденовский'),('Петровский'),('Куйбышевский'),

('Ворошиловский'),('Кировский'),('Калининский');

end;

$$;

ALTER FUNCTION public.generate\_districts() OWNER TO postgres;

--

-- TOC entry 245 (class 1255 OID 16811)

-- Name: generate\_product\_types(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.generate\_product\_types() RETURNS void

LANGUAGE plpgsql

AS $$

begin

insert into product\_types (product\_type)

values ('посуда'),('одежда'),('обувь'),

('игрушки'),('товары для дома'),('товары для детей'),('инструменты'),

('компьютерная техника'),('электроника');

end;

$$;

ALTER FUNCTION public.generate\_product\_types() OWNER TO postgres;

--

-- TOC entry 269 (class 1255 OID 16820)

-- Name: generate\_products(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.generate\_products(count integer) RETURNS void

LANGUAGE plpgsql

AS $$

declare

product\_count integer;

customer\_id integer ;

part1 varchar[] = array ['Горячий', 'Прекрасный', 'Холодный', 'Крутой', 'Вонючий', 'Ароматный', 'Веселый', 'Грустный', 'Обаятельный',

'Прикольный', 'Одаренный', 'Романтичный', 'Волевой', 'Призрачный', 'Дырявый', 'Рваный',

'Железный', 'Деревянный', 'Золотой', 'Серебрянный', 'Позолоченный', 'Поддельный', 'Натуральный'];

part2 varchar[] = array ['сапог', 'диск', 'шлепок', 'утюг', 'компьютер', 'носок', 'галстук', 'робот', 'автомат',

'приклад', 'блокнот', 'шнур', 'монитор', 'механизм', 'эспандер', 'провод', 'удлинитель',

'переходник', 'аксессуар', 'штатив', 'микрофон', 'ремень', 'кроссовок', 'мяч', 'конструктор'];

-- Совершеннолетие пользователя (товар можно сдавать только с 18 лет)

customer\_majority date ;

begin

for i in 1..count

loop

product\_count = (random() ^ 5) \* 100 + 1;

customer\_id = (select id from customers limit 1 offset trunc(random() \* (select count(\*) from customers)));

customer\_majority = (select birthday from customers where id = customer\_id) + 6575;

insert into products(quittance, id\_product\_type, product\_name, reception, count, remaining, price,

id\_customer, id\_shop, id\_woker)

VALUES ((select nextval('quittance\_num')),

(select id

from product\_types

limit 1

offset

trunc(random() \* (select count(\*) from product\_types))),

part1[trunc(random() \* array\_length(part1, 1)) + 1] || ' ' ||

part2[trunc(random() \* array\_length(part2, 1)) + 1] ||

trunc(random() \* 95 + 1) ,

customer\_majority + trunc(random() \* ('10-05-2021'::date - customer\_majority))::integer,

product\_count,

product\_count,

random() \* 10000 + 100,

customer\_id,

(select id from shops limit 1 offset trunc(random() \* (select count(\*) from shops))),

(select id from workers limit 1 offset trunc(random() \* (select count(\*) from workers))));

end loop;

end ;

$$;

ALTER FUNCTION public.generate\_products(count integer) OWNER TO postgres;

--

-- TOC entry 266 (class 1255 OID 16847)

-- Name: generate\_realization(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.generate\_realization(count integer) RETURNS integer

LANGUAGE plpgsql

AS $$

declare

real\_count integer = 0;

random\_id\_product integer;

reception\_date date;

begin

for i in 1..count

loop

random\_id\_product =

(select id

from products

where remaining > 0

limit 1

offset

trunc(random() \* (select count(\*) from products where remaining > 0)));

exit when random\_id\_product is null;

reception\_date = (select reception from products where products.id = random\_id\_product);

insert into realization (ticket, id\_product, realization\_date, realization\_count, id\_worker)

VALUES (nextval('ticket\_num'),

random\_id\_product,

reception\_date + trunc(random() \* ('10-05-2021'::date - reception\_date))::integer,

(trunc(random() \* (select remaining from products where products.id = random\_id\_product)) + 1),

(select id from workers limit 1 offset trunc(random() \* (select count(\*) from workers))));

real\_count = real\_count + 1;

end loop;

return real\_count;

end;

$$;

ALTER FUNCTION public.generate\_realization(count integer) OWNER TO postgres;

--

-- TOC entry 246 (class 1255 OID 16817)

-- Name: generate\_shops(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.generate\_shops(count integer) RETURNS void

LANGUAGE plpgsql

AS $$

begin

for i in 1..count

loop

insert into shops(name) values ('Комиссионный №' || trunc(random()\*1000000));

end loop;

end;

$$;

ALTER FUNCTION public.generate\_shops(count integer) OWNER TO postgres;

--

-- TOC entry 244 (class 1255 OID 16806)

-- Name: generate\_statuses(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.generate\_statuses() RETURNS void

LANGUAGE plpgsql

AS $$

begin

insert into statuses (status)

values ('пенсионер'),('предприниматель'),('домохозяйка'),

('госслужащий'),('студент'),('безработный');

end;

$$;

ALTER FUNCTION public.generate\_statuses() OWNER TO postgres;

--

-- TOC entry 264 (class 1255 OID 16808)

-- Name: generate\_workers(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.generate\_workers(count integer) RETURNS void

LANGUAGE plpgsql

AS $$

begin

for i in 1..count

loop

perform create\_worker('worker' || i, 'qwe',

(select id from shops limit 1 offset trunc(random() \* (select count(\*) from shops))));

end loop;

end;

$$;

ALTER FUNCTION public.generate\_workers(count integer) OWNER TO postgres;

--

-- TOC entry 260 (class 1255 OID 25126)

-- Name: instead\_of\_insert(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.instead\_of\_insert() RETURNS trigger

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

insert into products (quittance, id\_product\_type, product\_name, reception, count, remaining, price, id\_customer,

id\_shop, id\_woker)

values (nextval('quittance\_num'), (select id from product\_types where product\_type = new.product\_type),

new.product\_name, now()::date, new.remaining, new.remaining, new.price,

new.id\_customer, (select workers.id\_shop from workers where login = session\_user),

(select id from workers where login = session\_user));

return new;

end;

$$;

ALTER FUNCTION public.instead\_of\_insert() OWNER TO postgres;

--

-- TOC entry 263 (class 1255 OID 25262)

-- Name: is\_available\_check(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.is\_available\_check() RETURNS trigger

LANGUAGE plpgsql

AS $$

begin

if old.is\_available = false then return old; else return new;

end if;

end;

$$;

ALTER FUNCTION public.is\_available\_check() OWNER TO postgres;

--

-- TOC entry 277 (class 1255 OID 17149)

-- Name: query\_10\_1(character varying, integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_10\_1(status1 character varying, prod\_count integer) RETURNS TABLE(customer character varying, shop character varying, status character varying, products\_count integer)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select t1.customer, t1.shop, t1.status, t1.products\_count

from (select (c.first\_name || ' ' || c.second\_name) :: varchar as customer,

s.name as shop,

s2.status,

count(p.quittance)::integer as products\_count

from products p

inner join customers c on p.id\_customer = c.id

inner join shops s on p.id\_shop = s.id

inner join statuses s2 on c.id\_status = s2.id

where s2.status = status1

group by p.id\_customer, customer, shop, s2.status

having count(p.quittance) > prod\_count

order by count(p.quittance) desc) as t1;

end;

$$;

ALTER FUNCTION public.query\_10\_1(status1 character varying, prod\_count integer) OWNER TO postgres;

--

-- TOC entry 270 (class 1255 OID 17055)

-- Name: query\_1\_1(character varying); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_1\_1(product\_t character varying) RETURNS TABLE(id integer, product\_name character varying, recerption date, remaining integer, price integer, customer\_firs\_name character varying, customer\_second\_name character varying, shop character varying, product\_type character varying)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select products.id,

products.product\_name,

products.reception,

products.remaining,

products.price,

c.first\_name,

c.second\_name,

s.name,

pt.product\_type

from products

inner join customers c on products.id\_customer = c.id

inner join product\_types pt on products.id\_product\_type = pt.id

inner join shops s on products.id\_shop = s.id

where id\_product\_type =

(select product\_types.id from product\_types where product\_types.product\_type = product\_t);

end;

$$;

ALTER FUNCTION public.query\_1\_1(product\_t character varying) OWNER TO postgres;

--

-- TOC entry 271 (class 1255 OID 17057)

-- Name: query\_1\_2(character varying); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_1\_2(district\_name character varying) RETURNS TABLE(id integer, first\_name character varying, second\_name character varying, district character varying, work character varying, birthday date, phone character)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select customers.id,

customers.first\_name,

customers.second\_name,

d.district,

customers.work,

customers.birthday,

customers.phone

from customers

inner join districts d on customers.id\_district = d.id

where id\_district = (select districts.id from districts where districts.district = district\_name);

end;

$$;

ALTER FUNCTION public.query\_1\_2(district\_name character varying) OWNER TO postgres;

--

-- TOC entry 272 (class 1255 OID 17059)

-- Name: query\_1\_3(date); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_1\_3(birth date) RETURNS TABLE(id integer, first\_name character varying, second\_name character varying, district character varying, work character varying, birthday date, phone character)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select customers.id,

customers.first\_name,

customers.second\_name,

d.district,

customers.work,

customers.birthday,

customers.phone

from customers

inner join districts d on customers.id\_district = d.id

where customers.birthday > birth;

end;

$$;

ALTER FUNCTION public.query\_1\_3(birth date) OWNER TO postgres;

--

-- TOC entry 274 (class 1255 OID 17060)

-- Name: query\_1\_4(date); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_1\_4(reception\_date date) RETURNS TABLE(id integer, product\_name character varying, recerption date, remaining integer, price integer, customer\_firs\_name character varying, customer\_second\_name character varying, shop character varying, product\_type character varying)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select products.id,

products.product\_name,

products.reception,

products.remaining,

products.price,

c.first\_name,

c.second\_name,

s.name,

pt.product\_type

from products

inner join customers c on products.id\_customer = c.id

inner join product\_types pt on products.id\_product\_type = pt.id

inner join shops s on products.id\_shop = s.id

where reception > reception\_date;

end;

$$;

ALTER FUNCTION public.query\_1\_4(reception\_date date) OWNER TO postgres;

--

-- TOC entry 275 (class 1255 OID 17152)

-- Name: query\_8\_1(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_8\_1(remaining\_count integer) RETURNS TABLE(product\_name character varying, quittance integer, remaining integer, tickets\_count integer, reception date, shop character varying)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select t1.product\_name, t1.quittance, t1.remaining, t1.tikets\_count, t1.reception, t1.shop

from (select p.product\_name,

p.quittance,

p.remaining,

count(r.ticket):: integer as tikets\_count,

p.reception,

s.name as shop

from realization r

inner join products p on r.id\_product = p.id

inner join shops s on p.id\_shop = s.id

where p.remaining > remaining\_count

group by r.id\_product, p.product\_name, p.quittance, p.remaining, p.reception, s.name

) as t1;

end;

$$;

ALTER FUNCTION public.query\_8\_1(remaining\_count integer) OWNER TO postgres;

--

-- TOC entry 276 (class 1255 OID 17151)

-- Name: query\_8\_2(character varying); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_8\_2(part\_of\_phone character varying) RETURNS TABLE(first\_name character varying, second\_name character varying, phone character, product\_count integer)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select t1.first\_name, t1.second\_name, t1.phone, t1.count

from (select c.first\_name, c.second\_name, c.phone, count(p.quittance) :: integer as count

from customers c

inner join products p on c.id = p.id\_customer

where c.phone similar to '%' || part\_of\_phone || '%'

group by c.phone, c.second\_name, c.first\_name

order by count desc) as t1;

end;

$$;

ALTER FUNCTION public.query\_8\_2(part\_of\_phone character varying) OWNER TO postgres;

--

-- TOC entry 273 (class 1255 OID 17146)

-- Name: query\_8\_3(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_8\_3(product\_id integer) RETURNS TABLE(product\_name character varying, quittance integer, shop character varying, realizations\_count integer)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select t1.product\_name, t1.quittance, t1.shop, t1.realizations\_count

from (select p.product\_name,

p.quittance,

s.name as shop,

count(r.realization\_count)::integer as realizations\_count

from realization r

right join products p on r.id\_product = p.id

inner join shops s on p.id\_shop = s.id

where r.id\_product = product\_id

group by p.id, s.name, p.quittance) as t1;

end;

$$;

ALTER FUNCTION public.query\_8\_3(product\_id integer) OWNER TO postgres;

--

-- TOC entry 278 (class 1255 OID 17150)

-- Name: query\_8\_4(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_8\_4(shop\_id integer) RETURNS TABLE(shop character varying, login integer)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select t1.name, count(t1.login) :: integer

from (select s.name, w.login

from shops s

right join workers w on s.id = w.id\_shop

where s.id = shop\_id) as t1

group by t1.name;

end;

$$;

ALTER FUNCTION public.query\_8\_4(shop\_id integer) OWNER TO postgres;

--

-- TOC entry 261 (class 1255 OID 17148)

-- Name: query\_9\_1(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.query\_9\_1(min\_sold\_count integer) RETURNS TABLE(product\_name character varying, shop character varying, sold\_count integer)

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

return query select t1.product\_name, t1.shop, t1.sold

from (select p.product\_name, s.name as shop, sum(r.realization\_count)::integer as sold

from realization r

inner join products p on r.id\_product = p.id

inner join shops s on p.id\_shop = s.id

group by p.id, p.product\_name, shop

having sum(r.realization\_count) > min\_sold\_count

order by shop, sold) as t1;

end;

$$;

ALTER FUNCTION public.query\_9\_1(min\_sold\_count integer) OWNER TO postgres;

--

-- TOC entry 247 (class 1255 OID 16987)

-- Name: regenerate\_all\_tebles(integer, integer, integer, integer, integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.regenerate\_all\_tebles(shops\_count integer, workers\_count integer, products\_count integer, customers\_count integer, realization\_count integer) RETURNS void

LANGUAGE plpgsql

AS $$

begin

perform delete\_from\_all\_tables();

perform generate\_all\_tables(shops\_count, workers\_count, products\_count,

customers\_count, realization\_count);

end;

$$;

ALTER FUNCTION public.regenerate\_all\_tebles(shops\_count integer, workers\_count integer, products\_count integer, customers\_count integer, realization\_count integer) OWNER TO postgres;

--

-- TOC entry 240 (class 1255 OID 16960)

-- Name: replace\_worker(text); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.replace\_worker(username text) RETURNS void

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

execute format('drop role %s', username);

delete from workers where login = username;

end;

$$;

ALTER FUNCTION public.replace\_worker(username text) OWNER TO postgres;

--

-- TOC entry 241 (class 1255 OID 25221)

-- Name: sell\_product(integer, integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.sell\_product(quittance\_ integer, count\_ integer) RETURNS void

LANGUAGE plpgsql SECURITY DEFINER

AS $$

begin

insert into realization (ticket, id\_product, realization\_date, realization\_count, id\_worker)

VALUES (nextval('ticket\_num'),

quittance\_,

now()::date,

count\_,

(select id from workers where login = session\_user));

end;

$$;

ALTER FUNCTION public.sell\_product(quittance\_ integer, count\_ integer) OWNER TO postgres;

SET default\_tablespace = '';

SET default\_table\_access\_method = heap;

--

-- TOC entry 200 (class 1259 OID 16627)

-- Name: customers; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.customers (

id integer NOT NULL,

first\_name character varying(30) NOT NULL,

second\_name character varying(30) NOT NULL,

id\_district integer NOT NULL,

id\_status integer NOT NULL,

work character varying(60),

birthday date NOT NULL,

phone character(10) NOT NULL,

CONSTRAINT years\_18 CHECK ((age((CURRENT\_DATE)::timestamp with time zone, (birthday)::timestamp with time zone) >= '18 years'::interval))

);

ALTER TABLE public.customers OWNER TO postgres;

--

-- TOC entry 201 (class 1259 OID 16631)

-- Name: customers\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.customers\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.customers\_id\_seq OWNER TO postgres;

--

-- TOC entry 3240 (class 0 OID 0)

-- Dependencies: 201

-- Name: customers\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.customers\_id\_seq OWNED BY public.customers.id;

--

-- TOC entry 202 (class 1259 OID 16633)

-- Name: districts; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.districts (

id integer NOT NULL,

district character varying(30)

);

ALTER TABLE public.districts OWNER TO postgres;

--

-- TOC entry 212 (class 1259 OID 16659)

-- Name: statuses; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.statuses (

id integer NOT NULL,

status character varying(30)

);

ALTER TABLE public.statuses OWNER TO postgres;

--

-- TOC entry 231 (class 1259 OID 25189)

-- Name: customers\_view; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.customers\_view AS

SELECT c.id,

c.first\_name,

c.second\_name,

d.district,

s.status,

c.work,

c.birthday,

c.phone

FROM ((public.customers c

JOIN public.statuses s ON ((c.id\_status = s.id)))

JOIN public.districts d ON ((c.id\_district = d.id)));

ALTER TABLE public.customers\_view OWNER TO postgres;

--

-- TOC entry 203 (class 1259 OID 16636)

-- Name: districts\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.districts\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.districts\_id\_seq OWNER TO postgres;

--

-- TOC entry 3245 (class 0 OID 0)

-- Dependencies: 203

-- Name: districts\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.districts\_id\_seq OWNED BY public.districts.id;

--

-- TOC entry 204 (class 1259 OID 16638)

-- Name: product\_types; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.product\_types (

id integer NOT NULL,

product\_type character varying(30)

);

ALTER TABLE public.product\_types OWNER TO postgres;

--

-- TOC entry 205 (class 1259 OID 16641)

-- Name: product\_types\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.product\_types\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.product\_types\_id\_seq OWNER TO postgres;

--

-- TOC entry 3248 (class 0 OID 0)

-- Dependencies: 205

-- Name: product\_types\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.product\_types\_id\_seq OWNED BY public.product\_types.id;

--

-- TOC entry 206 (class 1259 OID 16643)

-- Name: products; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.products (

id integer NOT NULL,

quittance integer,

id\_product\_type integer,

product\_name character varying(30),

reception date,

count integer,

remaining integer,

price integer,

id\_customer integer,

id\_shop integer,

id\_woker integer,

CONSTRAINT remaining\_less\_or\_equal\_count CHECK ((remaining <= count))

);

ALTER TABLE public.products OWNER TO postgres;

--

-- TOC entry 207 (class 1259 OID 16647)

-- Name: products\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.products\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.products\_id\_seq OWNER TO postgres;

--

-- TOC entry 3251 (class 0 OID 0)

-- Dependencies: 207

-- Name: products\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.products\_id\_seq OWNED BY public.products.id;

--

-- TOC entry 210 (class 1259 OID 16654)

-- Name: shops; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.shops (

id integer NOT NULL,

name character varying(30) NOT NULL

);

ALTER TABLE public.shops OWNER TO postgres;

--

-- TOC entry 232 (class 1259 OID 25214)

-- Name: products\_modify\_view; Type: VIEW; Schema: public; Owner: testworker

--

CREATE VIEW public.products\_modify\_view WITH (security\_barrier='true') AS

SELECT p.quittance,

pt.product\_type,

p.product\_name,

p.reception,

p.remaining,

p.price,

p.id\_customer,

c.first\_name,

c.second\_name,

c.phone,

s.name AS shop

FROM (((public.products p

JOIN public.product\_types pt ON ((p.id\_product\_type = pt.id)))

JOIN public.customers c ON ((p.id\_customer = c.id)))

JOIN public.shops s ON ((p.id\_shop = s.id)))

ORDER BY p.reception DESC, p.quittance DESC;

ALTER TABLE public.products\_modify\_view OWNER TO testworker;

--

-- TOC entry 214 (class 1259 OID 16664)

-- Name: workers; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.workers (

id integer NOT NULL,

login character varying(30) NOT NULL,

id\_shop integer NOT NULL,

is\_available boolean DEFAULT true

);

ALTER TABLE public.workers OWNER TO postgres;

--

-- TOC entry 234 (class 1259 OID 25236)

-- Name: products\_view; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.products\_view AS

SELECT p.quittance,

pt.product\_type,

p.product\_name,

p.reception,

p.count,

p.remaining,

p.price,

c.first\_name,

c.second\_name,

s.name,

w.login

FROM ((((public.products p

JOIN public.customers c ON ((p.id\_customer = c.id)))

JOIN public.shops s ON ((p.id\_shop = s.id)))

JOIN public.workers w ON ((p.id\_woker = w.id)))

JOIN public.product\_types pt ON ((p.id\_product\_type = pt.id)));

ALTER TABLE public.products\_view OWNER TO postgres;

--

-- TOC entry 233 (class 1259 OID 25222)

-- Name: query\_11\_1; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_11\_1 AS

SELECT (( SELECT avg(products.price) AS avg

FROM public.products

WHERE (products.id\_shop = s.id)))::integer AS average\_price,

s.name AS shop

FROM public.shops s

ORDER BY ((( SELECT avg(products.price) AS avg

FROM public.products

WHERE (products.id\_shop = s.id)))::integer) DESC;

ALTER TABLE public.query\_11\_1 OWNER TO postgres;

--

-- TOC entry 226 (class 1259 OID 17200)

-- Name: query\_12\_1; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_12\_1 AS

SELECT products.id,

products.quittance,

products.id\_product\_type,

products.product\_name,

products.reception,

products.count,

products.remaining,

products.price,

products.id\_customer,

products.id\_shop,

products.id\_woker

FROM public.products

WHERE (products.count = products.remaining)

UNION ALL

SELECT products.id,

products.quittance,

products.id\_product\_type,

products.product\_name,

products.reception,

products.count,

products.remaining,

products.price,

products.id\_customer,

products.id\_shop,

products.id\_woker

FROM public.products

WHERE (products.remaining = 0)

ORDER BY 7 DESC;

ALTER TABLE public.query\_12\_1 OWNER TO postgres;

--

-- TOC entry 227 (class 1259 OID 17205)

-- Name: query\_13\_1; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_13\_1 AS

SELECT customers.first\_name,

customers.second\_name,

customers.phone

FROM public.customers

WHERE (customers.id IN ( SELECT products.id\_customer

FROM public.products

WHERE (products.remaining > 0)));

ALTER TABLE public.query\_13\_1 OWNER TO postgres;

--

-- TOC entry 228 (class 1259 OID 17214)

-- Name: query\_13\_2; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_13\_2 AS

SELECT customers.first\_name,

customers.second\_name,

customers.phone

FROM public.customers

WHERE (NOT (customers.id IN ( SELECT products.id\_customer

FROM public.products

WHERE (products.remaining > 0))));

ALTER TABLE public.query\_13\_2 OWNER TO postgres;

--

-- TOC entry 229 (class 1259 OID 17249)

-- Name: query\_13\_3; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_13\_3 AS

SELECT products.quittance,

products.product\_name,

products.price,

products.remaining,

products.count,

CASE

WHEN (((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer = 100) THEN (('Не покупается ('::text || ((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer) || '%)'::text)

WHEN ((((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer >= 80) AND (((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer <= 99)) THEN (('Очень плохо покупается ('::text || ((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer) || '%)'::text)

WHEN ((((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer >= 60) AND (((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer <= 80)) THEN (('Плохо покупается ('::text || ((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer) || '%)'::text)

WHEN ((((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer >= 40) AND (((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer <= 60)) THEN (('Покупается ('::text || ((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer) || '%)'::text)

WHEN ((((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer >= 20) AND (((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer <= 40)) THEN (('Хорошо покупается ('::text || ((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer) || '%)'::text)

WHEN ((((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer >= 1) AND (((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer <= 20)) THEN (('Очень хорошо покупается ('::text || ((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer) || '%)'::text)

WHEN (((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer = 0) THEN (('Раскупили ('::text || ((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer) || '%)'::text)

ELSE (((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer || ' error'::text)

END AS state

FROM public.products

ORDER BY (((((products.remaining)::double precision / (products.count)::double precision) \* (100)::double precision))::integer), products.count DESC;

ALTER TABLE public.query\_13\_3 OWNER TO postgres;

--

-- TOC entry 230 (class 1259 OID 17262)

-- Name: query\_13\_4; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_13\_4 AS

SELECT products.quittance,

products.product\_name,

products.price,

s.name AS shop

FROM (public.products

JOIN public.shops s ON ((products.id\_shop = s.id)))

WHERE ((products.price)::numeric > ( SELECT avg(products\_1.price) AS avg

FROM public.products products\_1))

ORDER BY s.name, products.price;

ALTER TABLE public.query\_13\_4 OWNER TO postgres;

--

-- TOC entry 220 (class 1259 OID 17094)

-- Name: query\_2\_1; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_2\_1 AS

SELECT (((customers.first\_name)::text || ' '::text) || (customers.second\_name)::text) AS customer\_name,

customers.work,

p.quittance,

p.remaining,

s.name

FROM ((public.customers

JOIN public.products p ON ((customers.id = p.id\_customer)))

JOIN public.shops s ON ((p.id\_shop = s.id)))

ORDER BY (((customers.first\_name)::text || ' '::text) || (customers.second\_name)::text);

ALTER TABLE public.query\_2\_1 OWNER TO postgres;

--

-- TOC entry 208 (class 1259 OID 16649)

-- Name: realization; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.realization (

id integer NOT NULL,

ticket integer,

id\_product integer,

realization\_date date,

realization\_count integer,

id\_worker integer

);

ALTER TABLE public.realization OWNER TO postgres;

--

-- TOC entry 218 (class 1259 OID 17078)

-- Name: query\_2\_2; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_2\_2 AS

SELECT products.quittance,

products.product\_name,

products.reception,

r.realization\_count,

r.realization\_date,

s.name AS shop

FROM ((public.products

JOIN public.realization r ON ((products.id = r.id\_product)))

JOIN public.shops s ON ((products.id\_shop = s.id)))

ORDER BY s.name, products.reception, r.realization\_date, r.realization\_count;

ALTER TABLE public.query\_2\_2 OWNER TO postgres;

--

-- TOC entry 219 (class 1259 OID 17086)

-- Name: query\_2\_3; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_2\_3 AS

SELECT realization.ticket,

realization.realization\_date,

realization.realization\_count,

w.login,

s.name

FROM (((public.realization

JOIN public.products p ON ((realization.id\_product = p.id)))

JOIN public.shops s ON ((p.id\_shop = s.id)))

JOIN public.workers w ON ((p.id\_woker = w.id)))

ORDER BY s.name, realization.realization\_date, w.login;

ALTER TABLE public.query\_2\_3 OWNER TO postgres;

--

-- TOC entry 221 (class 1259 OID 17108)

-- Name: query\_3\_1; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_3\_1 AS

SELECT products.quittance,

products.count,

products.remaining,

r.ticket

FROM (public.products

LEFT JOIN public.realization r ON ((products.id = r.id\_product)))

ORDER BY products.quittance;

ALTER TABLE public.query\_3\_1 OWNER TO postgres;

--

-- TOC entry 222 (class 1259 OID 17112)

-- Name: query\_4\_1; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_4\_1 AS

SELECT r.ticket,

r.realization\_date,

r.realization\_count,

workers.login

FROM (public.workers

RIGHT JOIN public.realization r ON ((workers.id = r.id\_worker)));

ALTER TABLE public.query\_4\_1 OWNER TO postgres;

--

-- TOC entry 223 (class 1259 OID 17116)

-- Name: query\_5\_1; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_5\_1 AS

SELECT products.quittance,

products.product\_name,

products.reception,

( SELECT workers.login

FROM public.workers

WHERE (workers.id = products.id\_woker)) AS login

FROM public.products;

ALTER TABLE public.query\_5\_1 OWNER TO postgres;

--

-- TOC entry 224 (class 1259 OID 17127)

-- Name: query\_6\_1; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_6\_1 AS

SELECT sold\_products.quittance,

sold\_products.count,

sold\_products.reception,

s.name

FROM (( SELECT products.id,

products.quittance,

products.id\_product\_type,

products.product\_name,

products.reception,

products.count,

products.remaining,

products.price,

products.id\_customer,

products.id\_shop,

products.id\_woker

FROM public.products

WHERE (products.remaining = 0)) sold\_products

JOIN public.shops s ON ((sold\_products.id\_shop = s.id)))

ORDER BY s.name, sold\_products.reception;

ALTER TABLE public.query\_6\_1 OWNER TO postgres;

--

-- TOC entry 225 (class 1259 OID 17133)

-- Name: query\_7\_1; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.query\_7\_1 AS

SELECT count(\*) AS count,

( SELECT count(\*) AS sold

FROM public.products products\_1

WHERE (products\_1.remaining = 0)) AS sold

FROM public.products;

ALTER TABLE public.query\_7\_1 OWNER TO postgres;

--

-- TOC entry 216 (class 1259 OID 16818)

-- Name: quittance\_num; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.quittance\_num

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.quittance\_num OWNER TO postgres;

--

-- TOC entry 209 (class 1259 OID 16652)

-- Name: realization\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.realization\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.realization\_id\_seq OWNER TO postgres;

--

-- TOC entry 3273 (class 0 OID 0)

-- Dependencies: 209

-- Name: realization\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.realization\_id\_seq OWNED BY public.realization.id;

--

-- TOC entry 235 (class 1259 OID 25241)

-- Name: realization\_view; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.realization\_view AS

SELECT r.ticket,

p.product\_name,

r.realization\_date,

r.realization\_count,

w.login

FROM ((public.realization r

JOIN public.products p ON ((r.id\_product = p.id)))

JOIN public.workers w ON ((r.id\_worker = w.id)))

ORDER BY r.id\_product;

ALTER TABLE public.realization\_view OWNER TO postgres;

--

-- TOC entry 211 (class 1259 OID 16657)

-- Name: shops\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.shops\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.shops\_id\_seq OWNER TO postgres;

--

-- TOC entry 3276 (class 0 OID 0)

-- Dependencies: 211

-- Name: shops\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.shops\_id\_seq OWNED BY public.shops.id;

--

-- TOC entry 213 (class 1259 OID 16662)

-- Name: statuses\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.statuses\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.statuses\_id\_seq OWNER TO postgres;

--

-- TOC entry 3278 (class 0 OID 0)

-- Dependencies: 213

-- Name: statuses\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.statuses\_id\_seq OWNED BY public.statuses.id;

--

-- TOC entry 237 (class 1259 OID 25250)

-- Name: test\_view; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.test\_view AS

SELECT pt.product\_type,

sum(r.realization\_count) AS sum

FROM ((public.realization r

JOIN public.products p ON ((r.id\_product = p.id)))

JOIN public.product\_types pt ON ((p.id\_product\_type = pt.id)))

GROUP BY pt.product\_type;

ALTER TABLE public.test\_view OWNER TO postgres;

--

-- TOC entry 217 (class 1259 OID 16843)

-- Name: ticket\_num; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.ticket\_num

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.ticket\_num OWNER TO postgres;

--

-- TOC entry 215 (class 1259 OID 16667)

-- Name: workers\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.workers\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public.workers\_id\_seq OWNER TO postgres;

--

-- TOC entry 3282 (class 0 OID 0)

-- Dependencies: 215

-- Name: workers\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.workers\_id\_seq OWNED BY public.workers.id;

--

-- TOC entry 236 (class 1259 OID 25246)

-- Name: workers\_view; Type: VIEW; Schema: public; Owner: postgres

--

CREATE VIEW public.workers\_view AS

SELECT w.login,

s.name,

w.is\_available

FROM (public.workers w

JOIN public.shops s ON ((w.id\_shop = s.id)))

ORDER BY w.id DESC;

ALTER TABLE public.workers\_view OWNER TO postgres;

--

-- TOC entry 3014 (class 2604 OID 16669)

-- Name: customers id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.customers ALTER COLUMN id SET DEFAULT nextval('public.customers\_id\_seq'::regclass);

--

-- TOC entry 3016 (class 2604 OID 16670)

-- Name: districts id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.districts ALTER COLUMN id SET DEFAULT nextval('public.districts\_id\_seq'::regclass);

--

-- TOC entry 3017 (class 2604 OID 16671)

-- Name: product\_types id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.product\_types ALTER COLUMN id SET DEFAULT nextval('public.product\_types\_id\_seq'::regclass);

--

-- TOC entry 3018 (class 2604 OID 16672)

-- Name: products id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.products ALTER COLUMN id SET DEFAULT nextval('public.products\_id\_seq'::regclass);

--

-- TOC entry 3020 (class 2604 OID 16673)

-- Name: realization id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.realization ALTER COLUMN id SET DEFAULT nextval('public.realization\_id\_seq'::regclass);

--

-- TOC entry 3021 (class 2604 OID 16674)

-- Name: shops id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.shops ALTER COLUMN id SET DEFAULT nextval('public.shops\_id\_seq'::regclass);

--

-- TOC entry 3022 (class 2604 OID 16675)

-- Name: statuses id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.statuses ALTER COLUMN id SET DEFAULT nextval('public.statuses\_id\_seq'::regclass);

--

-- TOC entry 3023 (class 2604 OID 16676)

-- Name: workers id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.workers ALTER COLUMN id SET DEFAULT nextval('public.workers\_id\_seq'::regclass);

--

-- TOC entry 3216 (class 0 OID 16627)

-- Dependencies: 200

-- Data for Name: customers; Type: TABLE DATA; Schema: public; Owner: postgres

--

--

-- TOC entry 3026 (class 2606 OID 16678)

-- Name: customers customers\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.customers

ADD CONSTRAINT customers\_pkey PRIMARY KEY (id);

--

-- TOC entry 3028 (class 2606 OID 16680)

-- Name: districts districts\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.districts

ADD CONSTRAINT districts\_pkey PRIMARY KEY (id);

--

-- TOC entry 3030 (class 2606 OID 16682)

-- Name: product\_types product\_types\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.product\_types

ADD CONSTRAINT product\_types\_pkey PRIMARY KEY (id);

--

-- TOC entry 3032 (class 2606 OID 16684)

-- Name: products products\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.products

ADD CONSTRAINT products\_pkey PRIMARY KEY (id);

--

-- TOC entry 3034 (class 2606 OID 16686)

-- Name: realization realization\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.realization

ADD CONSTRAINT realization\_pkey PRIMARY KEY (id);

--

-- TOC entry 3036 (class 2606 OID 16688)

-- Name: shops shops\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.shops

ADD CONSTRAINT shops\_pkey PRIMARY KEY (id);

--

-- TOC entry 3038 (class 2606 OID 16690)

-- Name: statuses statuses\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.statuses

ADD CONSTRAINT statuses\_pkey PRIMARY KEY (id);

--

-- TOC entry 3042 (class 2606 OID 16692)

-- Name: workers workers\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.workers

ADD CONSTRAINT workers\_pkey PRIMARY KEY (id);

--

-- TOC entry 3039 (class 1259 OID 25255)

-- Name: workers\_login\_idx; Type: INDEX; Schema: public; Owner: postgres

--

CREATE UNIQUE INDEX workers\_login\_idx ON public.workers USING btree (lower((login)::text));

--

-- TOC entry 3040 (class 1259 OID 25188)

-- Name: workers\_login\_uindex; Type: INDEX; Schema: public; Owner: postgres

--

CREATE UNIQUE INDEX workers\_login\_uindex ON public.workers USING btree (login);

--

-- TOC entry 3052 (class 2620 OID 25260)

-- Name: products product\_count\_check; Type: TRIGGER; Schema: public; Owner: postgres

--

CREATE TRIGGER product\_count\_check BEFORE UPDATE ON public.products FOR EACH ROW EXECUTE FUNCTION public.count\_check();

--

-- TOC entry 3053 (class 2620 OID 16886)

-- Name: realization product\_count\_decrease\_on\_realization\_before\_insert; Type: TRIGGER; Schema: public; Owner: postgres

--

CREATE TRIGGER product\_count\_decrease\_on\_realization\_before\_insert BEFORE INSERT ON public.realization FOR EACH ROW EXECUTE FUNCTION public.decrease\_products();

--

-- TOC entry 3055 (class 2620 OID 25219)

-- Name: products\_modify\_view products\_modify\_view\_instead\_of\_insert; Type: TRIGGER; Schema: public; Owner: testworker

--

CREATE TRIGGER products\_modify\_view\_instead\_of\_insert INSTEAD OF INSERT ON public.products\_modify\_view FOR EACH ROW EXECUTE FUNCTION public.instead\_of\_insert();

--

-- TOC entry 3054 (class 2620 OID 25263)

-- Name: workers workers\_is\_available\_before\_update; Type: TRIGGER; Schema: public; Owner: postgres

--

CREATE TRIGGER workers\_is\_available\_before\_update BEFORE UPDATE ON public.workers FOR EACH ROW EXECUTE FUNCTION public.is\_available\_check();

--

-- TOC entry 3043 (class 2606 OID 16693)

-- Name: customers customers\_id\_district\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.customers

ADD CONSTRAINT customers\_id\_district\_fkey FOREIGN KEY (id\_district) REFERENCES public.districts(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3044 (class 2606 OID 16698)

-- Name: customers customers\_id\_status\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.customers

ADD CONSTRAINT customers\_id\_status\_fkey FOREIGN KEY (id\_status) REFERENCES public.statuses(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3045 (class 2606 OID 16703)

-- Name: products products\_id\_customer\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.products

ADD CONSTRAINT products\_id\_customer\_fkey FOREIGN KEY (id\_customer) REFERENCES public.customers(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3046 (class 2606 OID 16708)

-- Name: products products\_id\_product\_type\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.products

ADD CONSTRAINT products\_id\_product\_type\_fkey FOREIGN KEY (id\_product\_type) REFERENCES public.product\_types(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3047 (class 2606 OID 16713)

-- Name: products products\_id\_shop\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.products

ADD CONSTRAINT products\_id\_shop\_fkey FOREIGN KEY (id\_shop) REFERENCES public.shops(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3048 (class 2606 OID 16718)

-- Name: products products\_id\_woker\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.products

ADD CONSTRAINT products\_id\_woker\_fkey FOREIGN KEY (id\_woker) REFERENCES public.workers(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3049 (class 2606 OID 16723)

-- Name: realization realization\_id\_product\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.realization

ADD CONSTRAINT realization\_id\_product\_fkey FOREIGN KEY (id\_product) REFERENCES public.products(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3050 (class 2606 OID 16728)

-- Name: realization realization\_id\_worker\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.realization

ADD CONSTRAINT realization\_id\_worker\_fkey FOREIGN KEY (id\_worker) REFERENCES public.workers(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3051 (class 2606 OID 16812)

-- Name: workers workers\_shops\_id\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.workers

ADD CONSTRAINT workers\_shops\_id\_fk FOREIGN KEY (id\_shop) REFERENCES public.shops(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3215 (class 3256 OID 25201)

-- Name: workers administrator\_realization\_policy; Type: POLICY; Schema: public; Owner: postgres

--

CREATE POLICY administrator\_realization\_policy ON public.workers TO administrator USING (true);

--

-- TOC entry 3206 (class 0 OID 16643)

-- Dependencies: 206

-- Name: products; Type: ROW SECURITY; Schema: public; Owner: postgres

--

ALTER TABLE public.products ENABLE ROW LEVEL SECURITY;

--

-- TOC entry 3208 (class 0 OID 16649)

-- Dependencies: 208

-- Name: realization; Type: ROW SECURITY; Schema: public; Owner: postgres

--

ALTER TABLE public.realization ENABLE ROW LEVEL SECURITY;

--

-- TOC entry 3209 (class 3256 OID 25196)

-- Name: realization realization\_shop\_policy; Type: POLICY; Schema: public; Owner: postgres

--

CREATE POLICY realization\_shop\_policy ON public.realization TO worker USING ((EXISTS ( SELECT products.id

FROM public.products

WHERE ((products.id = realization.id\_product) AND (products.id\_shop = ( SELECT workers.id\_shop

FROM public.workers))))));

--

-- TOC entry 3211 (class 3256 OID 25168)

-- Name: shops shop\_id\_policy; Type: POLICY; Schema: public; Owner: postgres

--

CREATE POLICY shop\_id\_policy ON public.shops FOR SELECT TO worker USING ((id = ( SELECT workers.id\_shop

FROM public.workers

WHERE ((workers.login)::text = CURRENT\_USER))));

--

-- TOC entry 3212 (class 3256 OID 25230)

-- Name: shops shop\_to\_admin\_policy; Type: POLICY; Schema: public; Owner: postgres

--

CREATE POLICY shop\_to\_admin\_policy ON public.shops FOR SELECT TO administrator USING (true);

--

-- TOC entry 3210 (class 0 OID 16654)

-- Dependencies: 210

-- Name: shops; Type: ROW SECURITY; Schema: public; Owner: postgres

--

ALTER TABLE public.shops ENABLE ROW LEVEL SECURITY;

--

-- TOC entry 3213 (class 0 OID 16664)

-- Dependencies: 214

-- Name: workers; Type: ROW SECURITY; Schema: public; Owner: postgres

--

ALTER TABLE public.workers ENABLE ROW LEVEL SECURITY;

--

-- TOC entry 3214 (class 3256 OID 16985)

-- Name: workers workers\_id\_policy; Type: POLICY; Schema: public; Owner: postgres

--

CREATE POLICY workers\_id\_policy ON public.workers FOR SELECT TO worker USING ((CURRENT\_USER = (login)::text));

--

-- TOC entry 3207 (class 3256 OID 16977)

-- Name: products workers\_shop\_policy; Type: POLICY; Schema: public; Owner: postgres

--

CREATE POLICY workers\_shop\_policy ON public.products TO worker USING ((id\_shop = ( SELECT workers.id\_shop

FROM public.workers

WHERE ((workers.login)::text = CURRENT\_USER))));

--

-- TOC entry 3239 (class 0 OID 0)

-- Dependencies: 200

-- Name: TABLE customers; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT,INSERT ON TABLE public.customers TO worker;

GRANT SELECT ON TABLE public.customers TO holder;

--

-- TOC entry 3241 (class 0 OID 0)

-- Dependencies: 201

-- Name: SEQUENCE customers\_id\_seq; Type: ACL; Schema: public; Owner: postgres

--

GRANT USAGE ON SEQUENCE public.customers\_id\_seq TO worker;

--

-- TOC entry 3242 (class 0 OID 0)

-- Dependencies: 202

-- Name: TABLE districts; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.districts TO administrator;

GRANT SELECT ON TABLE public.districts TO holder;

GRANT SELECT ON TABLE public.districts TO worker;

--

-- TOC entry 3243 (class 0 OID 0)

-- Dependencies: 212

-- Name: TABLE statuses; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.statuses TO administrator;

GRANT SELECT ON TABLE public.statuses TO holder;

GRANT SELECT ON TABLE public.statuses TO worker;

--

-- TOC entry 3244 (class 0 OID 0)

-- Dependencies: 231

-- Name: TABLE customers\_view; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.customers\_view TO worker;

GRANT SELECT ON TABLE public.customers\_view TO holder;

--

-- TOC entry 3246 (class 0 OID 0)

-- Dependencies: 203

-- Name: SEQUENCE districts\_id\_seq; Type: ACL; Schema: public; Owner: postgres

--

GRANT USAGE ON SEQUENCE public.districts\_id\_seq TO administrator;

--

-- TOC entry 3247 (class 0 OID 0)

-- Dependencies: 204

-- Name: TABLE product\_types; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.product\_types TO administrator;

GRANT SELECT ON TABLE public.product\_types TO holder;

GRANT SELECT ON TABLE public.product\_types TO worker;

--

-- TOC entry 3249 (class 0 OID 0)

-- Dependencies: 205

-- Name: SEQUENCE product\_types\_id\_seq; Type: ACL; Schema: public; Owner: postgres

--

GRANT USAGE ON SEQUENCE public.product\_types\_id\_seq TO administrator;

--

-- TOC entry 3250 (class 0 OID 0)

-- Dependencies: 206

-- Name: TABLE products; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT,INSERT ON TABLE public.products TO worker;

GRANT SELECT ON TABLE public.products TO holder;

--

-- TOC entry 3252 (class 0 OID 0)

-- Dependencies: 207

-- Name: SEQUENCE products\_id\_seq; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT,USAGE ON SEQUENCE public.products\_id\_seq TO worker;

--

-- TOC entry 3253 (class 0 OID 0)

-- Dependencies: 210

-- Name: TABLE shops; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.shops TO holder;

GRANT SELECT ON TABLE public.shops TO worker;

GRANT SELECT ON TABLE public.shops TO administrator;

--

-- TOC entry 3254 (class 0 OID 0)

-- Dependencies: 232

-- Name: TABLE products\_modify\_view; Type: ACL; Schema: public; Owner: testworker

--

GRANT SELECT ON TABLE public.products\_modify\_view TO holder;

GRANT SELECT ON TABLE public.products\_modify\_view TO worker;

--

-- TOC entry 3255 (class 0 OID 0)

-- Dependencies: 214

-- Name: TABLE workers; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.workers TO administrator;

GRANT SELECT ON TABLE public.workers TO holder;

GRANT SELECT ON TABLE public.workers TO worker;

--

-- TOC entry 3256 (class 0 OID 0)

-- Dependencies: 234

-- Name: TABLE products\_view; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.products\_view TO holder;

--

-- TOC entry 3257 (class 0 OID 0)

-- Dependencies: 233

-- Name: TABLE query\_11\_1; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_11\_1 TO holder;

--

-- TOC entry 3258 (class 0 OID 0)

-- Dependencies: 226

-- Name: TABLE query\_12\_1; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_12\_1 TO holder;

--

-- TOC entry 3259 (class 0 OID 0)

-- Dependencies: 227

-- Name: TABLE query\_13\_1; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_13\_1 TO holder;

--

-- TOC entry 3260 (class 0 OID 0)

-- Dependencies: 228

-- Name: TABLE query\_13\_2; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_13\_2 TO holder;

--

-- TOC entry 3261 (class 0 OID 0)

-- Dependencies: 229

-- Name: TABLE query\_13\_3; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_13\_3 TO holder;

--

-- TOC entry 3262 (class 0 OID 0)

-- Dependencies: 230

-- Name: TABLE query\_13\_4; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_13\_4 TO holder;

--

-- TOC entry 3263 (class 0 OID 0)

-- Dependencies: 220

-- Name: TABLE query\_2\_1; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_2\_1 TO holder;

--

-- TOC entry 3264 (class 0 OID 0)

-- Dependencies: 208

-- Name: TABLE realization; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT,INSERT ON TABLE public.realization TO worker;

GRANT SELECT ON TABLE public.realization TO holder;

--

-- TOC entry 3265 (class 0 OID 0)

-- Dependencies: 218

-- Name: TABLE query\_2\_2; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_2\_2 TO holder;

--

-- TOC entry 3266 (class 0 OID 0)

-- Dependencies: 219

-- Name: TABLE query\_2\_3; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_2\_3 TO holder;

--

-- TOC entry 3267 (class 0 OID 0)

-- Dependencies: 221

-- Name: TABLE query\_3\_1; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_3\_1 TO holder;

--

-- TOC entry 3268 (class 0 OID 0)

-- Dependencies: 222

-- Name: TABLE query\_4\_1; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_4\_1 TO holder;

--

-- TOC entry 3269 (class 0 OID 0)

-- Dependencies: 223

-- Name: TABLE query\_5\_1; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_5\_1 TO holder;

--

-- TOC entry 3270 (class 0 OID 0)

-- Dependencies: 224

-- Name: TABLE query\_6\_1; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_6\_1 TO holder;

--

-- TOC entry 3271 (class 0 OID 0)

-- Dependencies: 225

-- Name: TABLE query\_7\_1; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.query\_7\_1 TO holder;

--

-- TOC entry 3272 (class 0 OID 0)

-- Dependencies: 216

-- Name: SEQUENCE quittance\_num; Type: ACL; Schema: public; Owner: postgres

--

GRANT USAGE ON SEQUENCE public.quittance\_num TO worker;

--

-- TOC entry 3274 (class 0 OID 0)

-- Dependencies: 209

-- Name: SEQUENCE realization\_id\_seq; Type: ACL; Schema: public; Owner: postgres

--

GRANT USAGE ON SEQUENCE public.realization\_id\_seq TO worker;

--

-- TOC entry 3275 (class 0 OID 0)

-- Dependencies: 235

-- Name: TABLE realization\_view; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.realization\_view TO holder;

--

-- TOC entry 3277 (class 0 OID 0)

-- Dependencies: 211

-- Name: SEQUENCE shops\_id\_seq; Type: ACL; Schema: public; Owner: postgres

--

GRANT USAGE ON SEQUENCE public.shops\_id\_seq TO administrator;

-- TOC entry 3279 (class 0 OID 0)

-- Dependencies: 213

-- Name: SEQUENCE statuses\_id\_seq; Type: ACL; Schema: public; Owner: postgres

--

GRANT USAGE ON SEQUENCE public.statuses\_id\_seq TO administrator;

--

-- TOC entry 3280 (class 0 OID 0)

-- Dependencies: 237

-- Name: TABLE test\_view; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.test\_view TO holder;

--

-- TOC entry 3281 (class 0 OID 0)

-- Dependencies: 217

-- Name: SEQUENCE ticket\_num; Type: ACL; Schema: public; Owner: postgres

--

GRANT USAGE ON SEQUENCE public.ticket\_num TO worker;

--

-- TOC entry 3283 (class 0 OID 0)

-- Dependencies: 236

-- Name: TABLE workers\_view; Type: ACL; Schema: public; Owner: postgres

--

GRANT SELECT ON TABLE public.workers\_view TO holder;

-- Completed on 2021-06-04 08:56:07

--

-- PostgreSQL database dump complete

--

# Приложение Г. Антиплагиат

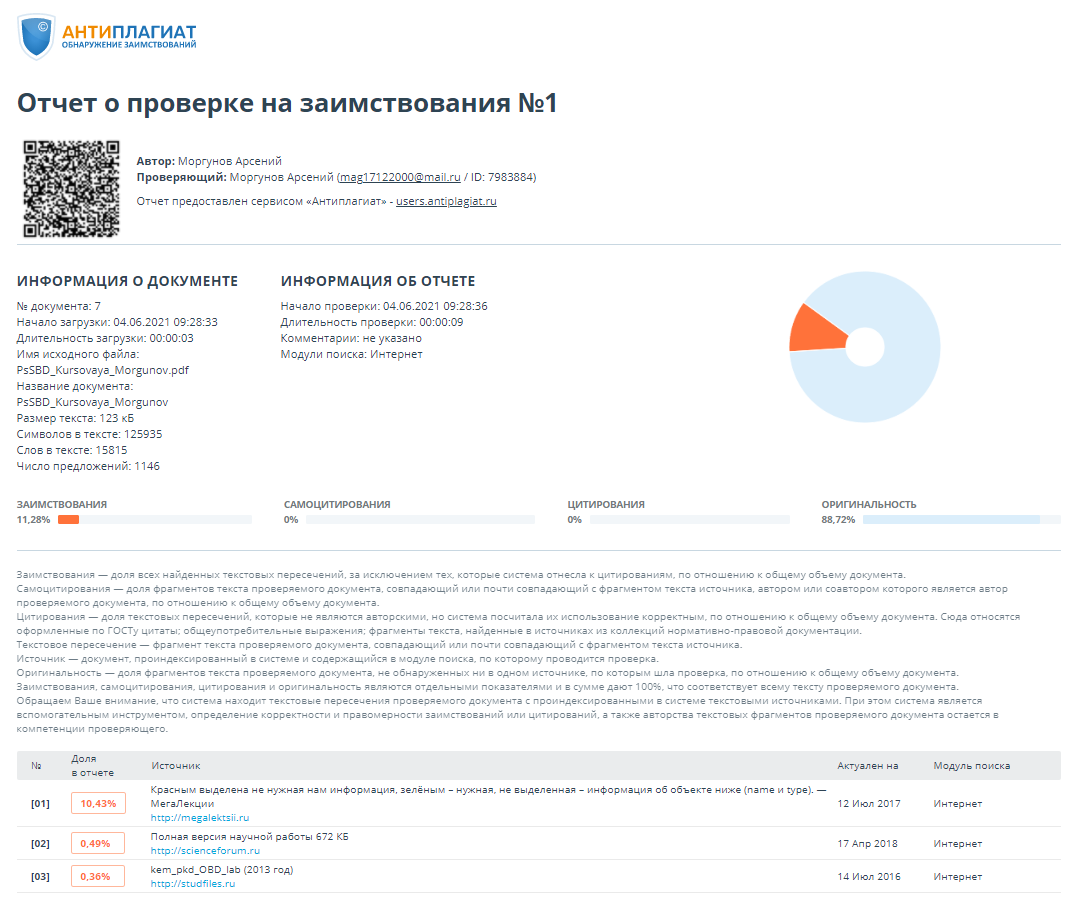


Рисунок Г.1 – Антиплагиат

# Приложение Д. Руководство работника

Рассмотрим вкладку «Продукты». В таблице выводится информация о товарах, которые распологаются в магазине работника, который вошел в систему, при этом выводятся только товары, количество которых на складе больше 0, т.е. еще есть на складе (это обеспечивается политиками работников). При нажатии на товар в таблице в блоке интерфейса справа отображается подробная информация о товаре, а также о клиенте, который его сдал в магазин. Также при нажатии на товар становится активной кнопка «Продать выделенный товар», при нажатии на которую продается указанное количество выделенного товара.Для поиска товара по части названия достаточно ввести часть названия товара в поле «Имя товара» и нажать кнопку «Поиск по имени». Нажатие на копку «Поиск товаров в наличии» выводит в таблицу все товары, которые распологаются в магазине работника, который вошел в систему, при этом выводятся только товары, количество которых на складе больше 0, т.е. еще есть на складе.Для добавления товара на склад нужно заполнить поля, которые находятся в правой нижней части формы, после чего нажать на кнопку «Добавить товар».

Рассмотрим вкладку «Реализация». В левой части формы расположена таблица, которая отображает информацию о всех продажах, совершенных в этом магазине. Если нажать на элемент в этой таблице, то в таблице, расположенной в правой части формы выведется информация о товаре, который был приобретен выбранным чеком. Если нажать на кнопку «Поиск по реализации», то в левой таблице будут выведены все продажи, в которых количество приобретенного товара будет больше указанного.В правой части формы расположена таблица, содержащая все товары, которые были и есть на складе в текущем магазине. При нажатии на элемент в этой таблице, то в таблице, которая расположена слева, выведутся все чеки, выписанные на выбранный товар. При нажатии на кнопку «Поиск по имени» в правой таблице выведутся все товары, у которых в названии содержится указанная последовательность символов.

Рассмотрим вкладку «Клиенты». В таблице «Информация о клиенте» выводится информация обо всех клиентах, которые содержатся в базе данных. При нажатии на элемент таблицы «Информация о клиенте» в таблице «Товары клиента» выведутся все товары клиента, которые он сдал в текущий магазин. При нажатии на кнопку «Поиск по всем клиентам» в таблице «Информация о клиенте» выведутся все клиенты, у которых в имени содержится последовательность символов, указанная в поле «Имя» и в фамилии содержится последовательность символов, указанная в поле «Фамилия». При нажатии на кнопку «Поиск по клиентам магазина» в таблице «Информация о клиенте» выведутся клиенты текущего магазина (т.е. они сдали хотя бы один товар в этот магазин), у которых в имени содержится последовательность символов, указанная в поле «Имя» и в фамилии содержится последовательность символов, указанная в поле «Фамилия». Для добавления клиента необходимо заполнить все поля, которые находятся в блоке «Добавление клиента», а затем нажать кнопку «Добавить клиента». В таблице «Товары клиента» выводится информация обо всех товарах, которые были или есть на складе текущего магазина. При нажатии на элемент таблицы «Товары клиента» в таблице «Информация о клиенте» будет выведен клиент, который сдал выделенный товар. При нажатии на кнопку «Поиск по товарам» в таблице «Товары клиента» будут выведены товары, в названии которых содержится последовательность символов, которая указана в поле «Название товара».

# Приложение е. Руководство Администратора

Рассмотрим вкладку «Роли». В таблице «Рабонтики» выводятся все работники, находящиеся в базе данных. При нажатии на кнопку «Найти работника» в таблицу «Роли» выводятся работники, у которых в логине есть последовательность символов, которая указана в поле «Логин работника». Чтобы добавть нового работника необходимо ввести его логин, пароль в поля «Логин работника» и «Пароль работника» соответственно, а также выбрать магазин, в который он будет добавлен. После этого нужно нажать на кнопку «Добавить работника». Для увольнения работника необходимо выделить элемент таблицы «Работники», который содержит работника, которого нужно уволить, а затем нажать на кнопку «Уволить работника», после чего занчение поля «is\_available» изменится на false, а роль работкника будет удалена из базы данных.

В таблице «Владельцы» выводятся все владельцы. При нажатии на кнопку «Найти владельца» в таблице «Владельцы» будут выведены владельцы, у которых в логине есть последовательность символов, которая указана в поле «Логин владельца». Чтобы добавть нового владельца необходимо ввести его логин, пароль в поля «Логин владельца» и «Пароль владельца» соответственно. После этого нужно нажать на кнопку «Добавить владельца». Для удаления владельца необходимо выделить элемент таблицы «Владельцы», который содержит владельца, которого нужно удалить, а затем нажать на кнопку «Удалить владельца».

Рассмотрим вкладку «Справочники». Для выбора справочника, необходимо в выпадающем списке выбрать имя справочника. Просле этого в таблице отобразится содержимое справочника. Для того чтобы найти запись в справочнике нужно ввести часть названия и нажать кнопку «Найти». Для добавления записи в справочник необходимо ввести название новой записи и нажать кнопку «Добавить». Для удаления записи необходимо выбрать запись в таблице и нажать кнопку «Удалить выбранное».

# Приложение Ё. Руководство владельца

Рассмотрим вкладку «Таблицы». Для выбора таблицы, необходимо в выпадающем списке выбрать имя таблицы. Просле этого в таблице на экране отобразится содержимое выбранной таблицы. Для того чтобы найти запись в таблице нужно ввести последовательность символов, которую должны включать искомые записи, и нажать кнопку «Найти». Для таблицы работников поиск происходит по логину, для таблицы товары и реализация – по названию товара, для таблицы заказчики по имени.

Рассмотрим вкладку «Запросы». Для выбора запроса необходимо выбрать его из выпадающего списка. Если в запросе есть параметр, то его необходимо записать в поле «Параметр», если есть второй параметр, то его нужно записать в поле «Параметр 2». Если все параметры введены, либо запрос не требует параметров, то нужно нажать кнопку «Выполнить запрос», для отображения результатов запроса в виде таблицы.

Рассмотрим вкладку «График». Для того, чтобы просмотреть статистику в виде графика и таблицы необходимо нажать кнопку «Построить диаграмму». Для того, чтобы сгенерировать Exel файл по статистике необходимо нажать на кнопку «Импорт в Exel», а затем выбрать расположение и имя создаваемого файла, после чего нажать на кнопку «Сохранить».