|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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;

$$;

1. 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;

$$;

1. 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;

$$;

1. 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;

$$;

1. 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\_other\_id)

VALUES (new\_message\_content, new\_message\_date\_time, new\_customer\_to\_orher\_id);

END IF;

END;

$$;

1. 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;

$$;

1. 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();

1. 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:**

****

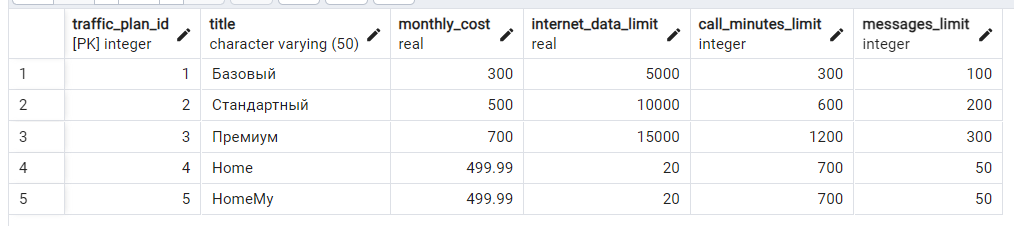
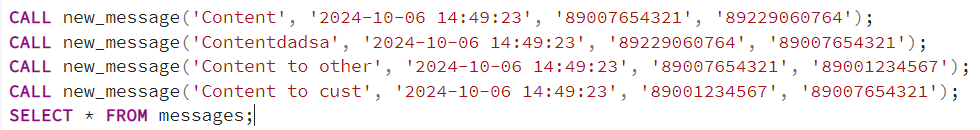
****

Рисунок 1 Добавление тарифного плана

****

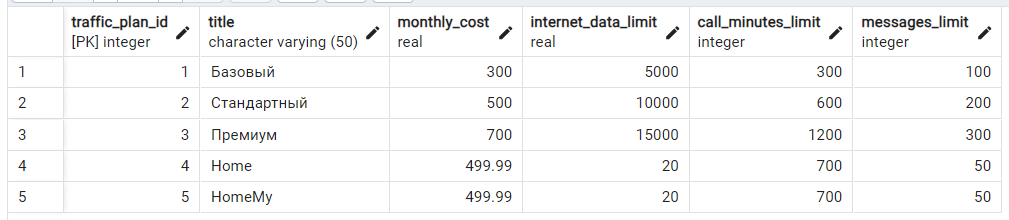
****

Рисунок 2 Добавление сообщений

****

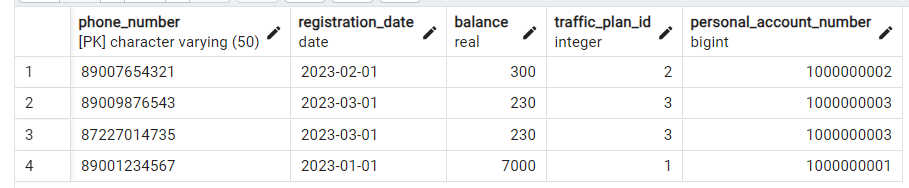
****

Рисунок 3 Бонусные 500 рублей при пополнении от 2000 рублей

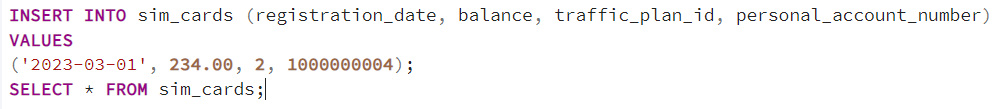
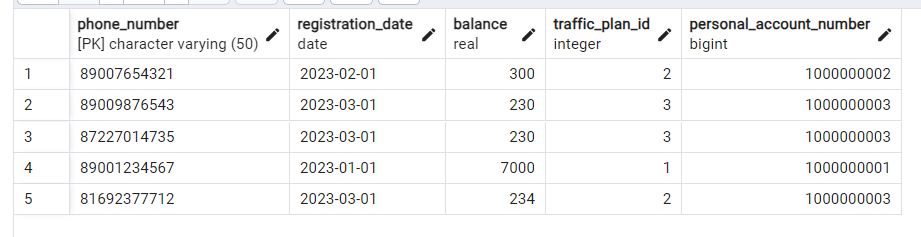
**** ****

Рисунок 4 Генерация номера телефона при добавлении новой sim-карты

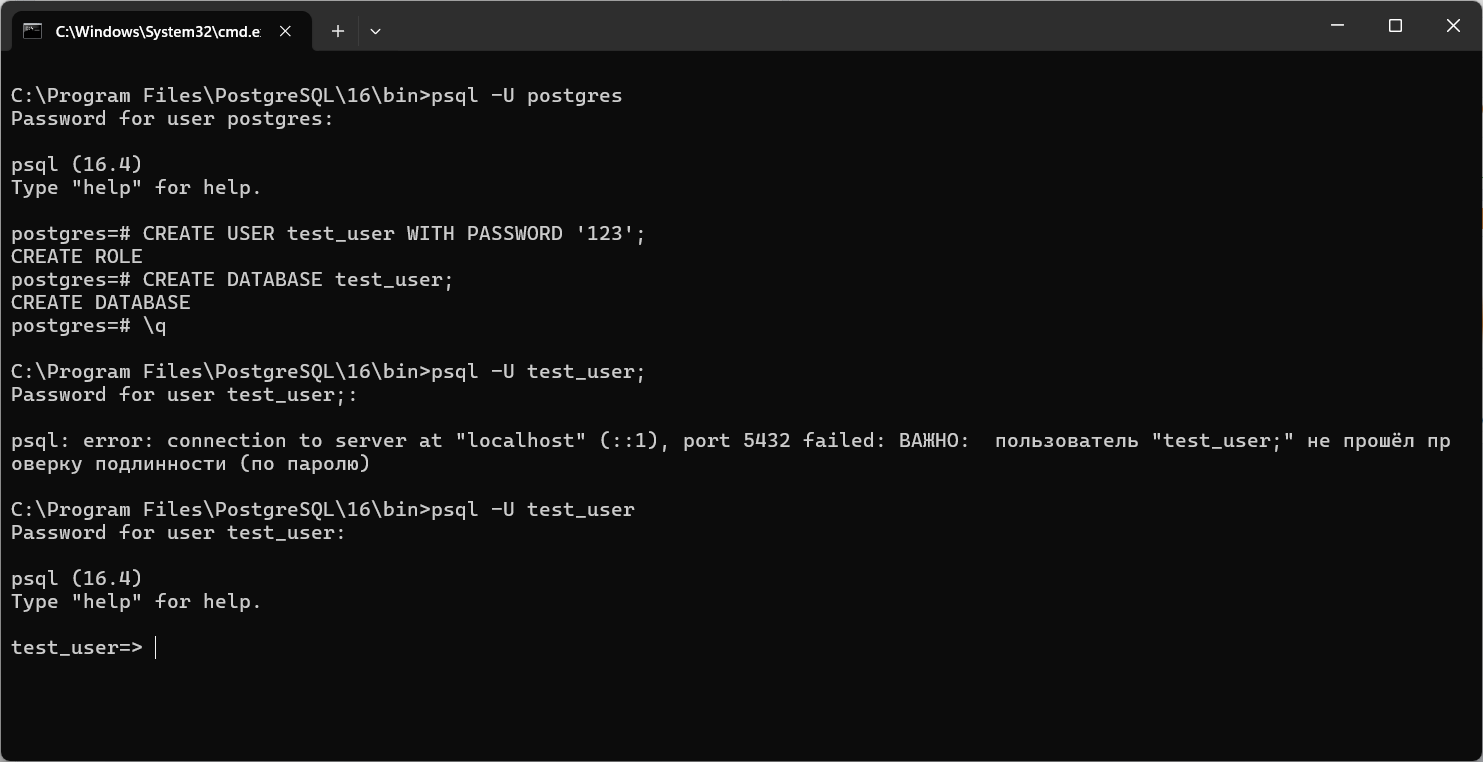


Рисунок 5 Создание пользователя

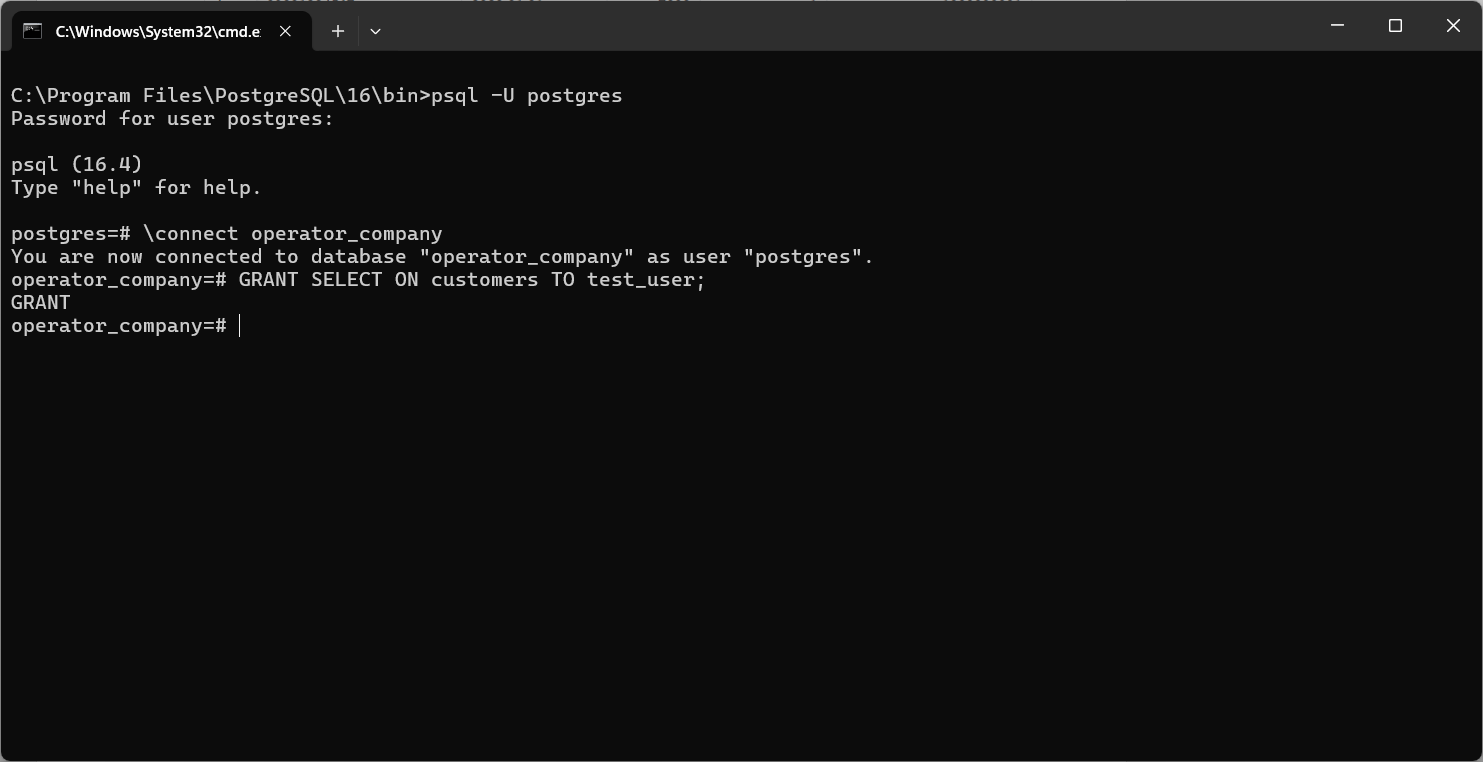


Рисунок 6 Добавление привилегий на запрос SELECT

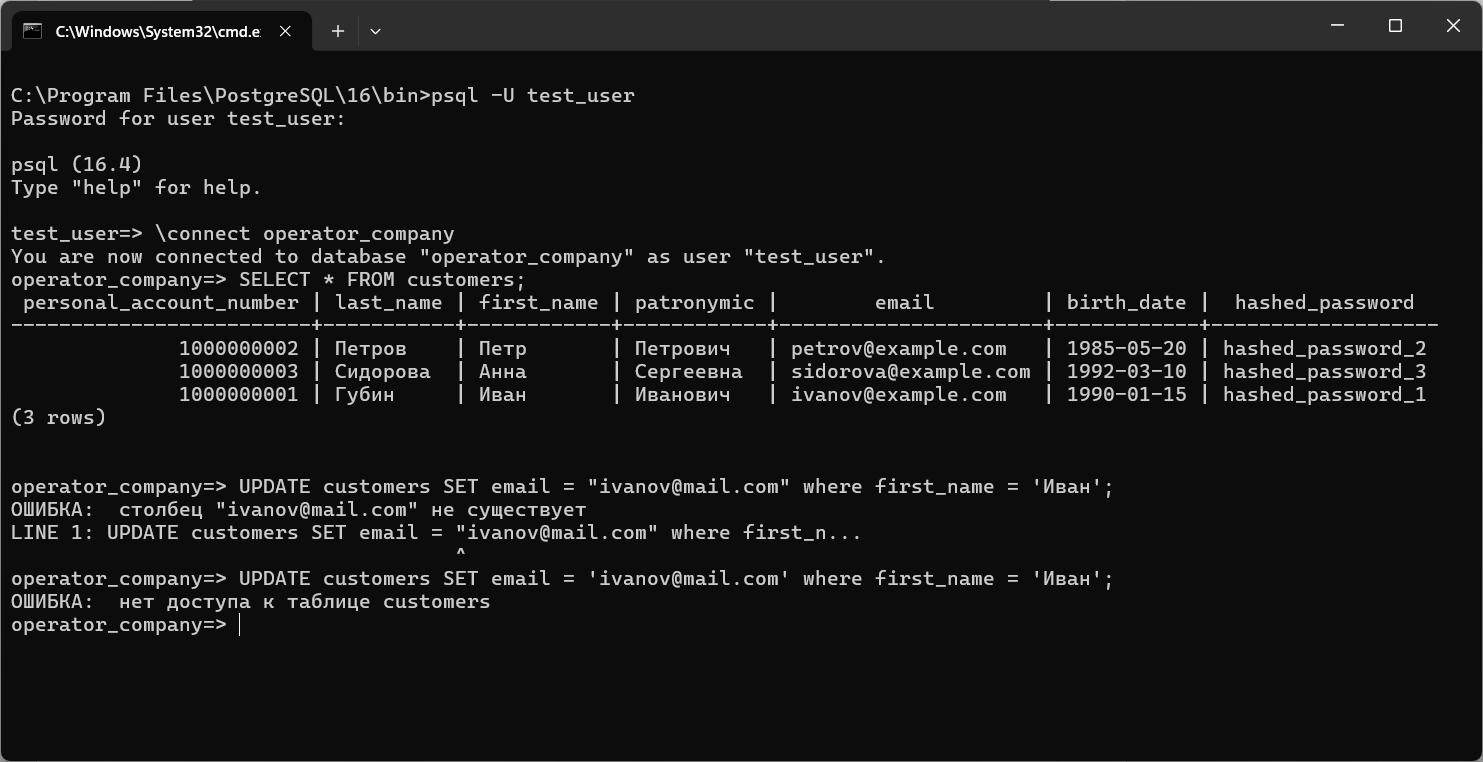


Рисунок 7 Попытка UPDATE test\_user

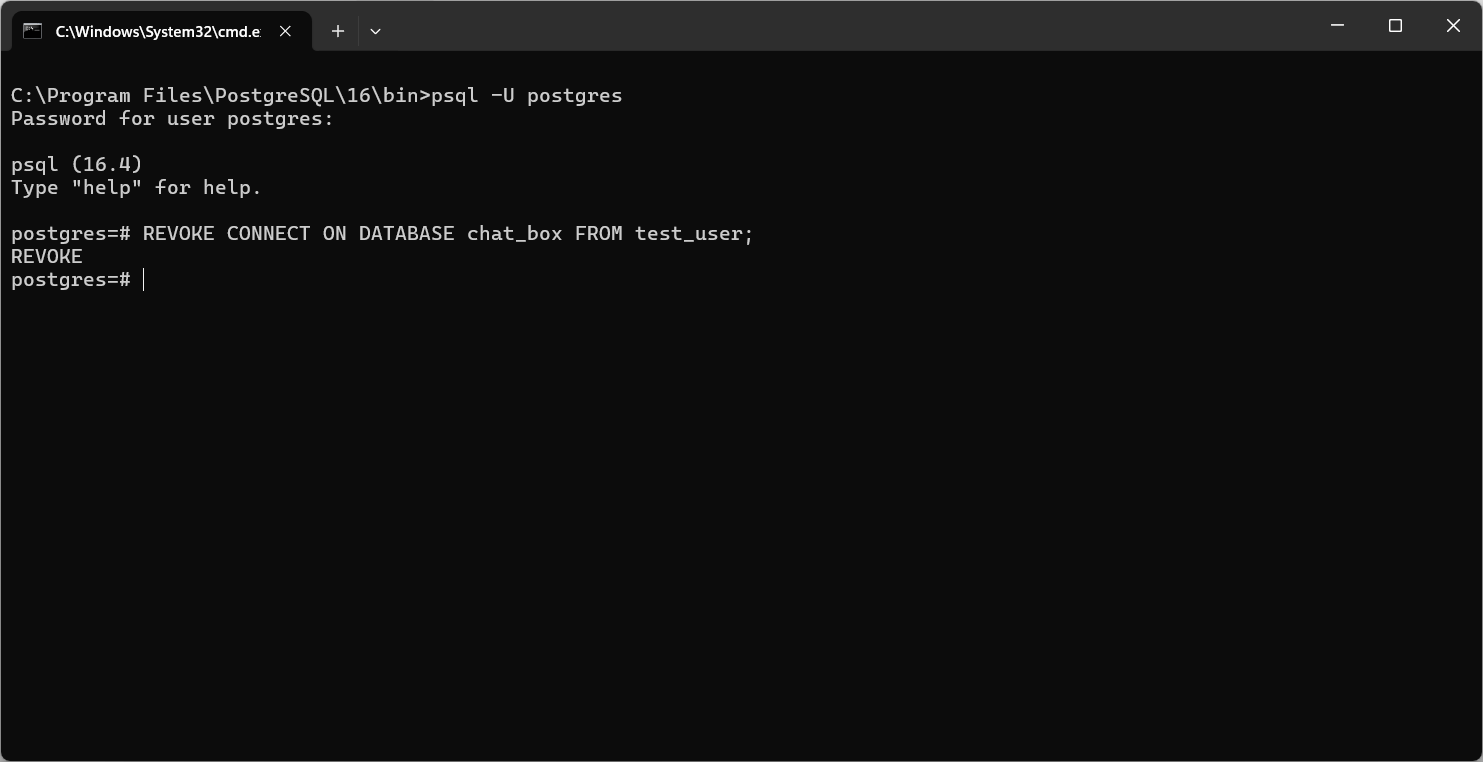


Рисунок 8 Запрет на подключение к базе данных chat\_box

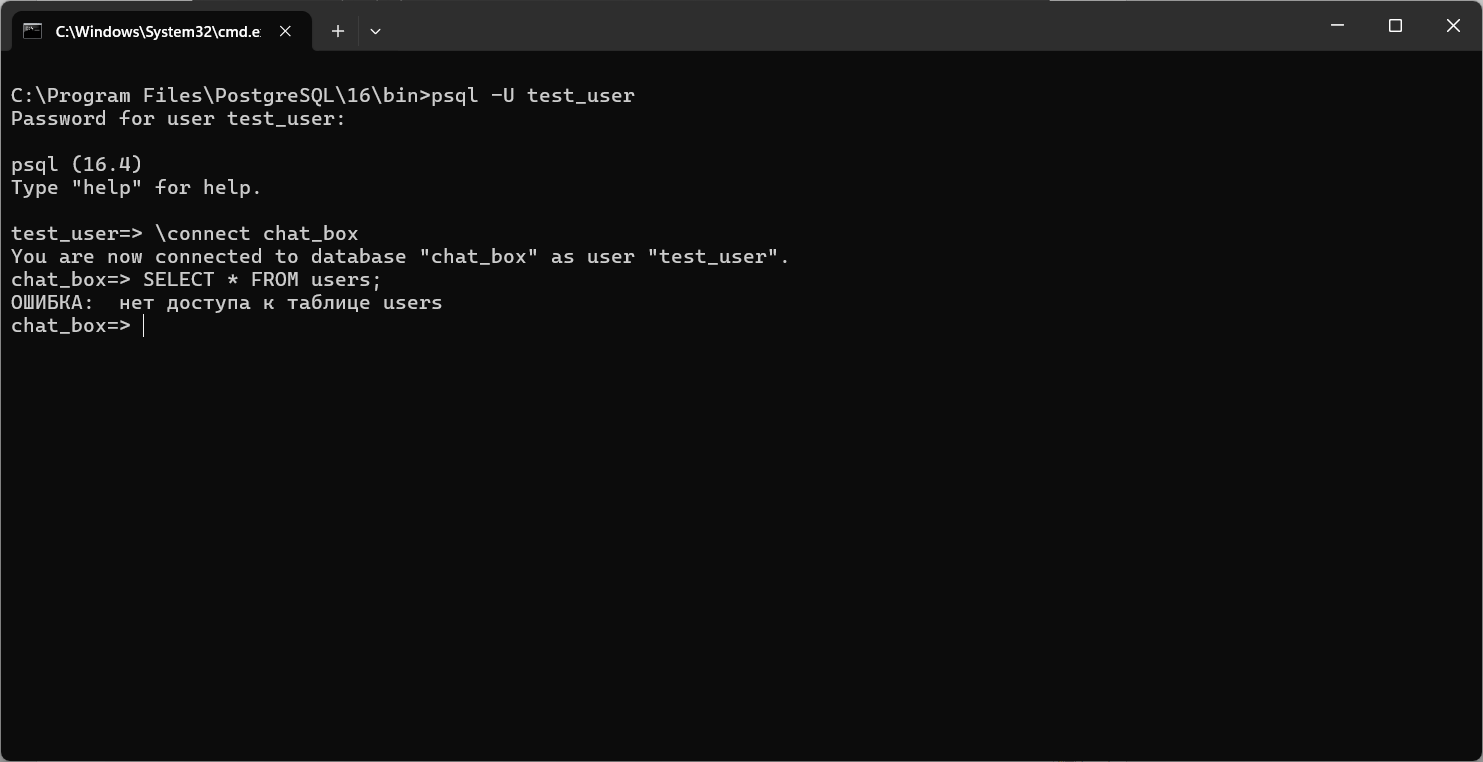


Рисунок 9 Попытка подключения к chat\_box

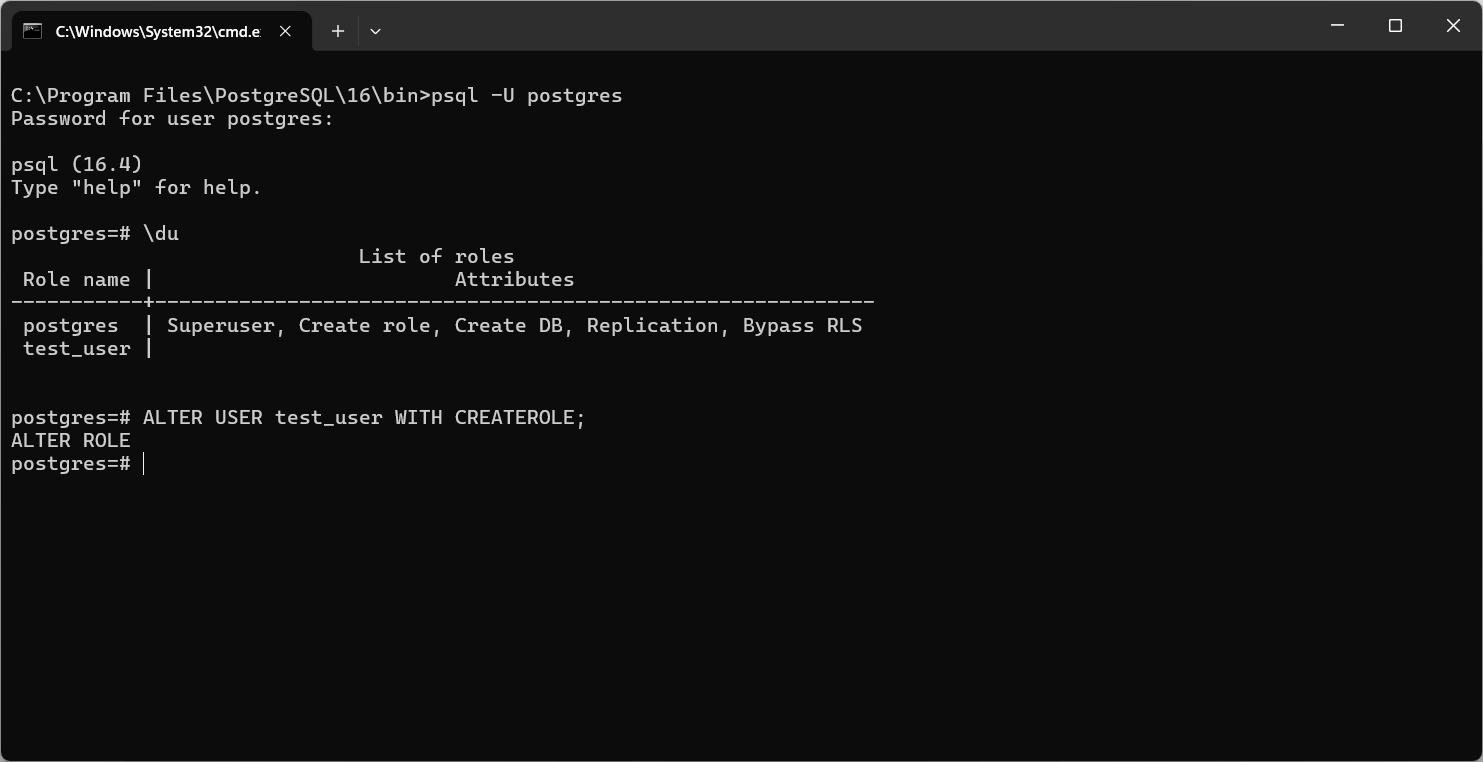


Рисунок 10 Добавление test\_user на создание ролей

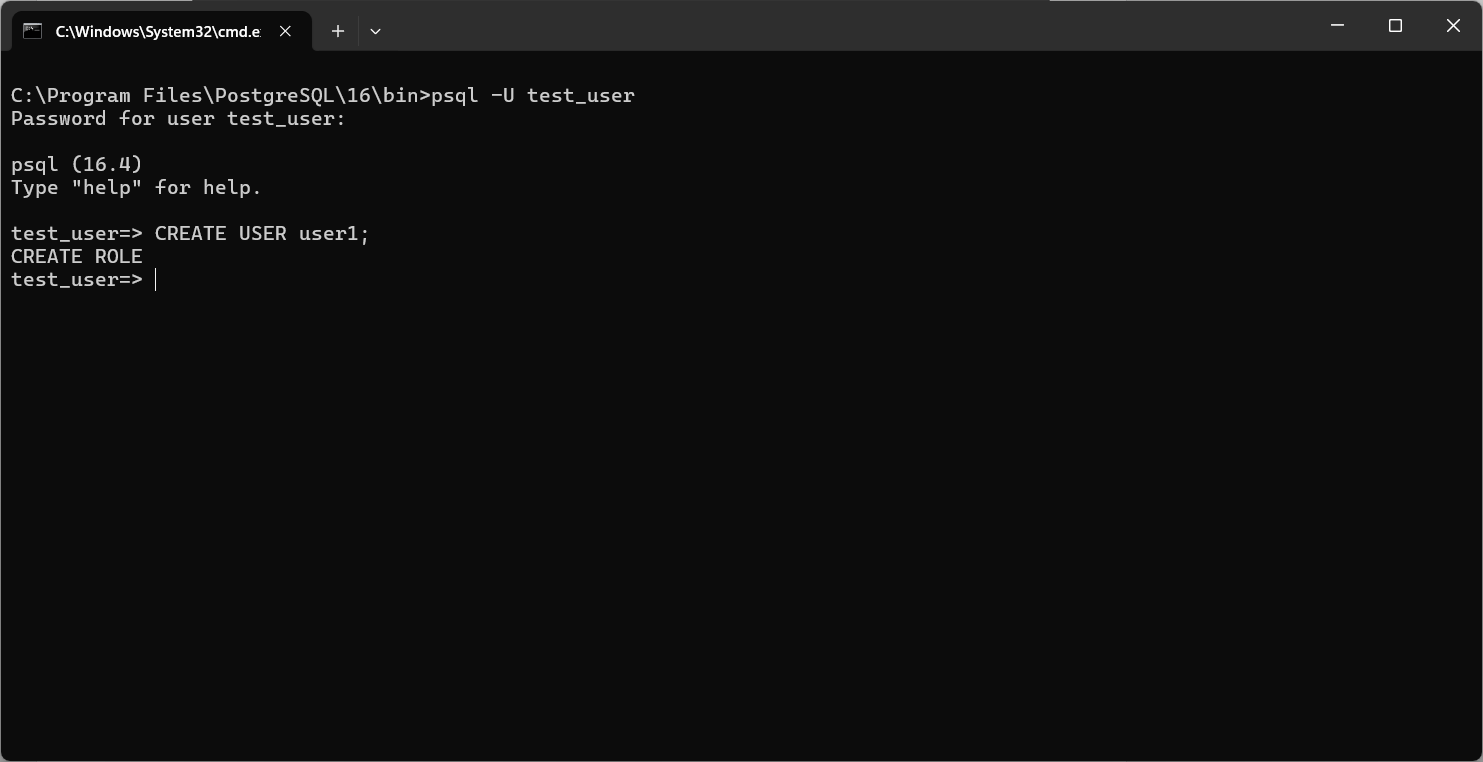


Рисунок 11 Создание ролей test\_user

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