



МИНОБРАЗОВАНИЯ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА—Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий
(наименование института, филиала)
Кафедра КБ-3 «Безопасность программных решений»
(наименование кафедры)

Утверждаю
Заведующий кафедрой КБ-3

Горин Д.С.
(Ф.И.О.) (подпись)
«11» февраля 2023 г.

ЗАДАНИЕ

на выполнение курсовой работы

по дисциплине Принципы, технологии и средства организации данных компонентов и программного обеспечения
(наименование дисциплины)

Тема курсовой работы Создание клиент-серверного приложения для работы с базой данных

Студент Саримов В.С. БСБО-04-21
(учебная группа, фамилия, имя, отчество студента) Саримов
(подпись студента)

Исходные данные Вариант № 99

Перечень вопросов, подлежащих обработке, и обязательного графического материала:

Создание отлаженного клиентского приложения для работы с БД

Создание БД, схемы таблиц, индексов, хранимых функций, процедур, триггеров, транзакций

Реализация прав доступа к объектам БД, подключение к БД и демонстрация работы с учетом различий по доступу в программе-клиенте

Реализация в программе-клиенте запросов согласно заданию

Срок предоставления к защите курсовой работы до « 20 » мая 2023 г.

Задание на курсовую работу выдал Филатов В.В.
(Ф.И.О. руководителя) Филатов
(подпись руководителя)

«11» февраля 2023 г.

Задание на курсовую работу получил Саримов В.С.
(Ф.И.О. исполнителя) Саримов
(подпись исполнителя)

«11» февраля 2023 г.

Оглавление

Задание	3
---------------	---

Схема БД.....	4
Введение	5
Техническая информация.....	6
База данных	8
Создание таблиц.....	8
Запросы	9
Составной многотабличный запрос с CASE-выражением	9
Многотабличный VIEW, с возможностью его обновления	10
Запросы, содержащий подзапрос в разделах SELECT, FROM, WHERE.....	13
Коррелированные подзапросы	14
Многотабличный запрос, содержащий группировку записей, агрегатные функции и параметр, используемый в разделе HAVING	15
Запросы, содержащий предикат ANY	16
Индексы	17
Триггеры	18
Операции добавления, удаления и обновления реализовать в виде хранимых процедур или функций с параметрами для всех таблиц.....	20
Хранимая процедура или функция, состоящая из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана	24
Курсор на обновления отдельных данных (вычисления значения полей выбранной таблицы)	25
Скалярная и векторная функции	26
Распределение прав пользователей: предусмотреть не менее двух пользователей с разным набором привилегий	27
Список литературы	28
Приложение	29
Приложение А	29

Задание

Разработать клиент-серверное приложение, серверная часть которого должна быть реализована на Microsoft SQL Server или PostgreSQL, представляющая собой модель предметной области в соответствии с вариантом задания. В рамках заданной предметной области реализовать заданную (по варианту) схему отношений, т.е. выделить сущности и их атрибуты, так чтобы связи между сущностями соответствовали представленной схеме.

99	Информационно-кассовая система	Учет покупок на кассе (цены, списание со склада), формирование чеков. Информация о кассире	3
----	--------------------------------	--	---

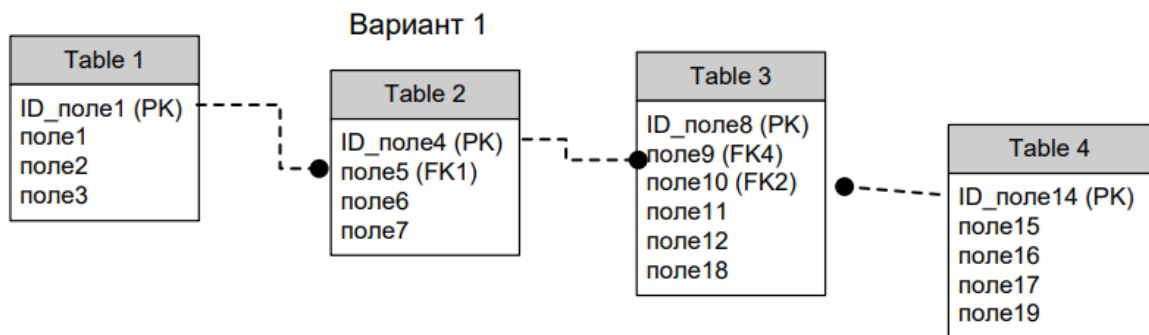
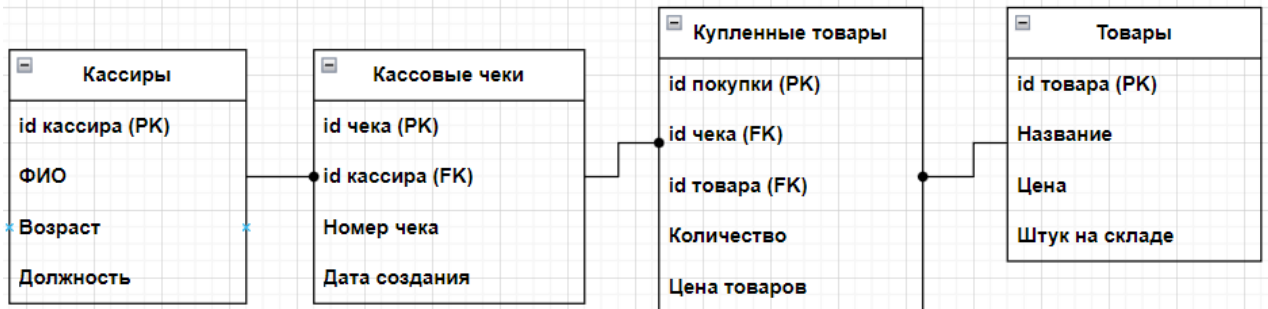


Схема БД



Введение

В данной курсовой работе задачей было разработать клиент-серверное приложение для информационной-кассовой системы. Для решения этой задачи, как сервер использовался PostgreSQL, а для клиентского интерфейса — язык программирования с#. Для соединения этих частей использовалась библиотека Npgsql. Для разделения прав доступа в базе данных с целью защиты информации, использовалось разделение на две роли: администратора и пользовательской. “Администратор” имеет полные права на базу данных, “юзер” — может получать только выборки. Сама база данных представляет из себя 4 таблицы, включающих в себя: таблицы с информацией о кассирах, товарах, чеках, которые пробивают кассиры, товарах, которые продаются в магазине и информацию о купленных товарах в этом магазине; триггеров и курсора; связей между таблицами, с помощью внешних ключей, процедуры, позволяющие заполнять, изменять и удалять данные из таблицы, процедуры и функции, возвращающие табличные и скалярные значения; индексы.

Внешние ключи соединяют: товары и купленные товары, кассиры и чеки, чеки и купленные товары; передавая значения идентификаторов записей из первых таблиц во вторые.

В клиентском интерфейсе предусмотрена защита от неправильно введенных данных. При несоответствии внешнего ключа с идентификаторами той таблице, которую он соединяет, будет введена ошибка и запрос не будет отправлен в БД.

Техническая информация

В данной главе описывается работа приложения со стороны клиента. При запуске приложения пользователь сначала попадает в окно авторизации. В предыдущем разделе было указано, что в базе данных предусмотрено два пользователя, для каждого из которых имеется пароль. После успешной авторизации клиент попадает в главное меню.

В главном меню пользователю предоставляются кнопки для перехода к различным выборкам и к основным четырём таблицам. Форма для работы с основными таблицами является динамической. При нажатии на кнопку создается конструктор формы с параметрами, которые определены в функции.

В форме для основных таблиц содержится таблица, созданная с использованием элемента DataGridView, а также три кнопки button, предназначенные для добавления, обновления и удаления данных, а также различные выборки по таблицам.

При загрузке формы происходит передача данных из параметра в переменные класса формы. Затем столбцам из этих данных присваивается имя, а с помощью запроса SELECT таблица в базе данных заполняется данными (за исключением идентификатора, который заполняется отдельным массивом). При удалении данных из таблицы извлекается индекс выделенной строки, который используется в качестве индекса массива идентификаторов. Затем создается и отправляется запрос в базу данных для удаления соответствующей строки.

Для заполнения полей при добавлении и изменении данных существует отдельная форма, которая вызывается в виде диалогового окна. В этот диалог также передаются типы полей и ограничения. После нажатия кнопки "Добавить" или "Изменить" поля проходят проверку на соответствие указанным типам данных. В случае несоответствия выводится ошибка, которая не позволяет закрыть диалоговое окно, и появляется MessageBox, информирующий пользователя об ошибке.

При успешном завершении диалогового окна обработанные данные извлекаются, а затем добавляются в строку с текстом запроса. При изменении данных в начале поля диалогового окна заполняются исходными данными. Кроме того, в приложении присутствуют 10 кнопок для выполнения отдельных запросов, функций, процедур и курсоров. Эти кнопки позволяют выполнять различные операции и получать необходимую информацию из базы данных.

База данных

Создание таблиц

```
CREATE TABLE kassiri(  
    id SERIAL PRIMARY KEY,  
    fio_kassira VARCHAR(50) NOT NULL,  
    age INT NOT NULL,  
    job varchar(50) NOT NULL  
);  
  
CREATE TABLE cheki(  
    id SERIAL PRIMARY KEY,  
    kas_id int,  
    number_chek SERIAL,  
    date_create TIMESTAMP(0) DEFAULT NOW(),  
    FOREIGN KEY (kas_id) REFERENCES kassiri(id) ON DELETE CASCADE  
);  
  
CREATE TABLE products(  
    id SERIAL PRIMARY KEY,  
    nazvanie VARCHAR(50) NOT NULL,  
    price DECIMAL(15,2) NOT NULL,  
    quantity_stock INT NOT NULL  
);  
  
CREATE TABLE kuplen_products(  
    id SERIAL PRIMARY KEY,  
    check_id INT,  
    product_id INT,  
    quantity INT NOT NULL,  
    price_products DECIMAL(15,2) NOT NULL,  
    FOREIGN KEY (check_id) REFERENCES cheki(id) ON DELETE CASCADE,  
    FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE  
);
```


Запросы

Составной многотабличный запрос с CASE-выражением

```
SELECT
    CASE
        WHEN total_cost >= 0 AND total_cost <= 500 THEN 'От 0 до 500
рублей'
        WHEN total_cost >= 501 AND total_cost <= 1500 THEN 'От 501 до 1500
рублей'
        WHEN total_cost >= 1501 AND total_cost <= 3000 THEN 'От 1501 до
3000 рублей'
        WHEN total_cost >= 3001 THEN 'От 3001 рубля'
    END AS category,
    COUNT(*) AS quantity_chek
FROM
    (
        SELECT ch.id, SUM(kp.price_products) AS total_cost
        FROM cheki ch JOIN kuplen_products kp ON ch.id = kp.check_id
        GROUP BY ch.id
    ) AS subquery
GROUP BY
    category;
```

Многотабличный VIEW, с возможностью его обновления

```
CREATE OR REPLACE VIEW my_view AS
```

```
SELECT
```

```
    k.fio_kassira,
```

```
    k.job,
```

```
    ch.id,
```

```
    ch.kas_id,
```

```
    ch.date_create
```

```
FROM cheki ch
```

```
    JOIN kassiri k ON ch.kas_id = k.id
```

```
    ORDER BY ch.number_chek;
```

```
CREATE OR REPLACE PROCEDURE Insert_my_view(_fio_kassira VARCHAR(50), _job  
VARCHAR(50))
```

```
LANGUAGE plpgsql
```

```
AS $$
```

```
    DECLARE _kas_id INT;
```

```
    BEGIN
```

```
        SELECT id INTO _kas_id FROM kassiri WHERE fio_kassira =  
_fio_kassira and job = _job;
```

```
        INSERT INTO my_view(kas_id) VALUES(_kas_id) ;
```

```
        IF (_kas_id IS NULL) THEN
```

```
            RAISE EXCEPTION 'Кассира не существует';
```

```
        END IF;
```

```
    END;
```

```
$$;
```

```
CREATE OR REPLACE FUNCTION insert_row_view()
```

```
RETURNS trigger AS
```

```
$$
```

```
BEGIN
```

```
    INSERT INTO cheki(kas_id) VALUES (NEW.kas_id);
```

```
    RETURN NULL;
```

```
END;
```

```
$$
```

```
LANGUAGE 'plpgsql';
```

```

CREATE OR REPLACE TRIGGER insert_view
INSTEAD OF INSERT ON my_view
FOR EACH ROW
EXECUTE PROCEDURE insert_row_view();

CREATE OR REPLACE PROCEDURE Delete_my_view(_id INT)
LANGUAGE plpgsql
AS $$
    BEGIN
        DELETE FROM my_view WHERE id = _id;
    END;
$$;

CREATE OR REPLACE FUNCTION delete_row_view()
RETURNS trigger AS
$$
BEGIN
    DELETE FROM cheki WHERE id = OLD.id;
    RETURN NULL;
END;
$$
LANGUAGE 'plpgsql';

CREATE OR REPLACE TRIGGER delete_view
INSTEAD OF DELETE ON my_view
FOR EACH ROW
EXECUTE PROCEDURE delete_row_view();

CREATE OR REPLACE PROCEDURE Update_my_view(_id INT, _fio_kassira
VARCHAR(50), _job VARCHAR(50))
LANGUAGE plpgsql
AS $$
    DECLARE _kas_id INT;
    BEGIN
        SELECT id INTO _kas_id FROM kassiri WHERE fio_kassira =
_fio_kassira and job = _job;
        UPDATE my_view
        SET kas_id = _kas_id WHERE id = _id;
    END;

```

```

        IF (_kas_id IS NULL) THEN
            RAISE EXCEPTION 'Кассира не существует';
        END IF;
    END;
$$;

CREATE OR REPLACE FUNCTION update_row_view()
RETURNS trigger AS
$$
BEGIN
    UPDATE cheki
        SET kas_id = NEW.kas_id WHERE id = OLD.id;
    RETURN NULL;
END;
$$
LANGUAGE 'plpgsql';

CREATE OR REPLACE TRIGGER update_view
INSTEAD OF UPDATE ON my_view
FOR EACH ROW
EXECUTE PROCEDURE update_row_view();

```

Запросы, содержащий подзапрос в разделах SELECT, FROM, WHERE

Первый запрос

```
SELECT
    ch.id,
    ch.number_chek,
    (SELECT SUM(kuplen_products.quantity)
     FROM kuplen_products
     WHERE kuplen_products.check_id = ch.id)
    AS number_positions
FROM
    cheki ch;
```

Второй запрос

```
SELECT p.id, p.nazvanie, sub.total_quantity
FROM products p
JOIN (
    SELECT product_id, SUM(quantity) AS total_quantity
    FROM kuplen_products
    GROUP BY product_id
) AS sub ON p.id = sub.product_id;
```

Третий запрос

```
SELECT k.id, k.fio_kassira
FROM kassiri k
WHERE k.id IN (
    SELECT c.kas_id
    FROM cheki c
    JOIN kuplen_products kp ON c.id = kp.check_id
    JOIN products p ON p.id = kp.product_id
    WHERE p.nazvanie = 'Батон'
);
```

Коррелированные подзапросы

Первый запрос

```
SELECT * FROM kassiri k
WHERE EXISTS (
    SELECT 1
    FROM cheki ch
    JOIN kuplen_products ON ch.id = kuplen_products.check_id
    WHERE k.id = ch.kas_id
    GROUP BY ch.kas_id
    HAVING SUM(kuplen_products.price_products) > 10000
);
```

Второй запрос

```
SELECT products.nazvanie, (
    SELECT COUNT(*) FROM kuplen_products kp
    WHERE kp.product_id = products.id)
    AS number_pokupok FROM products;
```

Третий запрос

```
SELECT * FROM kassiri k WHERE NOT EXISTS (
    SELECT ch.id FROM cheki ch
    WHERE ch.kas_id = k.id
    AND NOT EXISTS (SELECT 1 FROM kuplen_products kp
        WHERE kp.check_id = ch.id)
);
```

Многотабличный запрос, содержащий группировку записей, агрегатные функции и параметр, используемый в разделе HAVING

```
SELECT k.id, k.fio_kassira, COUNT(cheki.id) AS quantity_check
FROM kassiri k
JOIN cheki ON k.id = cheki.kas_id
GROUP BY k.id, k.fio_kassira
HAVING COUNT(cheki.id) > 0;
```


Запросы, содержащий предикат ANY

```
SELECT number_chek, SUM(kuplen_products.price_products) AS Сумма
FROM cheki
JOIN kuplen_products ON cheki.id = kuplen_products.check_id
WHERE cheki.id = ANY (SELECT id FROM cheki)
GROUP BY cheki.id;
```


Индексы


Первый индекс

```
CREATE INDEX indx_kas ON kassiri (id);
```

	QUERY PLAN	
	text	
1	Index Scan using kassiri_pkey on kassiri (cost=0.28..8.29 rows=1 width=35) (actual time=0.009..0.010 rows=1 loops=1)	
2	Index Cond: (id = 223)	
3	Planning Time: 2.015 ms	
4	Execution Time: 0.029 ms	


Второй индекс

```
CREATE INDEX indx_chek ON cheki USING hash (number_chek);
```

	QUERY PLAN	
	text	
1	Index Scan using indx_chek on cheki (cost=0.00..8.02 rows=1 width=20) (actual time=0.020..0.021 rows=1 loops=1)	
2	Index Cond: (number_chek = 193)	
3	Planning Time: 1.311 ms	
4	Execution Time: 0.039 ms	

Третий индекс

```
CREATE INDEX indx_tovar ON products USING gin (nazvanie gin_trgm_ops);
```

	QUERY PLAN	
	text	
1	Aggregate (cost=169.16..169.17 rows=1 width=8) (actual time=0.363..0.363 rows=1 loops=1)	
2	-> Bitmap Heap Scan on products (cost=52.34..169.05 rows=44 width=0) (actual time=0.361..0.361 rows=0 loops=1)	
3	Recheck Cond: ((nazvanie)::text = 'кефир'::text)	
4	Rows Removed by Index Recheck: 44	
5	Heap Blocks: exact=44	
6	-> Bitmap Index Scan on indx_tovar (cost=0.00..52.33 rows=44 width=0) (actual time=0.305..0.305 rows=44 loop=1)	
7	Index Cond: ((nazvanie)::text = 'кефир'::text)	
8	Planning Time: 1.833 ms	
9	Execution Time: 0.420 ms	

Триггеры

```
CREATE OR REPLACE FUNCTION dob()  
RETURNS trigger AS  
$$  
BEGIN  
        UPDATE products  
        SET quantity_stock = quantity_stock - NEW.quantity WHERE id =  
NEW.product_id;  
        RETURN NEW;  
END;  
$$  
LANGUAGE 'plpgsql';
```

```
CREATE TRIGGER trigger_add  
AFTER INSERT  
ON kuplen_products  
FOR EACH ROW  
EXECUTE FUNCTION dob();
```

```
CREATE OR REPLACE FUNCTION upd()  
RETURNS trigger AS  
$$  
BEGIN  
        UPDATE products  
        SET quantity_stock = quantity_stock - (NEW.quantity -  
OLD.quantity) WHERE id = NEW.product_id;  
        RETURN NEW;  
END;  
$$  
LANGUAGE 'plpgsql';
```

```
CREATE TRIGGER trigger_upd  
AFTER UPDATE  
ON kuplen_products  
FOR EACH ROW  
EXECUTE FUNCTION upd();
```

```
CREATE OR REPLACE FUNCTION del()
```

```

RETURNS trigger AS
$$
BEGIN
    UPDATE products
    SET quantity_stock = quantity_stock + OLD.quantity WHERE id =
OLD.product_id;
    RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

CREATE TRIGGER trigger_del
AFTER DELETE
ON kuplen_products
FOR EACH ROW
EXECUTE FUNCTION del();

```

Операции добавления, удаления и обновления реализовать в виде хранимых процедур или функций с параметрами для всех таблиц

```
CREATE OR REPLACE PROCEDURE Insert_products(_nazvanie VARCHAR(50), _price
DECIMAL(15,2), _quantity_stock INT)
LANGUAGE SQL
AS $$
    INSERT INTO products(nazvanie, price, quantity_stock) VALUES
    (_nazvanie, _price, _quantity_stock);
$$;
```

```
CREATE OR REPLACE PROCEDURE Insert_kassiri(_fio_kassira VARCHAR(50), _age
INT, _job VARCHAR(50))
LANGUAGE SQL
AS $$
    INSERT INTO kassiri(fio_kassira, age, job) VALUES (_fio_kassira, _age,
_job);
$$;
```

```
CREATE OR REPLACE PROCEDURE Insert_cheki(_fio_kassira VARCHAR(50), _job
VARCHAR(50))
AS $$
DECLARE
    DECLARE _kas_id INT;
BEGIN
    SELECT id INTO _kas_id FROM kassiri WHERE fio_kassira = _fio_kassira
and job = _job;
    BEGIN
        INSERT INTO cheki(kas_id) VALUES (_kas_id);
        IF (_kas_id IS NOT NULL) THEN
            COMMIT;
            RAISE NOTICE 'Транзакция зафиксирована.';
        ELSE
            ROLLBACK;
            RAISE EXCEPTION 'Кассир не существует';
        END IF;
    END;
END;
$$ LANGUAGE plpgsql;
```

```

CREATE OR REPLACE PROCEDURE Insert_kuplen_products(_nazvanie VARCHAR(50),
_quantity INT)
AS $$
DECLARE
    DECLARE _product_id INT;
BEGIN
    SELECT id INTO _product_id FROM products WHERE (nazvanie = _nazvanie
and quantity_stock>=_quantity);
    BEGIN
        INSERT INTO kuplen_products(product_id, check_id, quantity,
price_products)
            VALUES(_product_id, (SELECT id FROM cheki order by id DESC LIMIT
1), _quantity, _quantity*(SELECT price FROM products WHERE id =
_product_id));
        IF (_product_id IS NOT NULL) THEN
            COMMIT;
            RAISE NOTICE 'Транзакция зафиксирована.';
        ELSE
            ROLLBACK;
            RAISE EXCEPTION 'Товар не существует';
        END IF;
    END;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE PROCEDURE Delete_kuplen_products(_id INT)
LANGUAGE plpgsql
AS $$
    BEGIN
        DELETE FROM kuplen_products WHERE id = _id;
    END;
$$;

CREATE OR REPLACE PROCEDURE Delete_cheki(_id INT)
LANGUAGE plpgsql
AS $$
    BEGIN
        DELETE FROM cheki WHERE id = _id;

```

```

        END;
    $$;

CREATE OR REPLACE PROCEDURE Delete_kassiri(_id INT)
LANGUAGE plpgsql
AS $$
    BEGIN
        DELETE FROM kassiri WHERE id = _id;
    END;
$$;

CREATE OR REPLACE PROCEDURE Delete_products(_id INT)
LANGUAGE plpgsql
AS $$
    BEGIN
        DELETE FROM products WHERE id = _id;
    END;
$$;

CREATE OR REPLACE PROCEDURE Update_products(_id INT, _nazvanie
VARCHAR(50), _price DECIMAL(15,2), _quantity_stock INT)
LANGUAGE SQL
AS $$
    UPDATE products
        SET nazvanie = _nazvanie, price = _price, quantity_stock =
        _quantity_stock WHERE id = _id;
$$;

CREATE OR REPLACE PROCEDURE Update_kassiri(_id INT, _fio_kassira
VARCHAR(50), _age INT, _job VARCHAR(50))
LANGUAGE SQL
AS $$
    UPDATE kassiri
        SET fio_kassira = _fio_kassira, age = _age, job = _job WHERE id =
        _id;
$$;

CREATE OR REPLACE PROCEDURE Update_cheki(_id INT, _fio_kassira
VARCHAR(50), _job VARCHAR(50))

```

```

LANGUAGE plpgsql
AS $$
    DECLARE _kas_id INT;
    BEGIN
        SELECT id INTO _kas_id FROM kassiri WHERE fio_kassira =
        _fio_kassira and job = _job;
        UPDATE cheki
        SET kas_id = _kas_id WHERE id = _id;
        IF (_kas_id IS NULL) THEN
            RAISE EXCEPTION 'Кассира не существует';
        END IF;
    END;
$$;

CREATE OR REPLACE PROCEDURE Update_kuplen_products(_id INT, _nazvanie
VARCHAR(50), _quantity INT)
LANGUAGE plpgsql
AS $$
    DECLARE _product_id INT;
    BEGIN
        SELECT id INTO _product_id FROM products WHERE (nazvanie =
        _nazvanie and quantity_stock>=_quantity);
        UPDATE kuplen_products
        SET product_id = _product_id, check_id = (SELECT id FROM cheki
        order by id DESC LIMIT 1), quantity = _quantity, price_products = _quantity
        *(SELECT price FROM products WHERE id = _product_id) WHERE id = _id;
        IF (_product_id IS NULL) THEN
            RAISE EXCEPTION 'Кассира не существует';
        END IF;
    END;
$$;

```

Хранимая процедура или функция, состоящая из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана

```
CREATE OR REPLACE PROCEDURE Insert_kuplen_products(_nazvanie VARCHAR(50),
_quantity INT)
AS $$
DECLARE
    DECLARE _product_id INT;
BEGIN
    SELECT id INTO _product_id FROM products WHERE (nazvanie = _nazvanie
and quantity_stock>=_quantity);
    BEGIN
        INSERT INTO kuplen_products(product_id, check_id, quantity,
price_products)
            VALUES(_product_id, (SELECT id FROM cheki order by id DESC
LIMIT 1), _quantity, _quantity*(SELECT price FROM products WHERE id =
_product_id));
        IF (_product_id IS NOT NULL) THEN
            COMMIT;
            RAISE NOTICE 'Транзакция зафиксирована.';
        ELSE
            ROLLBACK;
            RAISE EXCEPTION 'Товар не существует';
        END IF;
    END;
END;
$$ LANGUAGE plpgsql;
```


Курсор на обновления отдельных данных (вычисления значения полей выбранной таблицы)

```
CREATE OR REPLACE FUNCTION price_inc()
RETURNS TABLE (id INT, nazvanie VARCHAR(50), price DECIMAL(15,2),
quantity_stock INT) AS $$
DECLARE
    cur_r RECORD;
    new_price DECIMAL(15,2);
BEGIN
    FOR cur_r IN SELECT * FROM products LOOP
        new_price := cur_r.price * 1.1;
        UPDATE products SET
            price = new_price
        WHERE products.id = cur_r.id;
    END LOOP;
    RETURN QUERY SELECT * FROM products;
END;
$$ LANGUAGE plpgsql;
```

Скалярная и векторная функции

Скалярная функция

```
CREATE OR REPLACE FUNCTION count_chek()  
RETURNS INT  
AS $$  
DECLARE  
    chek_counter INT;  
BEGIN  
    SELECT COUNT(*) INTO chek_counter  
    FROM cheki;  
  
    RETURN chek_counter;  
END;  
$$ LANGUAGE plpgsql;
```

Векторная функция

```
CREATE OR REPLACE FUNCTION kassi()  
RETURNS TABLE (id INT, fio_kassira VARCHAR(50), age INT, job VARCHAR(50))  
AS $$  
BEGIN  
    RETURN QUERY SELECT k.id, k.fio_kassira, k.age, k.job  
    FROM kassiri k;  
END;  
$$ LANGUAGE plpgsql;
```

**Распределение прав пользователей: предусмотреть не менее двух
пользователей с разным набором привилегий**

user

```
CREATE ROLE read_only;
GRANT CONNECT ON DATABASE kassa TO read_only;
GRANT SELECT ON TABLE kassiri to read_only;
GRANT SELECT ON TABLE kuplen_products to read_only;
GRANT SELECT ON TABLE products to read_only;
GRANT SELECT ON TABLE cheki to read_only;
GRANT SELECT ON my_view to read_only;
CREATE USER user1 WITH PASSWORD 'user1';
GRANT read_only TO user1;
```

admin

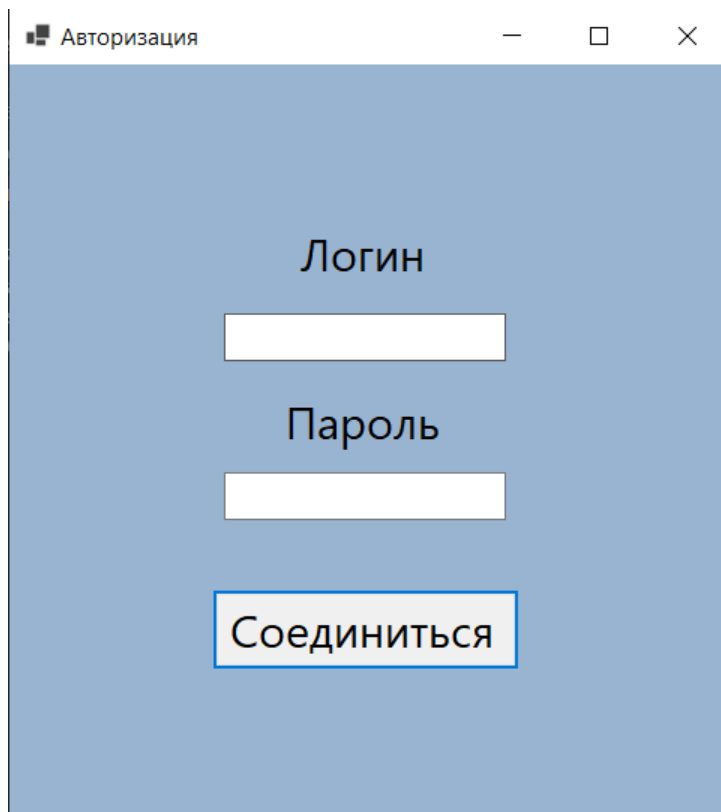
```
CREATE ROLE all_rights;
GRANT CONNECT ON DATABASE kassa TO all_rights;
GRANT ALL PRIVILEGES ON DATABASE "kassa" to all_rights;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO all_rights;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO all_rights;
CREATE USER admin1 with password 'admin1';
GRANT all_rights to admin1;
```

Список литературы

1. Документация PostgreSQL[<https://www.postgresql.org/>]
2. Документация Postgres Pro Standard[<https://postgrespro.ru/docs/>]
3. Интерактивный учебник по SQL[<http://www.sql-tutorial.ru/ru>]
4. Документация Microsoft по Windows Forms[<https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/windows-forms-overview?view=netframeworkdesktop-4.8>]
5. Язык программирования C#[<https://metanit.com/sharp/>]

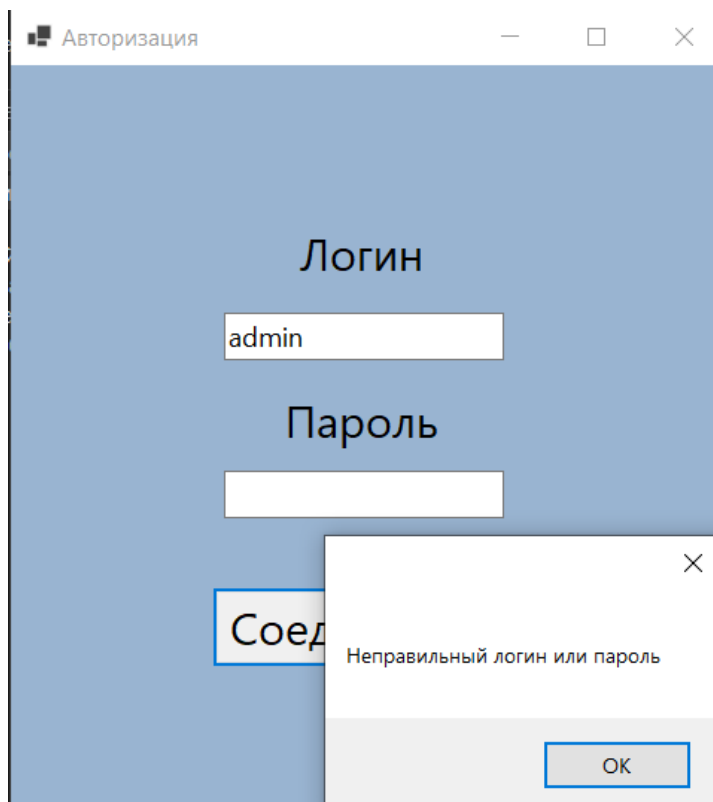
Приложение Приложение А

Вход в приложение



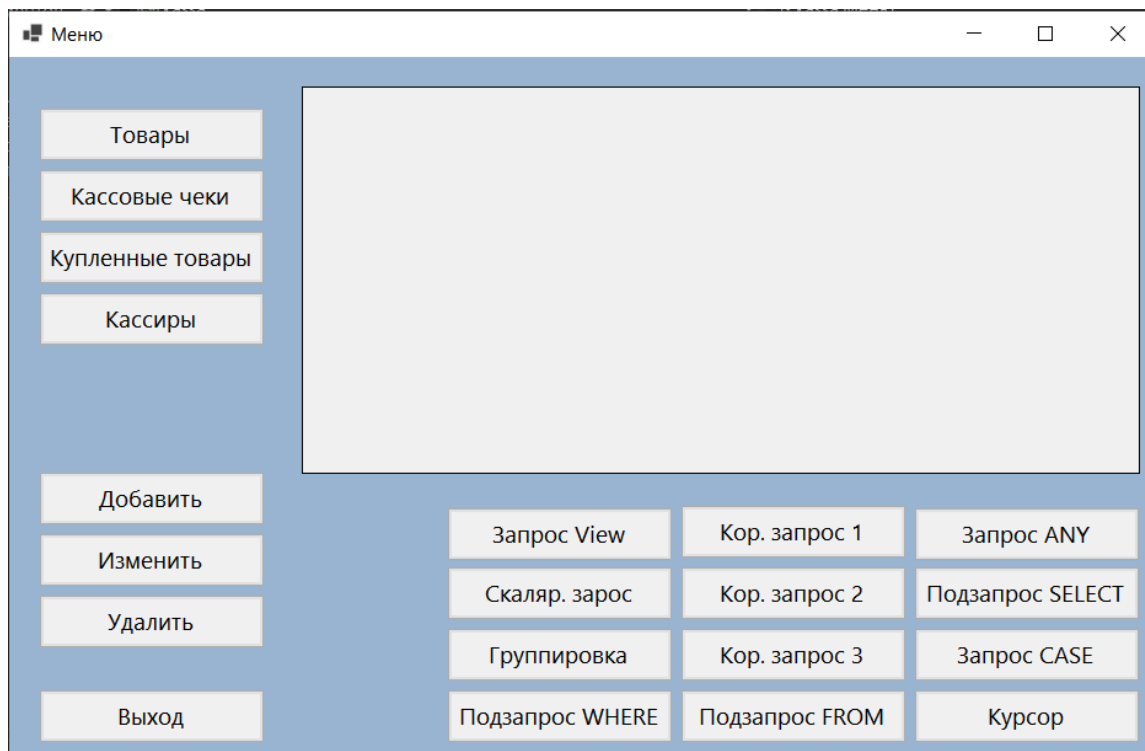
The screenshot shows a login window titled "Авторизация" (Authorization). It has a blue background. In the center, there are two white input fields. The first field is labeled "Логин" (Login) and the second is labeled "Пароль" (Password). Below the password field is a blue button with the text "Соединиться" (Connect).

Контроль доступа в приложение

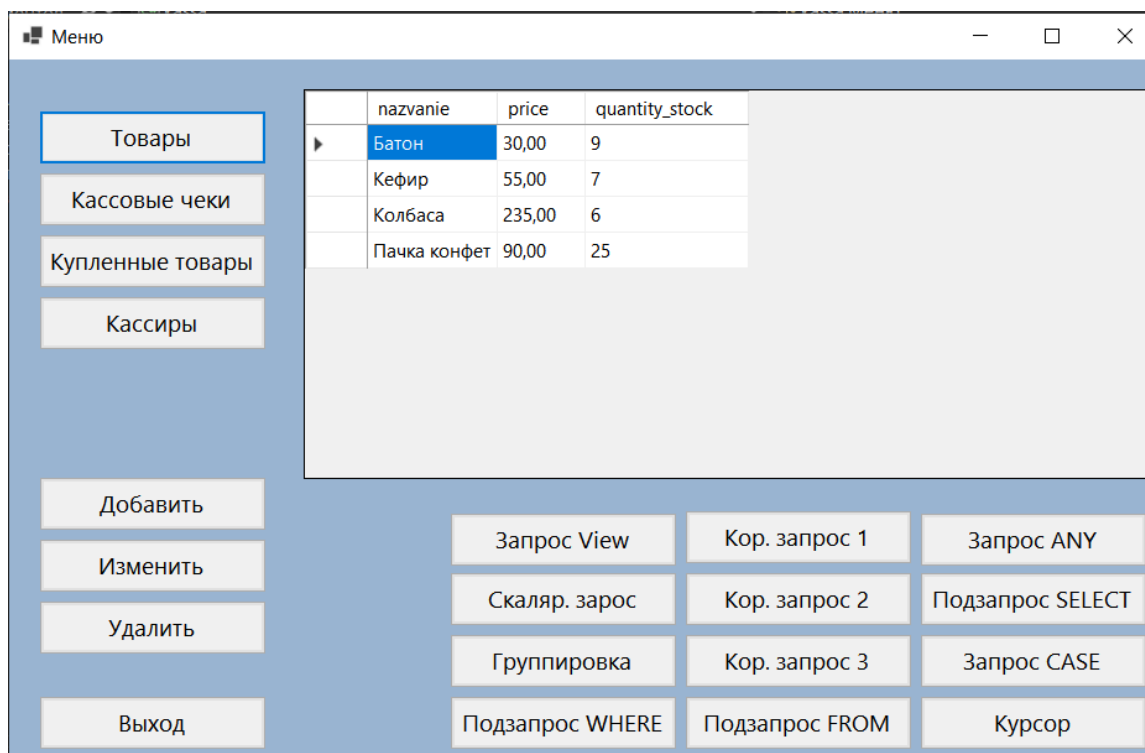


The screenshot shows the same login window as before, but with an error message dialog box overlaid. The dialog box is white with a close button (X) in the top right corner. It contains the text "Неправильный логин или пароль" (Incorrect login or password). At the bottom of the dialog is a blue button labeled "ОК" (OK). In the background, the login fields now contain the text "admin" and an empty password field. The "Соединиться" button is still visible.

Главное меню



Вывод таблицы



Изменение таблицы

Товары

Кассовые чеки

Купленные товары

Кассиры

Добавить

Изменить

Удалить

Выход

nazvanie	price	quantitv	stock
Батон	30,00		
Кефир	55,00		
Колбаса	235,00		
Пачка конфет	90,00		

Запрос

Скаляр

Группировка

Подзапросы

Товары_из

Название

Колбаса

Цена

235,00

Кол-во на складе

6

Изменить