



Министерство науки и высшего образования Российской Федерации  
Калужский филиал  
федерального государственного бюджетного  
образовательного учреждения высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ и информационные технологии»

## ЛАБОРАТОРНАЯ РАБОТА 2

ДИСЦИПЛИНА: «ПРОЕКТИРОВАНИЕ СИСТЕМ ХРАНЕНИЯ И  
ОБРАБОТКИ ДАННЫХ»

Выполнил: студент гр. ИУК4-52Б \_\_\_\_\_ (\_\_\_\_ Губин Е.В.\_\_\_\_)  
(Подпись) (Ф.И.О.)

Проверил: \_\_\_\_\_ (\_\_\_\_ Глебов С.А.\_\_\_\_)  
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

## Создание триггеров и функций:

### 1) Создание нового тарифного плана

```
CREATE OR REPLACE PROCEDURE create_new_traffic_plan
(new_title VARCHAR(50),
new_monthly_cost REAL,
new_internet_data_limit REAL,
new_call_minutes_limit INT,
new_messages_limit INT)
LANGUAGE plpgsql
AS
$$
BEGIN
    INSERT INTO traffic_plans (
        title,
        monthly_cost,
        internet_data_limit,
        call_minutes_limit,
        messages_limit
    )
    VALUES
    (
        new_title,
        new_monthly_cost,
        new_internet_data_limit,
        new_call_minutes_limit,
        new_messages_limit
    );
END;
$$;
```

### 2) Функция, проверяющая принадлежит ли номер телефона клиенту компании

```
CREATE OR REPLACE FUNCTION is_our_client(new_phone_number VARCHAR(50))
RETURNS BOOL
LANGUAGE plpgsql
AS
$$
BEGIN
    RETURN EXISTS (SELECT * FROM sim_cards WHERE phone_number =
new_phone_number);
END;
$$;
```

### 3) Функция, возвращающая id записи из таблицы связи номеров телефона клиент-клиент

```
CREATE OR REPLACE FUNCTION get_customer_to_customer_id
(new_initiator_phone_number VARCHAR(50),
new_interlocutor_phone_number VARCHAR(50))
RETURNS INT
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN (SELECT customer_to_customer_id FROM customer_to_customer
WHERE initiator_phone_number = new_initiator_phone_number
AND interlocutor_phone_number = new_interlocutor_phone_number);
END;
$$;
```

#### 4) Функция, возвращающая id записи из таблицы связи номеров телефона клиент-клиент другой компании

```
CREATE OR REPLACE FUNCTION get_customer_to_other_id
(new_initiator_phone_number VARCHAR(50),
new_interlocutor_phone_number VARCHAR(50),
new_customer_is_initiator BOOL)
RETURNS INT
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN (SELECT customer_to_other_id
            FROM customer_to_other
            WHERE customer_phone_number = new_interlocutor_phone_number AND
                  other_phone_number = new_initiator_phone_number AND customer_is_initiator
            = new_customer_is_initiator);
END;
$$;
```

#### 5) Процедура, вставляющая новое сообщение в зависимости от типа связи

```
CREATE OR REPLACE PROCEDURE insert_message
(new_message_content VARCHAR(255),
new_message_date_time TIMESTAMP,
new_customer_to_customer_id INT,
new_customer_to_orher_id INT)
LANGUAGE plpgsql
AS
$$
BEGIN
    IF (new_customer_to_orher_id IS NULL) THEN
        INSERT INTO messages (message_content, message_date_time,
customer_to_customer_id)
        VALUES (new_message_content, new_message_date_time,
new_customer_to_customer_id);
    ELSE
        INSERT INTO messages (message_content, message_date_time,
customer_to_orher_id)
        VALUES (new_message_content, new_message_date_time,
new_customer_to_orher_id);
    END IF;
END;
$$;
```

#### 6) Процедура создания нового сообщения

```
CREATE OR REPLACE PROCEDURE new_message
(new_message_content VARCHAR(255),
new_message_date_time TIMESTAMP,
new_initiator_phone_number VARCHAR(50),
new_interlocutor_phone_number VARCHAR(50))
LANGUAGE plpgsql
AS
$$
DECLARE
    new_id INT;
BEGIN
    IF (SELECT is_our_client(new_initiator_phone_number))
    AND (SELECT is_our_client(new_interlocutor_phone_number))
    THEN
        IF EXISTS
            (SELECT * FROM customer_to_customer
            WHERE initiator_phone_number = new_initiator_phone_number
            AND interlocutor_phone_number = new_interlocutor_phone_number)
        THEN
```

```

        SELECT
get_customer_to_customer_id(new_initiator_phone_number,
new_interlocutor_phone_number) INTO new_id;

        CALL insert_message(new_message_content,
new_message_date_time, new_id, NULL);
    ELSE
        INSERT INTO customer_to_customer (initiator_phone_number,
interlocutor_phone_number)
VALUES (new_initiator_phone_number,
new_interlocutor_phone_number);

        SELECT
get_customer_to_customer_id(new_initiator_phone_number,
new_interlocutor_phone_number) INTO new_id;

        CALL insert_message(new_message_content,
new_message_date_time, new_id, NULL);
    END IF;
    ELSIF (SELECT is_our_client(new_initiator_phone_number))
OR (SELECT is_our_client(new_interlocutor_phone_number)) THEN

        IF (SELECT is_our_client(new_initiator_phone_number)) THEN
            IF EXISTS (SELECT * FROM customer_to_other
WHERE customer_phone_number = new_initiator_phone_number AND
other_phone_number = new_interlocutor_phone_number AND
customer_is_initiator = TRUE) THEN
                SELECT get_customer_to_other_id
(new_initiator_phone_number,
new_interlocutor_phone_number,
TRUE) INTO new_id;

                CALL insert_message(new_message_content,
new_message_date_time, NULL, new_id);
            ELSE
                INSERT INTO customer_to_other (customer_is_initiator,
other_phone_number, customer_phone_number)
VALUES (TRUE, new_interlocutor_phone_number,
new_initiator_phone_number);

                SELECT get_customer_to_other_id
(new_initiator_phone_number,
new_interlocutor_phone_number,
TRUE) INTO new_id;

                CALL insert_message(new_message_content,
new_message_date_time, NULL, new_id);
            END IF;
        ELSE
            IF EXISTS (SELECT * FROM customer_to_other
WHERE customer_phone_number = new_interlocutor_phone_number
AND
other_phone_number = new_initiator_phone_number AND
customer_is_initiator = FALSE) THEN
                SELECT get_customer_to_other_id
(new_initiator_phone_number,
new_interlocutor_phone_number,
FALSE) INTO new_id;

                CALL insert_message(new_message_content,
new_message_date_time, NULL, new_id);
            ELSE
                INSERT INTO customer_to_other (customer_is_initiator,
other_phone_number, customer_phone_number)
VALUES (FALSE, new_initiator_phone_number,
new_interlocutor_phone_number);

                SELECT get_customer_to_other_id
(new_initiator_phone_number,
new_interlocutor_phone_number,
FALSE) INTO new_id;

```

```

                                CALL insert_message(new_message_content,
new_message_date_time, NULL, new_id);
                                END IF;
                                END IF;
                                END IF;
END;
$$;

```

## 7) Триггерная функция и триггер, который добавляет бонусные 500 рублей на баланс, если клиент сделал sim-карту в определённый период

```

CREATE OR REPLACE FUNCTION add_to_balance()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
BEGIN
    IF (NEW.balance - OLD.balance >= 2000) THEN
        UPDATE sim_cards SET balance = NEW.balance + 500
        WHERE phone_number = OLD.phone_number;
    END IF;
    RETURN NEW;
END;
$$;

CREATE OR REPLACE TRIGGER add_bonus
AFTER UPDATE ON sim_cards
FOR EACH ROW
EXECUTE FUNCTION add_to_balance();

```

## 8) Триггерная функция и триггер, который генерирует новый номер телефона при создании новой sim-карты

```

CREATE OR REPLACE FUNCTION genetate_phone_number()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
DECLARE
    gen_phone_number VARCHAR(50);
BEGIN
    LOOP
        gen_phone_number := '8' || FLOOR(RANDOM() * 89999 + 10000)::TEXT
        || FLOOR(RANDOM() * 89999 + 10000)::TEXT;
        IF NOT EXISTS (SELECT * FROM sim_cards WHERE phone_number =
gen_phone_number) THEN
            NEW.phone_number := gen_phone_number;
            EXIT;
        END IF;
    END LOOP;
    RETURN NEW;
END;
$$;

CREATE OR REPLACE TRIGGER new_sim_card_with_generate_phone_number
BEFORE INSERT ON sim_cards
FOR EACH ROW
EXECUTE FUNCTION genetate_phone_number();

```

Результаты выполнения работы:

```
CALL create_new_traffic_plan('HomeMy', 499.99, 20.0, 700, 50);
SELECT * FROM traffic_plans;
```

	traffic_plan_id [PK] integer	title character varying (50)	monthly_cost real	internet_data_limit real	call_minutes_limit integer	messages_limit integer
1	1	Базовый	300	5000	300	100
2	2	Стандартный	500	10000	600	200
3	3	Премиум	700	15000	1200	300
4	4	Home	499.99	20	700	50
5	5	HomeMy	499.99	20	700	50

Рисунок 1 Добавление тарифного плана

```
CALL new_message('Content', '2024-10-06 14:49:23', '89007654321', '89229060764');
CALL new_message('Contentdadsa', '2024-10-06 14:49:23', '89229060764', '89007654321');
CALL new_message('Content to other', '2024-10-06 14:49:23', '89007654321', '89001234567');
CALL new_message('Content to cust', '2024-10-06 14:49:23', '89001234567', '89007654321');
SELECT * FROM messages;
```

	traffic_plan_id [PK] integer	title character varying (50)	monthly_cost real	internet_data_limit real	call_minutes_limit integer	messages_limit integer
1	1	Базовый	300	5000	300	100
2	2	Стандартный	500	10000	600	200
3	3	Премиум	700	15000	1200	300
4	4	Home	499.99	20	700	50
5	5	HomeMy	499.99	20	700	50

Рисунок 2 Добавление сообщений

```
UPDATE sim_cards SET balance = 6500 WHERE phone_number = '89001234567';
SELECT * from sim_cards;
```

	phone_number [PK] character varying (50)	registration_date date	balance real	traffic_plan_id integer	personal_account_number bigint
1	89007654321	2023-02-01	300	2	1000000002
2	89009876543	2023-03-01	230	3	1000000003
3	87227014735	2023-03-01	230	3	1000000003
4	89001234567	2023-01-01	7000	1	1000000001

Рисунок 3 Бонусные 500 рублей при пополнении от 2000 рублей

```
INSERT INTO sim_cards (registration_date, balance, traffic_plan_id, personal_account_number)
VALUES
('2023-03-01', 234.00, 2, 1000000004);
SELECT * FROM sim_cards;
```

	phone_number [PK] character varying (50)	registration_date date	balance real	traffic_plan_id integer	personal_account_number bigint
1	89007654321	2023-02-01	300	2	1000000002
2	89009876543	2023-03-01	230	3	1000000003
3	87227014735	2023-03-01	230	3	1000000003
4	89001234567	2023-01-01	7000	1	1000000001
5	81692377712	2023-03-01	234	2	1000000003

Рисунок 4 Генерация номера телефона при добавлении новой sim-карты

```
C:\Windows\System32\cmd.e x + v
C:\Program Files\PostgreSQL\16\bin>psql -U postgres
Password for user postgres:

psql (16.4)
Type "help" for help.

postgres=# CREATE USER test_user WITH PASSWORD '123';
CREATE ROLE
postgres=# CREATE DATABASE test_user;
CREATE DATABASE
postgres=# \q

C:\Program Files\PostgreSQL\16\bin>psql -U test_user;
Password for user test_user;

psql: error: connection to server at "localhost" (::1), port 5432 failed: ВАЖНО: пользователь "test_user;" не прошёл пр
оверку подлинности (по паролю)

C:\Program Files\PostgreSQL\16\bin>psql -U test_user
Password for user test_user:

psql (16.4)
Type "help" for help.

test_user=> |
```

Рисунок 5 Создание пользователя

```
C:\Windows\System32\cmd.e x + v

C:\Program Files\PostgreSQL\16\bin>psql -U postgres
Password for user postgres:

psql (16.4)
Type "help" for help.

postgres=# \connect operator_company
You are now connected to database "operator_company" as user "postgres".
operator_company=# GRANT SELECT ON customers TO test_user;
GRANT
operator_company=# |
```

Рисунок 6 Добавление привилегий на запрос SELECT

```
C:\Windows\System32\cmd.e x + v

C:\Program Files\PostgreSQL\16\bin>psql -U test_user
Password for user test_user:

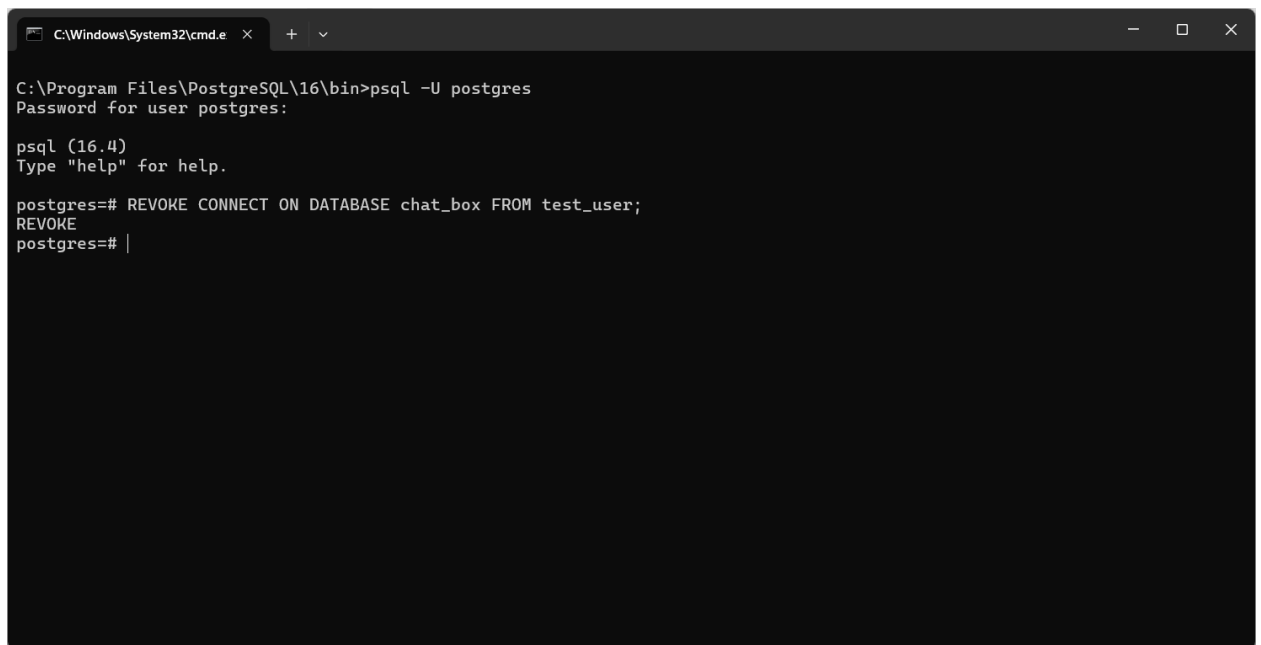
psql (16.4)
Type "help" for help.

test_user=> \connect operator_company
You are now connected to database "operator_company" as user "test_user".
operator_company=> SELECT * FROM customers;
 personal_account_number | last_name | first_name | patronymic | email | birth_date | hashed_password
-----+-----+-----+-----+-----+-----+-----
1000000002 | Петров | Петр | Петрович | petrov@example.com | 1985-05-20 | hashed_password_2
1000000003 | Сидорова | Анна | Сергеевна | sidorova@example.com | 1992-03-10 | hashed_password_3
1000000001 | Губин | Иван | Иванович | ivanov@example.com | 1990-01-15 | hashed_password_1
(3 rows)

operator_company=> UPDATE customers SET email = "ivanov@mail.com" where first_name = 'Иван';
ОШИБКА: столбец "ivanov@mail.com" не существует
LINE 1: UPDATE customers SET email = "ivanov@mail.com" where first_n...
^
operator_company=> UPDATE customers SET email = 'ivanov@mail.com' where first_name = 'Иван';
ОШИБКА: нет доступа к таблице customers
operator_company=> |
```

Рисунок 7 Попытка UPDATE test\_user





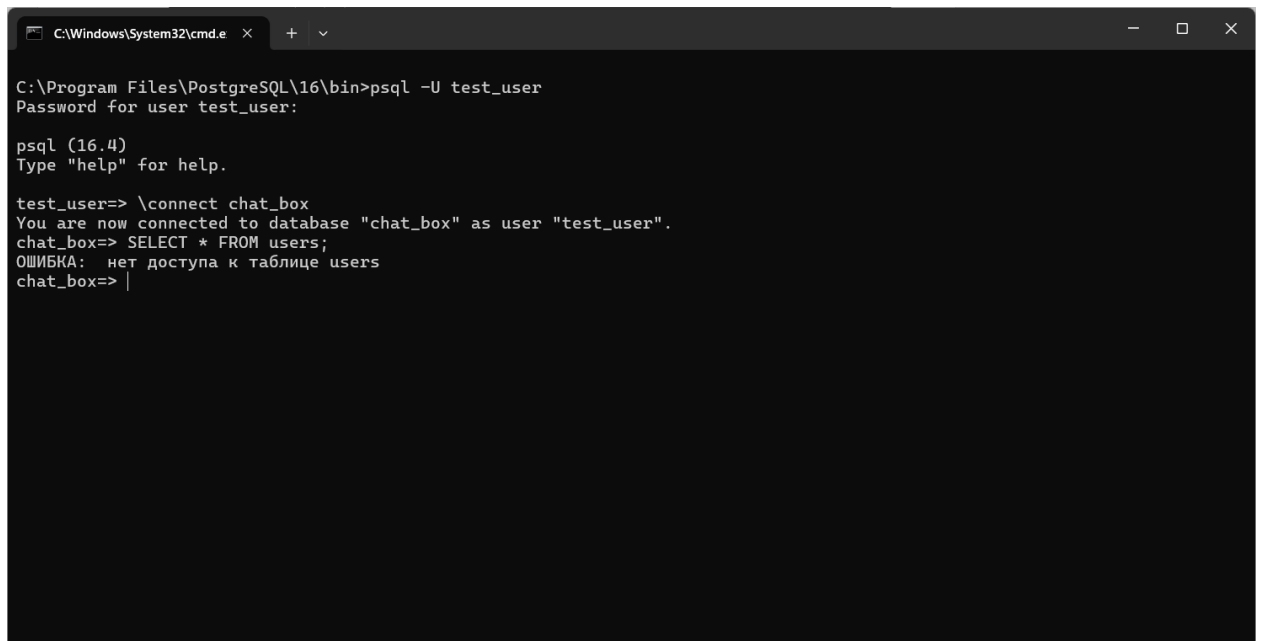
```
C:\Windows\System32\cmd.e x + v

C:\Program Files\PostgreSQL\16\bin>psql -U postgres
Password for user postgres:

psql (16.4)
Type "help" for help.

postgres=# REVOKE CONNECT ON DATABASE chat_box FROM test_user;
REVOKE
postgres=# |
```

Рисунок 8 Запрет на подключение к базе данных chat\_box



```
C:\Windows\System32\cmd.e x + v

C:\Program Files\PostgreSQL\16\bin>psql -U test_user
Password for user test_user:

psql (16.4)
Type "help" for help.

test_user=> \connect chat_box
You are now connected to database "chat_box" as user "test_user".
chat_box=> SELECT * FROM users;
ОШИБКА: нет доступа к таблице users
chat_box=> |
```

Рисунок 9 Попытка подключения к chat\_box

```
C:\Windows\System32\cmd.e x + v
C:\Program Files\PostgreSQL\16\bin>psql -U postgres
Password for user postgres:

psql (16.4)
Type "help" for help.

postgres=# \du
               List of roles
Role name | Attributes
-----+-----
postgres | Superuser, Create role, Create DB, Replication, Bypass RLS
test_user |

postgres=# ALTER USER test_user WITH CREATEROLE;
ALTER ROLE
postgres=#
```

Рисунок 10 Добавление test\_user на создание ролей

```
C:\Windows\System32\cmd.e x + v
C:\Program Files\PostgreSQL\16\bin>psql -U test_user
Password for user test_user:

psql (16.4)
Type "help" for help.

test_user=> CREATE USER user1;
CREATE ROLE
test_user=>
```

Рисунок 11 Создание ролей test\_user

**Выводы:** в ходе выполнения лабораторной работы были получены навыки и практический опыт написания функций, процедур и триггеров.