|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 4 |

**Название:**

Массивы, курсоры, триггеры, роли

**Дисциплина:** Базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35 Б |  |  | Дулина И.А. |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Кудрявцев А.П. |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Вариант 8**

**Цель:**

Данная лабораторная работа призвана сформировать у студента понимание назначения массивов, курсоров, триггеров и ролей, их написание и использование.

**Задачи:**

* Ознакомиться с использованием массивов.
* Научиться (изменять\добавлять\удалять) данные в массиве с помощью встроенных операций.
* Получить знания о курсорах и научиться использовать курсоры.
* Узнать о ролях и пользователях.
* Научиться пользоваться командами GRANT и REVOKE для того, чтобы (предоставлять\отзывать) доступ к данным.

**Задание:**

Написать и протестировать триггеры, выполняющие следующие действия для своей предметной области. (Проекты – Поручения – Работник)

* Контроль соответствия дат выдачи, плановой даты окончания и реальной даты окончания *поручения*.
* Контроль дублирования *проекта*.
* Запрет на удаления *работника*, если у него есть *поручение*.
* Создать таблицу, состоящую из двух целочисленных полей и содержащую одну запись, для хранения количества проектов и работников.

Написать триггеры для таблиц *проекты* и *работники*, подсчитывающие при добавлении и удалении общее количество проектов и работников и, сохраняющие итоги в созданной таблице.

* Создать пользователя test и выдать ему доступ к базе данных.
* Составить и выполнить скрипты присвоения новому пользователю прав доступа к таблицам, созданным в практическом задании 1. При этом права доступа к различным таблицам должны быть различными, а именно:
* По крайней мере, для одной таблицы новому пользователю присваиваются права SELECT, INSERT, UPDATE в полном объеме.
* По крайней мере, для одной таблицы новому пользователю присваиваются права SELECT и UPDATE только избранных столбцов.
* По крайней мере, для одной таблицы новому пользователю присваивается только право SELECT.

**Задание 1**

Для таблицы assignments напишем триггер, который не даст завести поручение с датой выдачи больше плановой даты окончания, большей реальной датой окончания

1. триггер должен быть установлен для операций INSERT и UPDATE, при чем для UPDATE надо контролировать изменение только столбцов handout\_date, planned\_end\_date, real\_end\_date.
2. триггер должен быть построчным.
3. Триггер - BEFORE, что позволяет отказаться от внесении изменения, не затрагивая другие строки таблицы.

**Триггерная функция:**

create or replace function fun\_assign\_date() returns trigger

language plpgsql

as $$

declare

begin

if (new.handout\_date>new.planned\_end\_date)or(new.planned\_end\_date>new.real\_end\_date) then

raise warning 'Несоотвествие дат';

return null;

end if;

return new;

end;

$$;

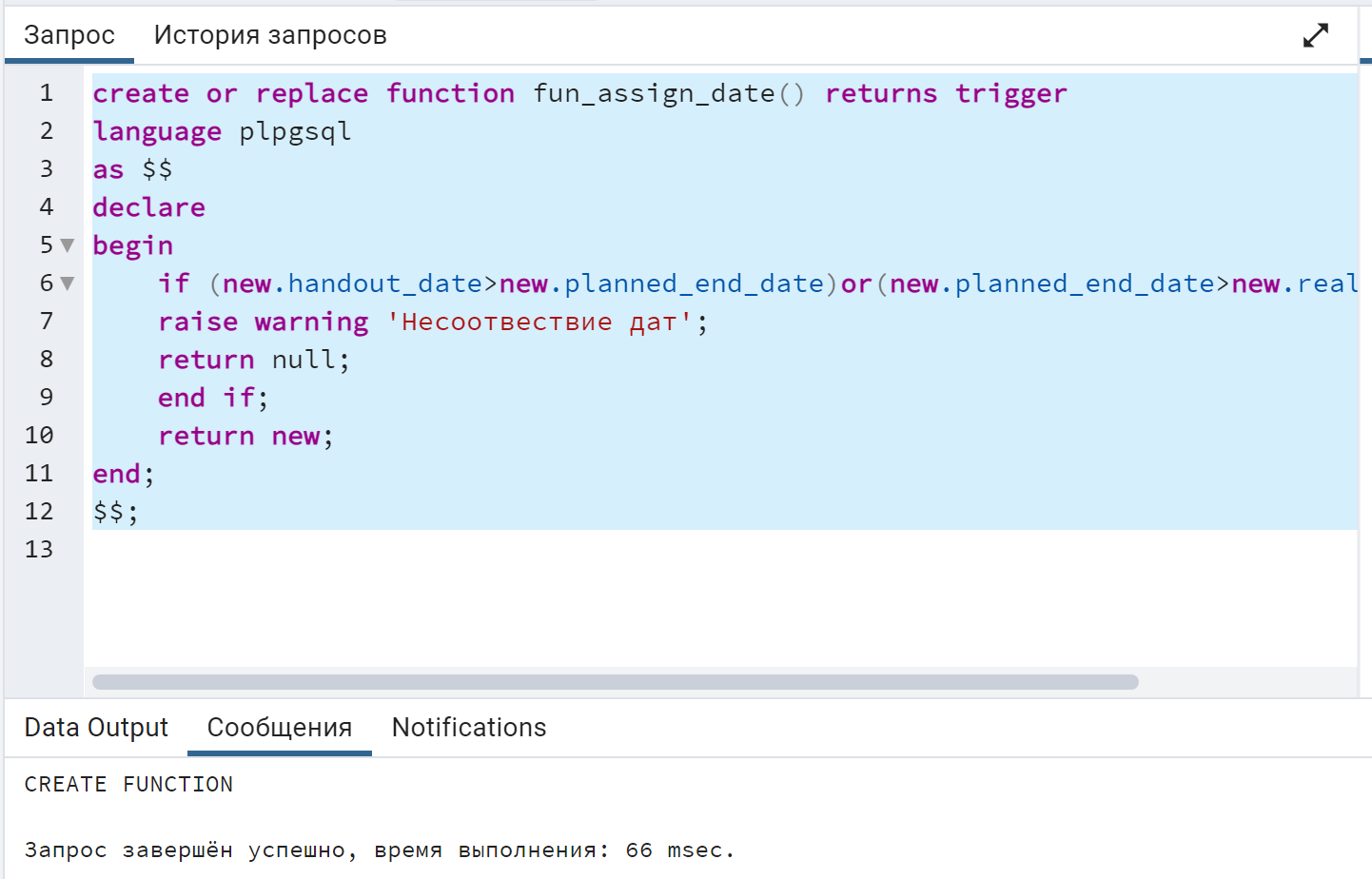


Рисунок 1.1 – создание триггерной функции

**Создание триггера:**

create trigger tr\_assign\_date

before insert or update of handout\_date, planned\_end\_date, real\_end\_date

on assignment

for row

execute function fun\_assign\_date();

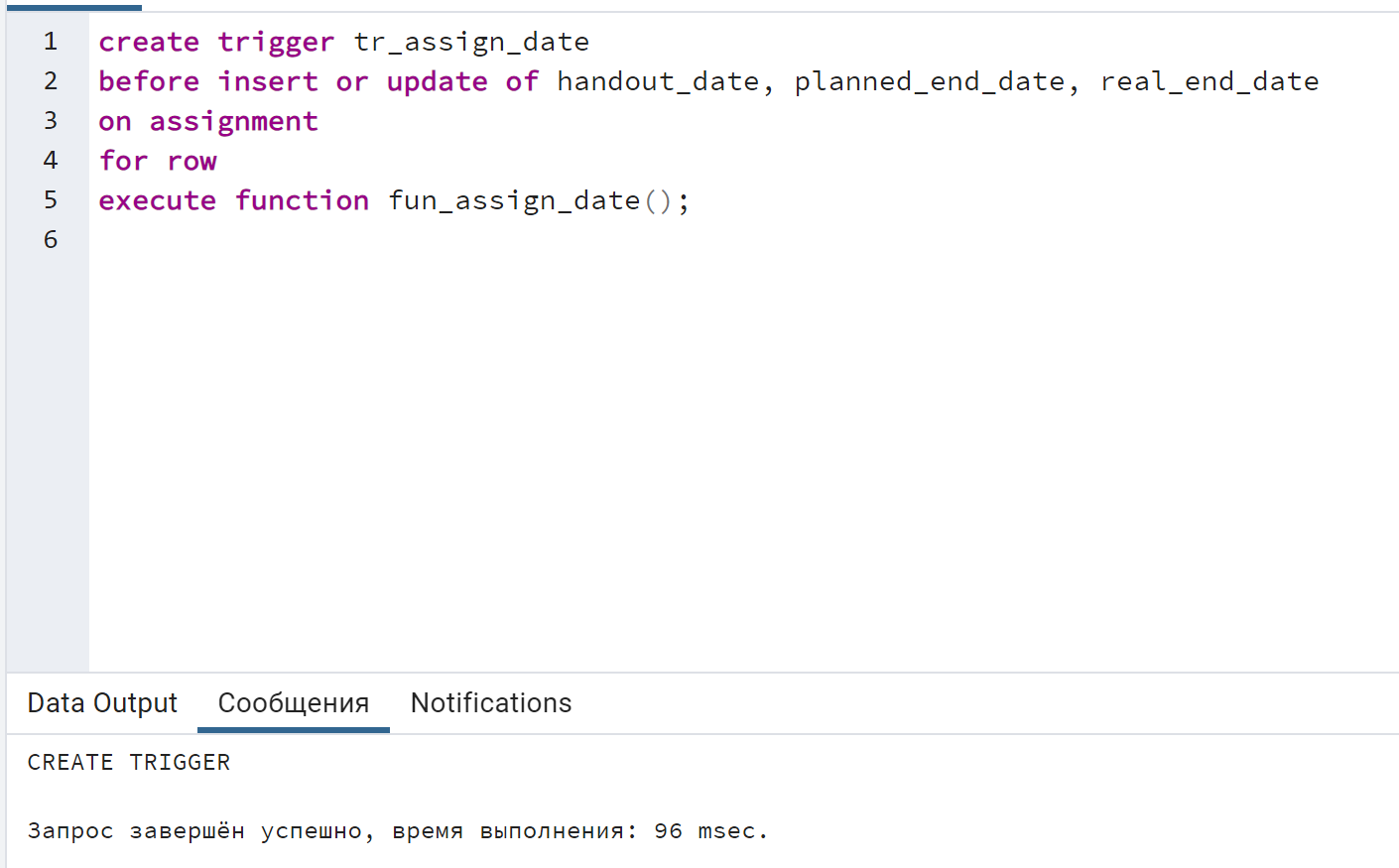


Рисунок 1.2 – создание триггера INSERT/UPDATE для assignment

**Проверка для insert:**

insert into assignment(project\_no, worker\_id,

handout\_date, planned\_end\_date, real\_end\_date,

assignment\_complexity)

values (2, 3, '2022-12-12', '2021-02-02', '2023-09-09', 20),

(2, 3, '2020-03-03', '2022-12-12', '2019-03-03', 15),

(2, 3, '2015-02-02', '2017-04-04', '2022-01-01', 13);

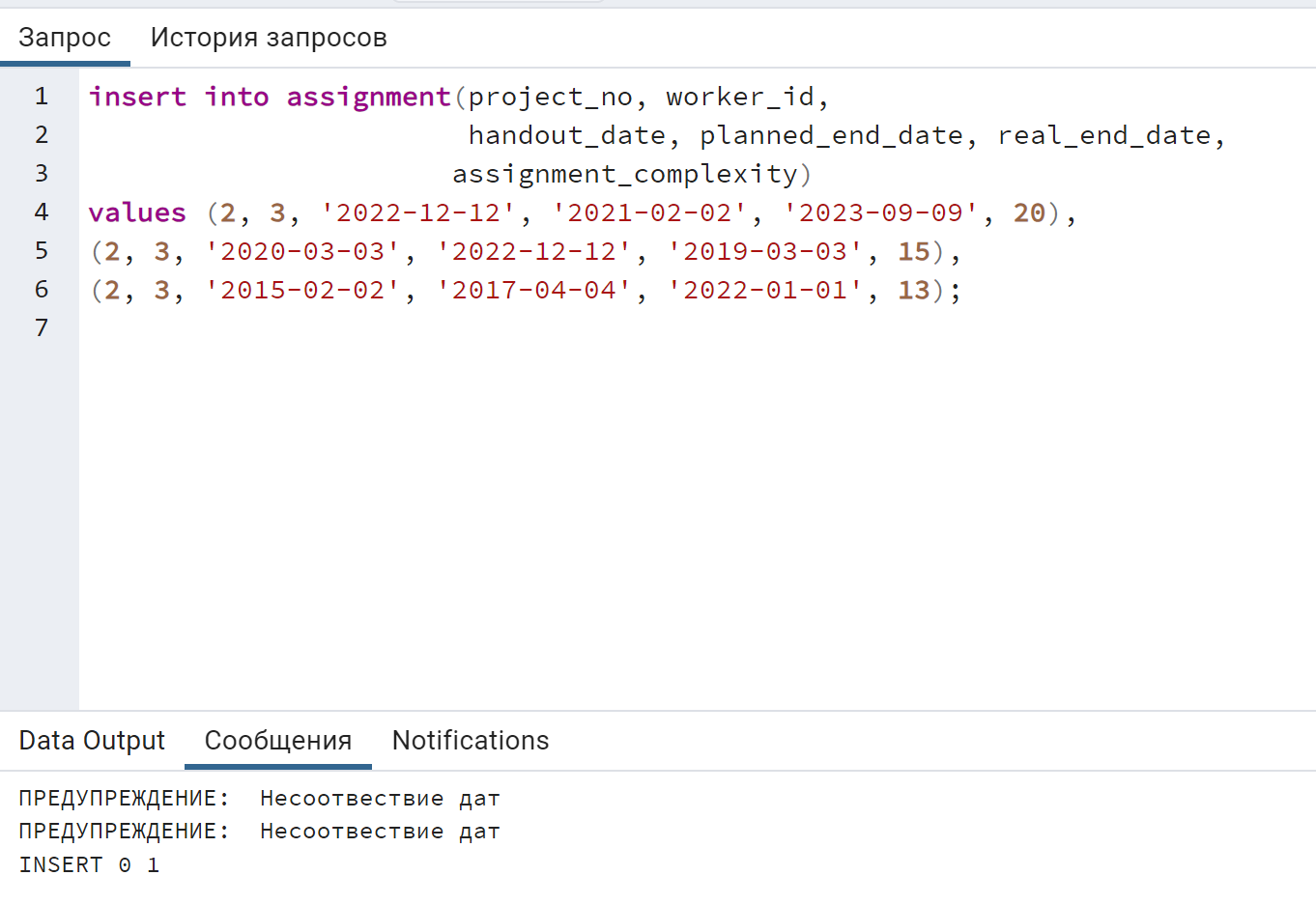


Рисунок 1.3 – проверка для insert

**Проверка для update:**

update assignment a

set handout\_date='2022-12-12',

planned\_end\_date='2021-02-02',

real\_end\_date='2023-09-09'

where a.assignment\_id=1;



Рисунок 1.4 – проверка для update

**Задание 2**

Для таблицы projects напишем триггер, который не даст завести проект, имя которого уже присутствует в списке проектов

1. триггер должен быть установлен для операций INSERT и UPDATE, при чем для UPDATE надо контролировать изменение только столбца project\_name.
2. триггер должен быть построчным.
3. Триггер - BEFORE, что позволяет отказаться от внесении изменения, не затрагивая другие строки таблицы.

**Триггерная функция:**

create or replace function fun\_proj\_name() returns trigger

language plpgsql

as $$

declare

begin

if exists(select 1 from project where new.project\_name=project\_name) then

raise warning 'Проект с таким названием уже существует';

return null;

end if;

return new;

end;

$$;

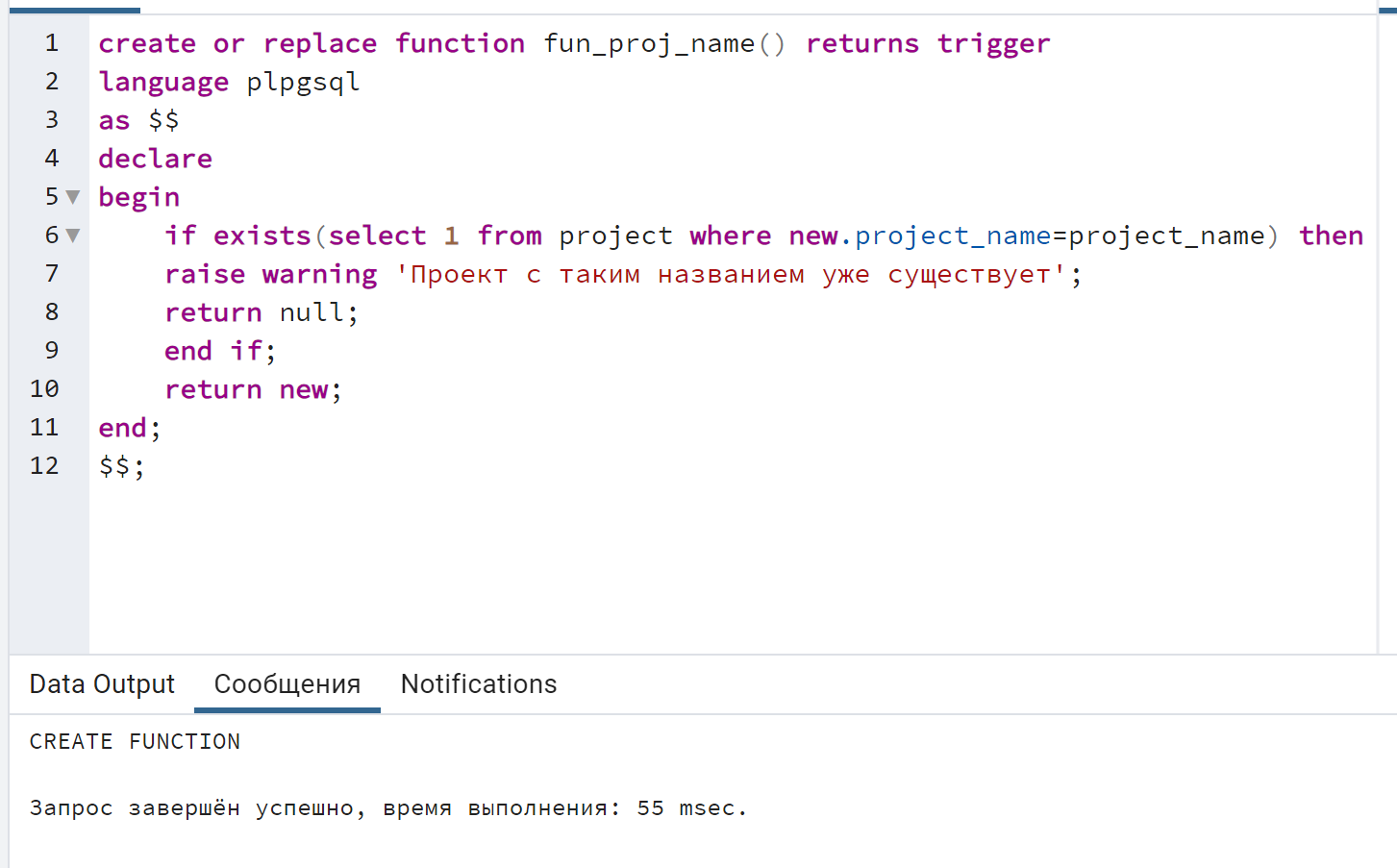


Рисунок 2.1 – создание триггерной функции

**Создание триггера:**

create trigger tr\_proj\_name

before insert or update of project\_name

on project

for row

execute function fun\_proj\_name();

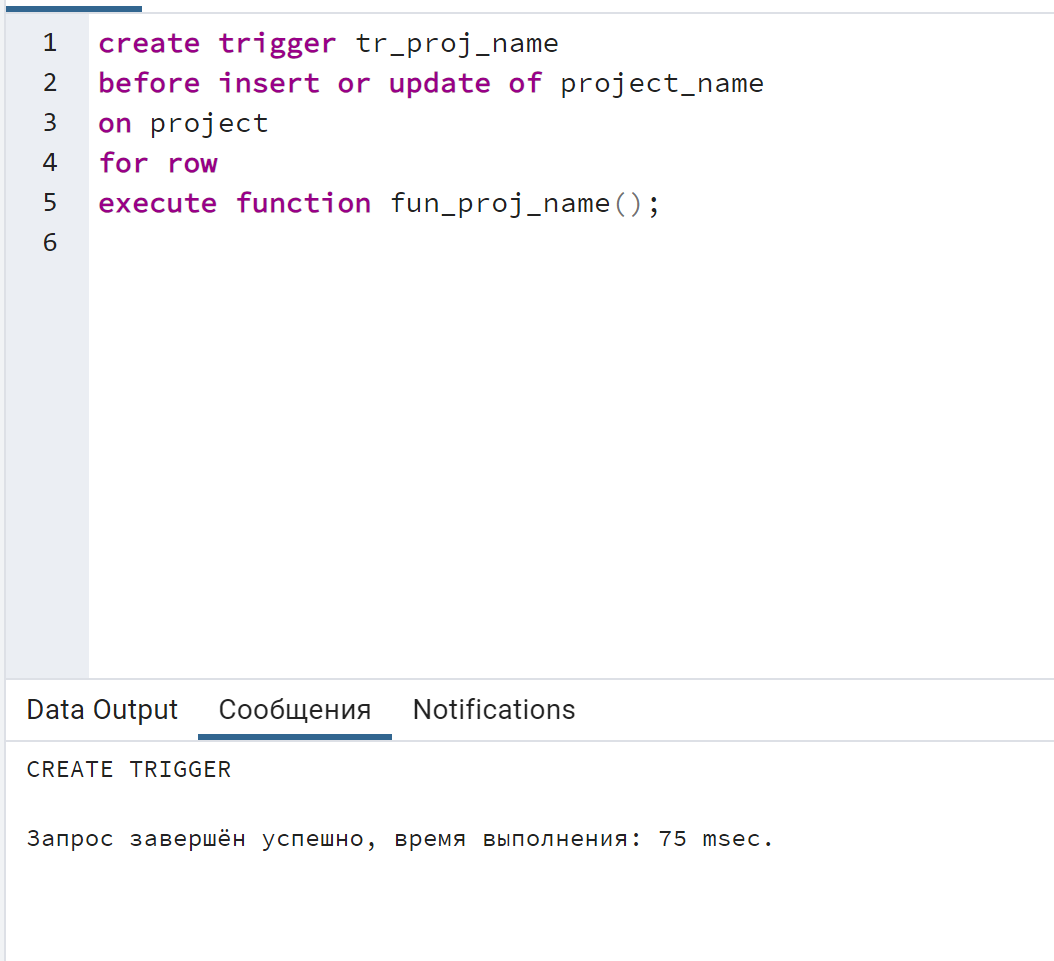


Рисунок 2.2 – создание триггера INSERT/UPDATE для project

**Проверка для insert:**

insert into project (project\_name, project\_complexity, deadline)

values ('Бойцовский клуб', 13, '2020-03-20'),

('Властелин колец', 72, '2010-10-10');

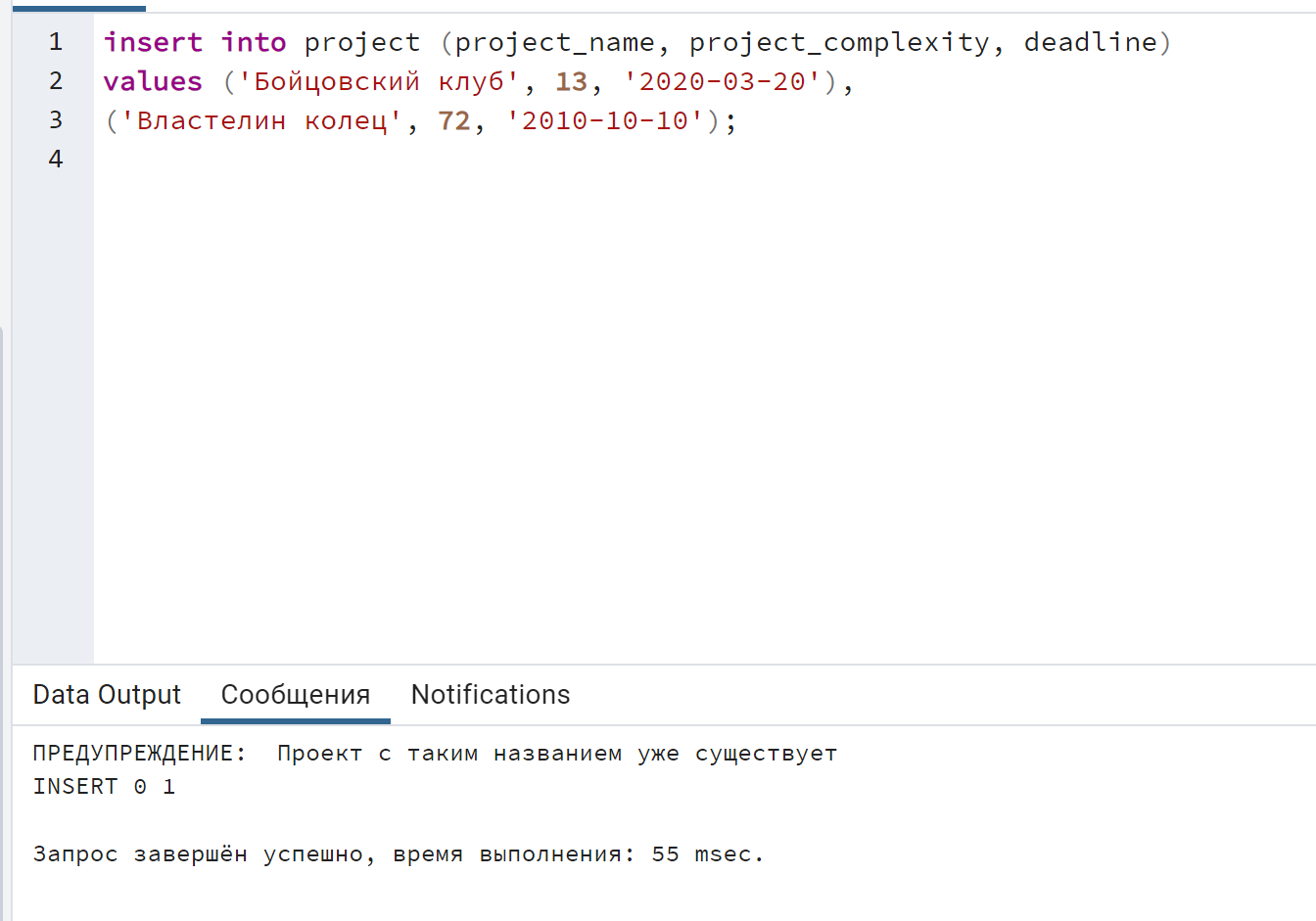


Рисунок 2.3 – проверка для insert

**Проверка для update:**

update project

set project\_name='Тор' where project\_no=10;

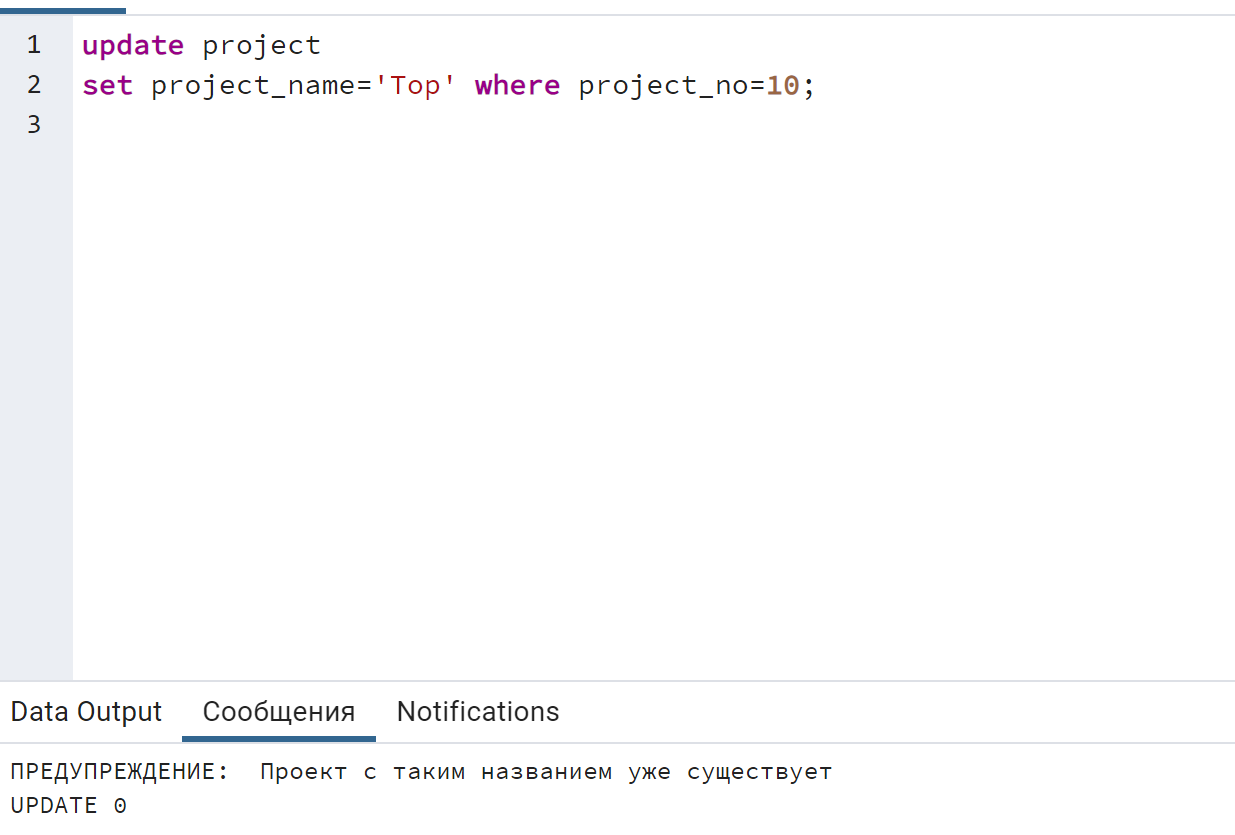


Рисунок 2.4 – проверка для update

**Задание 3**

Запрет на удаления *работника*, если он всё ещё занимается поручением.

Создадим работника Андрея, который будет заниматься поручением:

insert into worker(worker\_name, position)

values ('Андрей', 'продюссер');

select \* from worker;

insert into assignment (project\_no,

worker\_id,

handout\_date,

planned\_end\_date,

real\_end\_date,

assignment\_complexity)

values (1, 16, '2020-12-12', '2021-02-02', '2025-02-02', 34);

**Триггерная функция:**

create or replace function fun\_worker\_del() returns trigger

language plpgsql

as $$

begin

if exists(select 1 from assignment as a

where a.worker\_id=old.worker\_id and

real\_end\_date>'2023-11-12') then

raise warning 'Работник имеет поручение';

return null;

end if;

delete from assignment a where a.worker\_id=old.worker\_id;

raise warning 'успешно';

return old;

end;

$$;

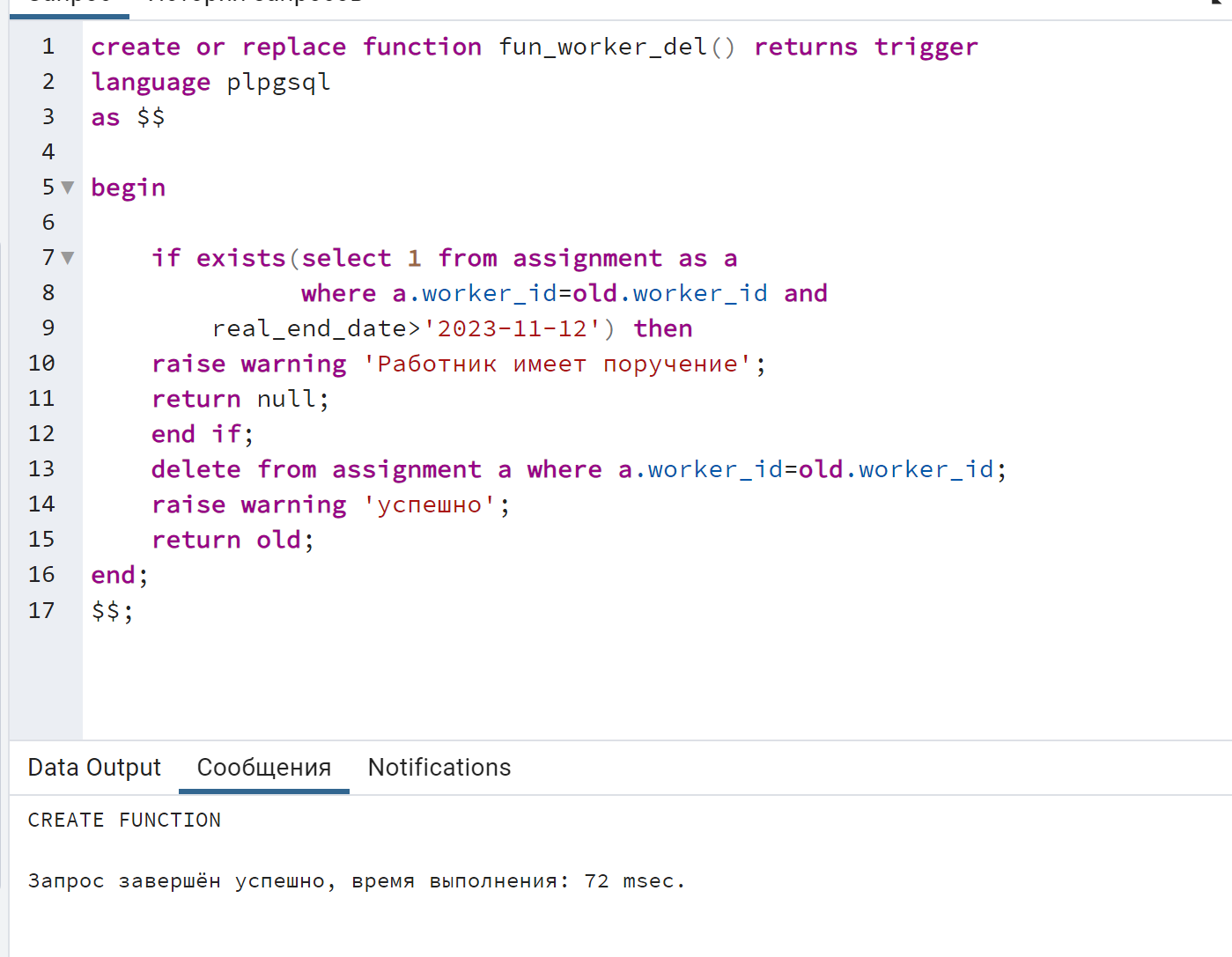


Рисунок 3.1 – создание триггерной функции

**Создание триггера**:

create or replace trigger tr\_work\_del

before delete

on worker

for row

execute function fun\_worker\_del();

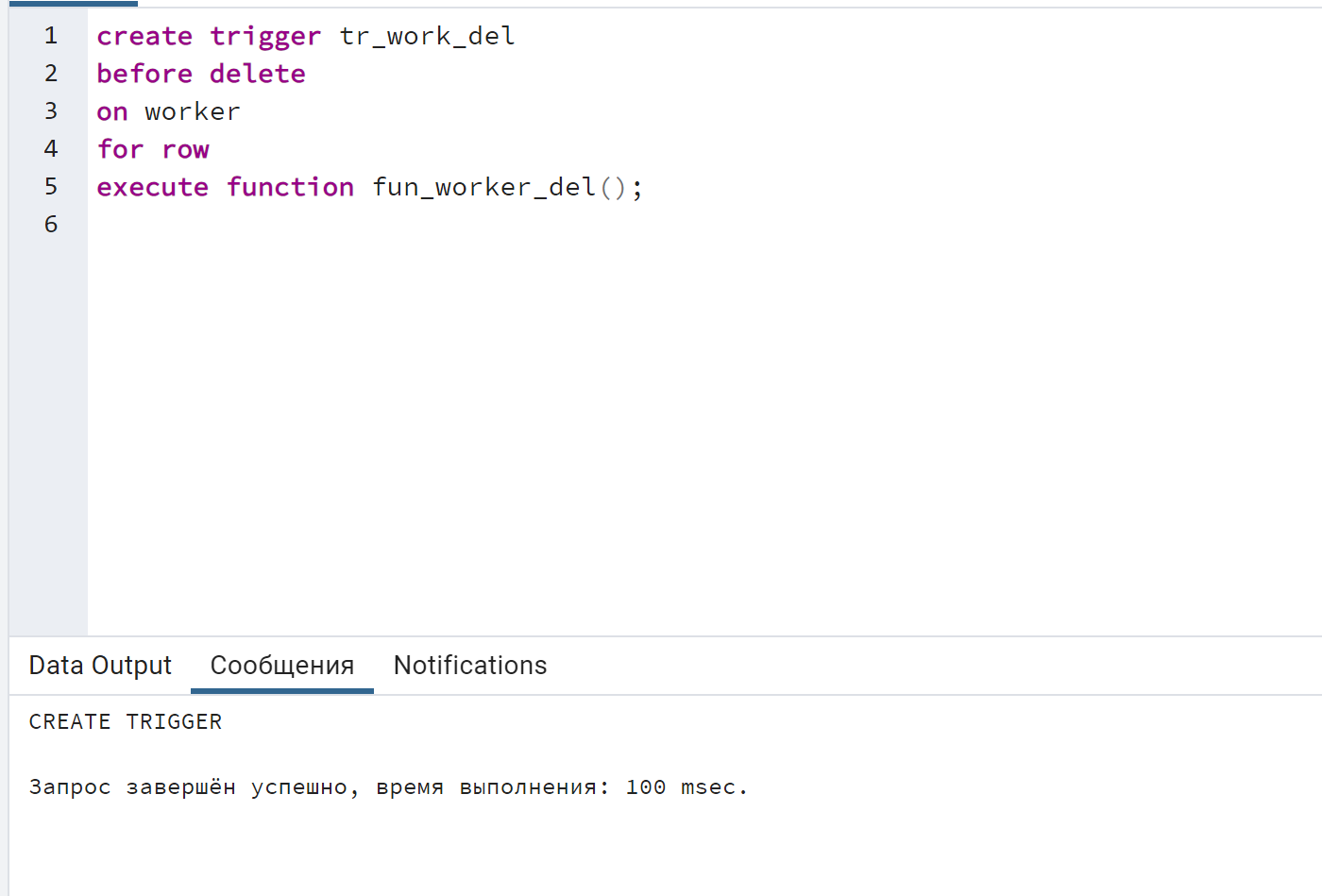


Рисунок 3.2 – создание триггера DELETE для worker

**Проверка для delete**:

delete from worker where worker\_name='Андрей';

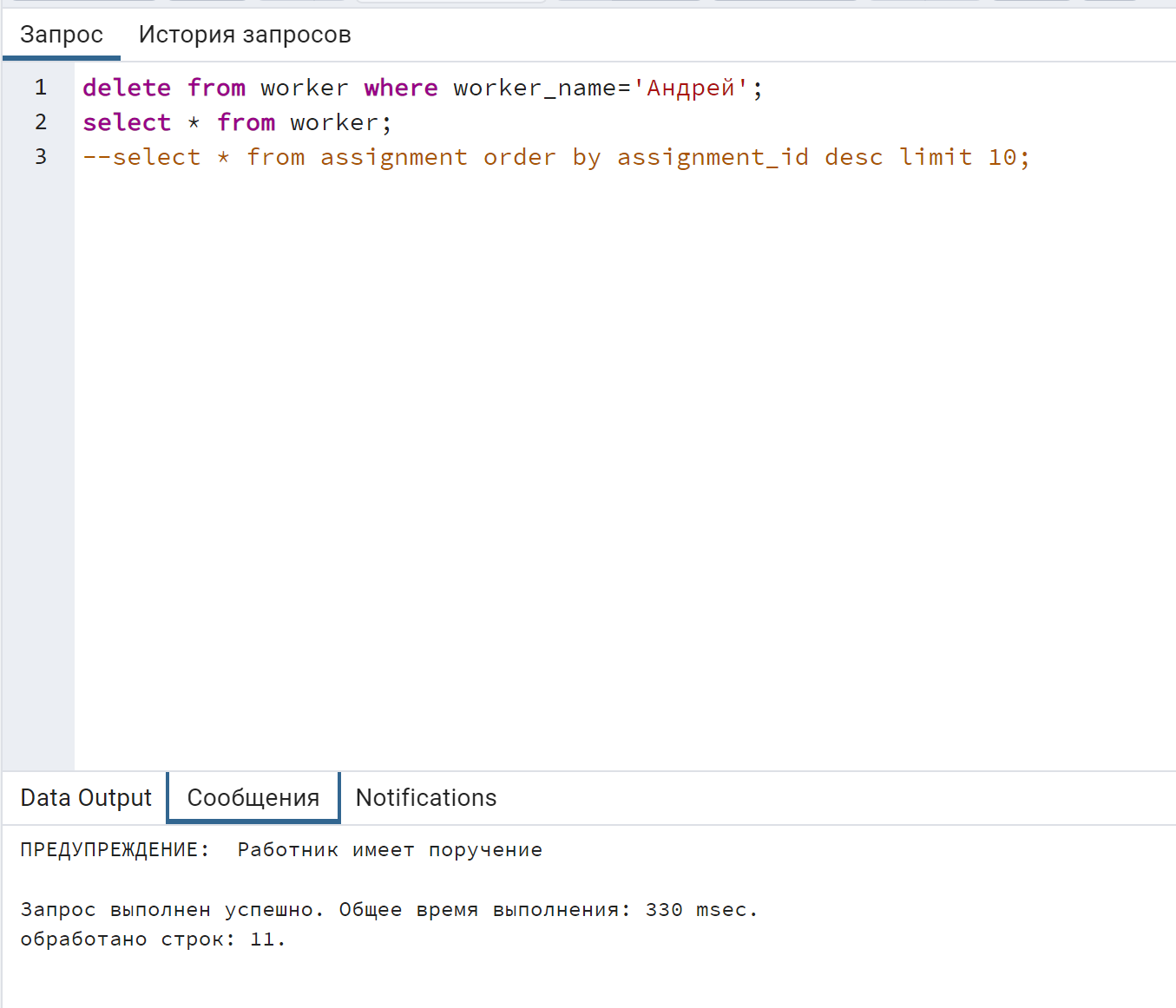


Рисунок 3.3 – проверка на неудачное удаление

Изменим дату завершения поручения, по которой Андрей уже будет свободен и его можно будет удалить:

update assignment

set real\_end\_date='2023-09-09' where worker\_id='16';

delete from worker where worker\_name='Андрей';

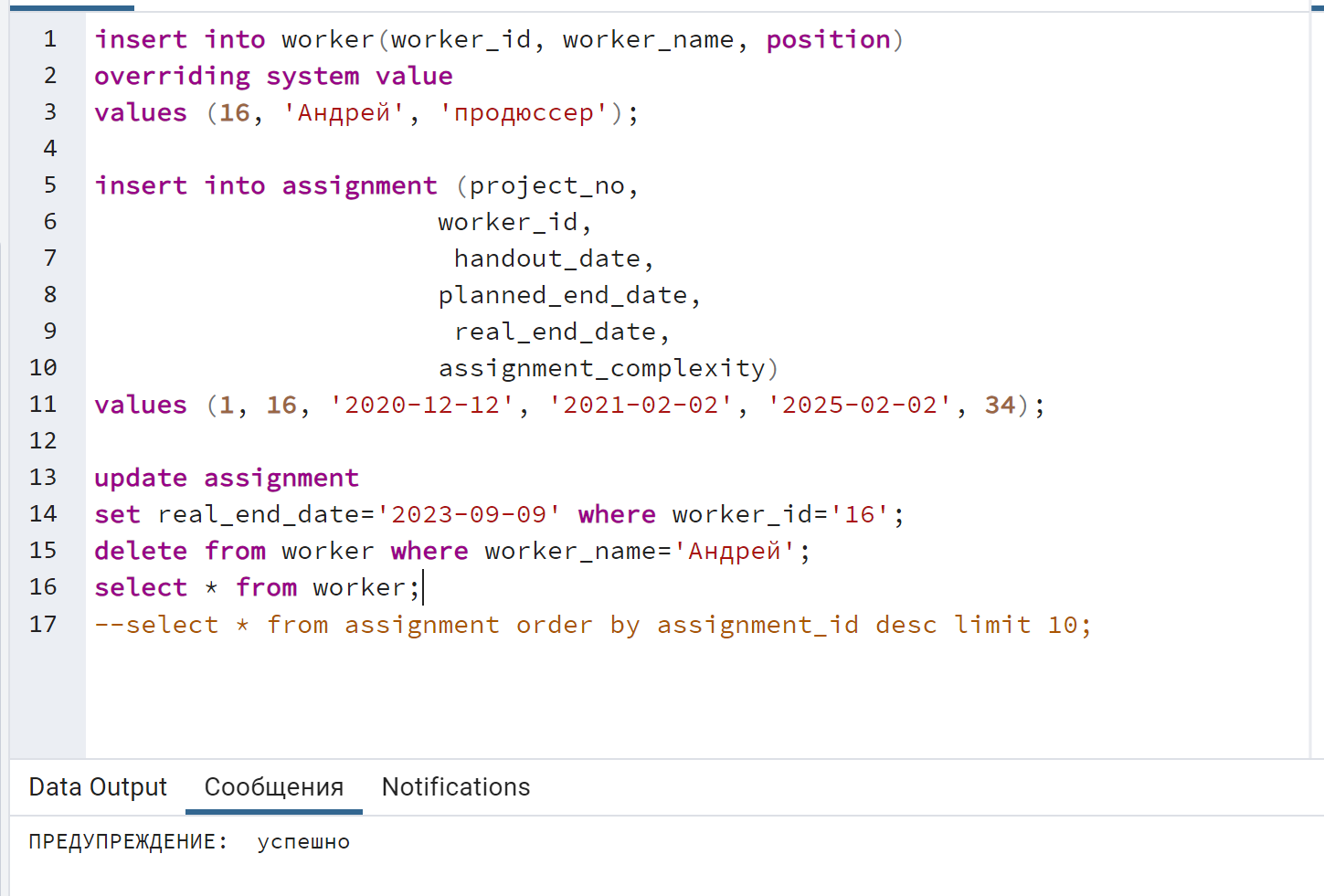


Рисунок 3.4 – проверка на удачное удаление

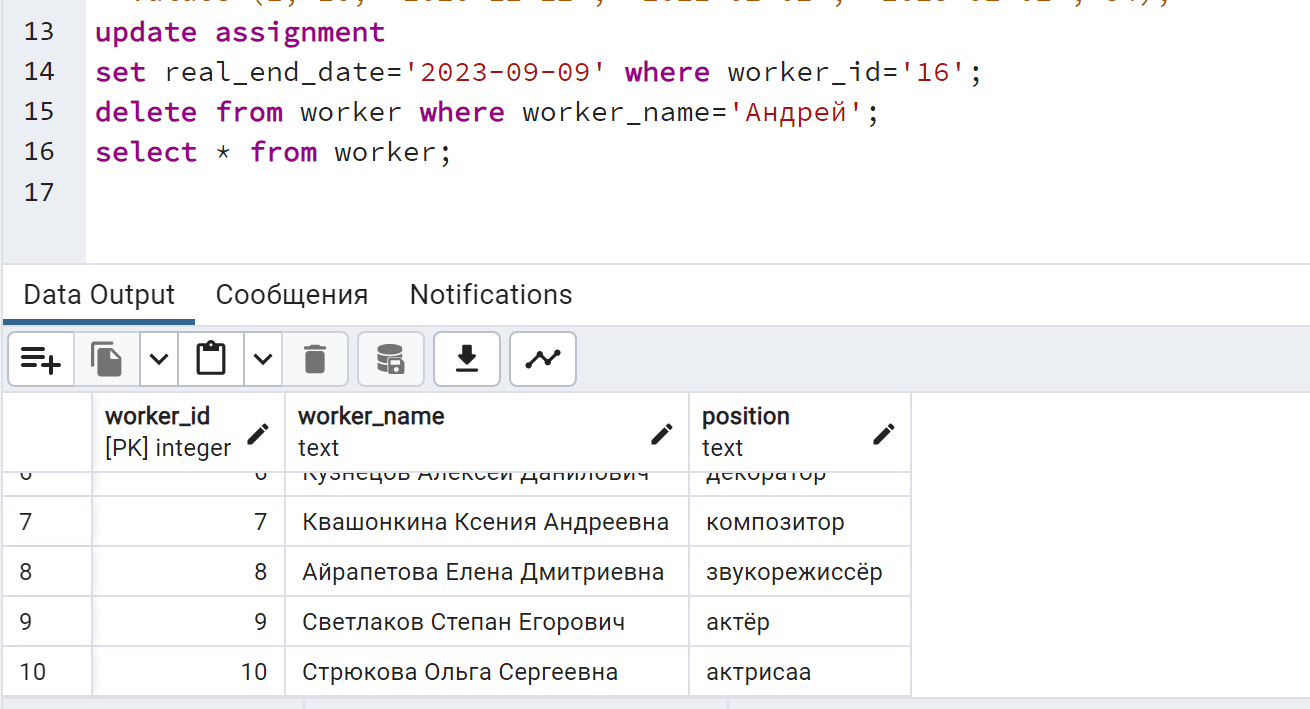


Рисунок 3.5 – доказательство удачного удаления

**Задание 4**

Создать таблицу, состоящую из двух целочисленных полей и содержащую одну запись, для хранения количества проектов и работников.

Написать триггеры для таблиц проекты и работники, подсчитывающие при добавлении и удалении общее количество проектов и работников и, сохраняющие итоги в созданной таблице.

**Создание таблицы:**

create table journal(

count\_p integer not null,

count\_w integer not null,

op text not null,

id integer not null primary key generated always as identity);

**Триггерная функция:**

СREATE OR REPLACE FUNCTION fun\_journal() RETURNS trigger

LANGUAGE plpgsql

AS $$

BEGIN

IF TG\_LEVEL != 'STATEMENT' OR TG\_WHEN != 'AFTER' THEN

RAISE EXCEPTION 'Ошибка в установки триггера';

RETURN NULL;

END IF;

IF TG\_OP = 'INSERT' OR TG\_OP = 'UPDATE' THEN

INSERT INTO journal

SELECT (select Count(\*) from project),

(select count(\*) from worker), (select TG\_OP FROM new\_tab);

ELSE

INSERT INTO journal

SELECT (select Count(\*) from project),

(select count(\*) from worker), (select TG\_OP FROM old\_tab);

END IF;

RETURN NULL;

END;

$$;

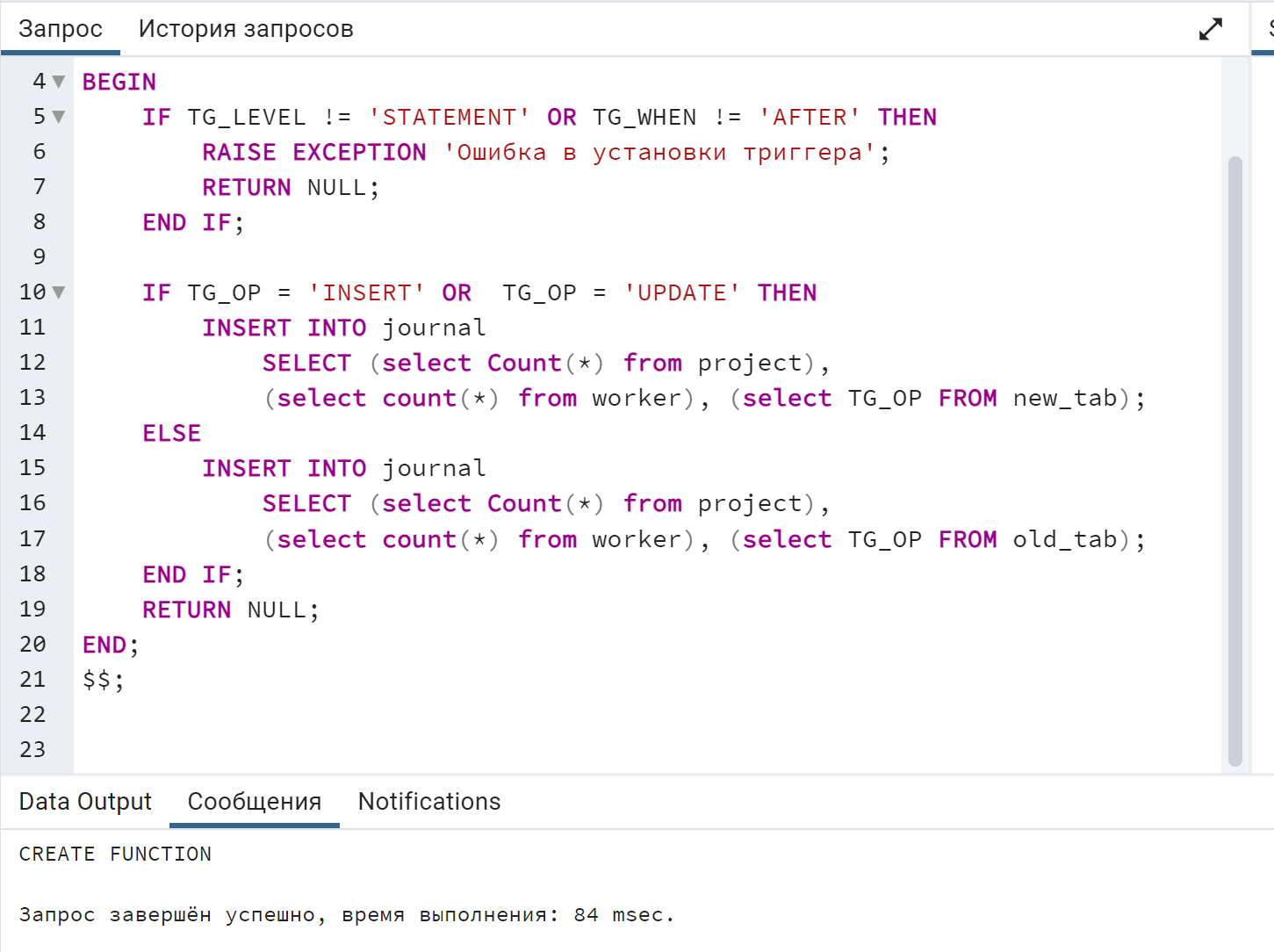


Рисунок 4.1 – создание триггерной функции (INSERT/UPDATE/DELETE) для journal

**Создание триггера для INSERT** в таблицы project и worker:

CREATE TRIGGER tr\_ins\_project

AFTER INSERT ON project

REFERENCING NEW TABLE AS new\_tab

FOR EACH STATEMENT

EXECUTE FUNCTION fun\_journal();

create trigger tr\_ins\_worker

after insert on worker

referencing new table as new\_tab

for each statement

execute function fun\_journal();



Рисунок 4.2 – создание триггера INSERT для project и worker

**Проверка:**

--insert into worker(worker\_name, position)

--values ('Илья', 'режиссёр');

--insert into project(project\_name, project\_complexity, deadline)

--values ('Хоббит', 32, '2020-02-02');

--delete from worker where worker\_name='Илья';

--delete from project where project\_name='Хоббит';

--delete from journal;

select \* from journal;

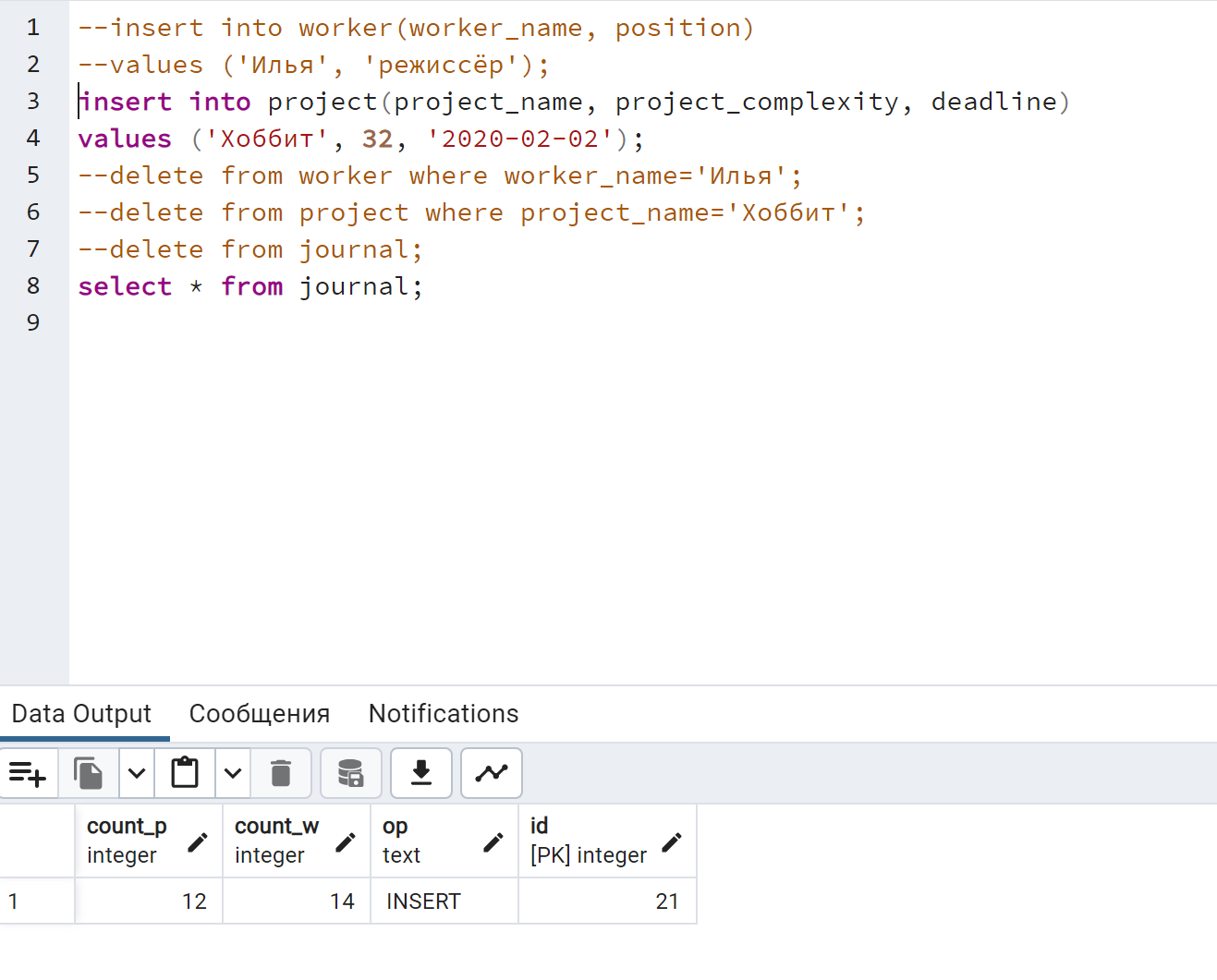


Рисунок 4.3 – проверка INSERT для project

**Создание триггера для UPDATE** для таблиц project и worker:

CREATE TRIGGER tr\_upd\_project

AFTER UPDATE ON project

REFERENCING NEW TABLE AS new\_tab

FOR EACH STATEMENT

EXECUTE FUNCTION fun\_journal();

CREATE TRIGGER tr\_upd\_worker

AFTER UPDATE ON worker

REFERENCING NEW TABLE AS new\_tab

FOR EACH STATEMENT

EXECUTE FUNCTION fun\_journal();

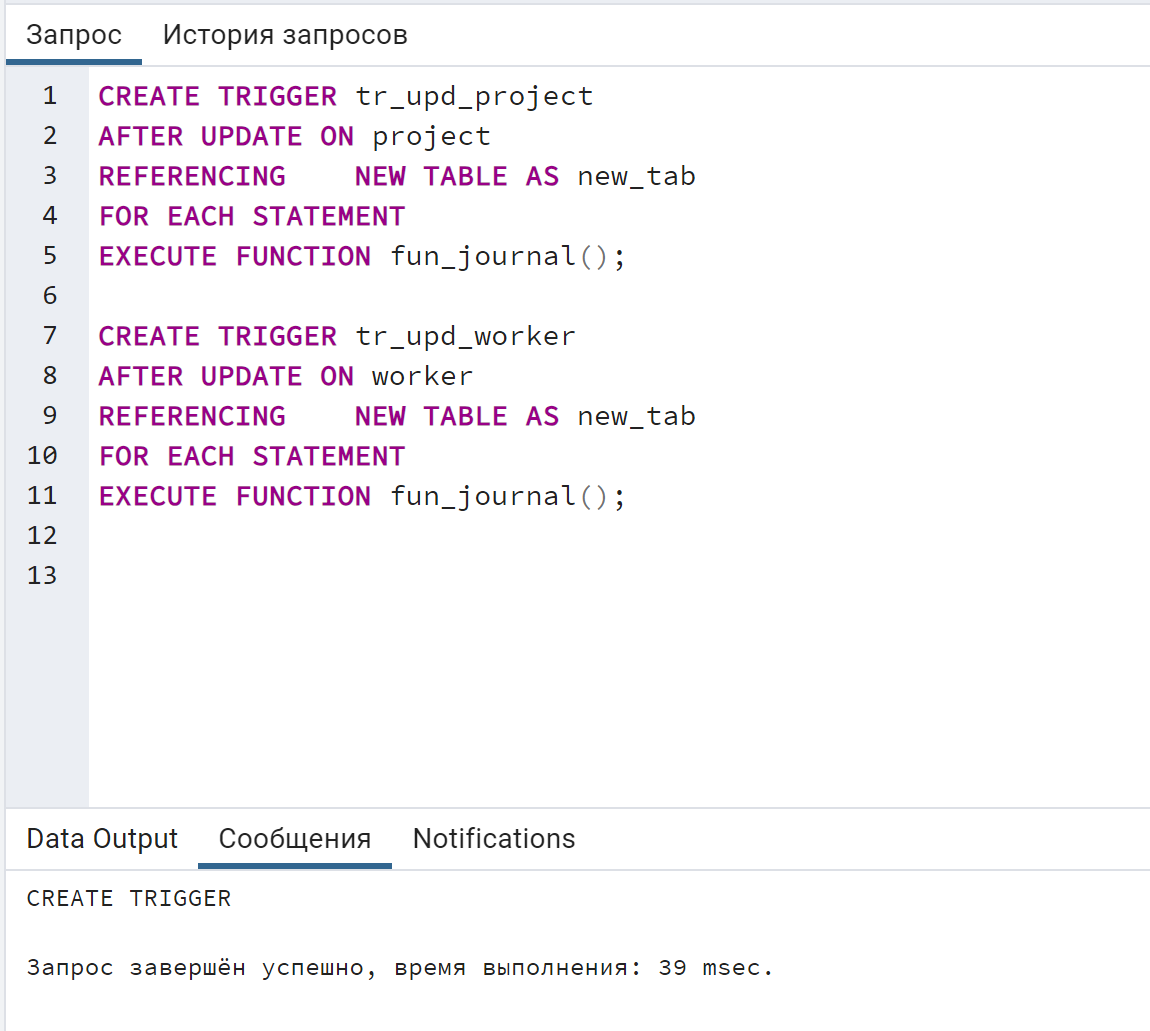


Рисунок 4.4 – создание триггера UPDATE для project и worker

**Проверка:**

UPDATE project

SET project\_complexity = project\_complexity+10,

project\_name = project\_name || 'a'

WHERE project\_no = 10;

update worker

set position=position||'a' where worker\_id=10;

SELECT \* FROM journal;

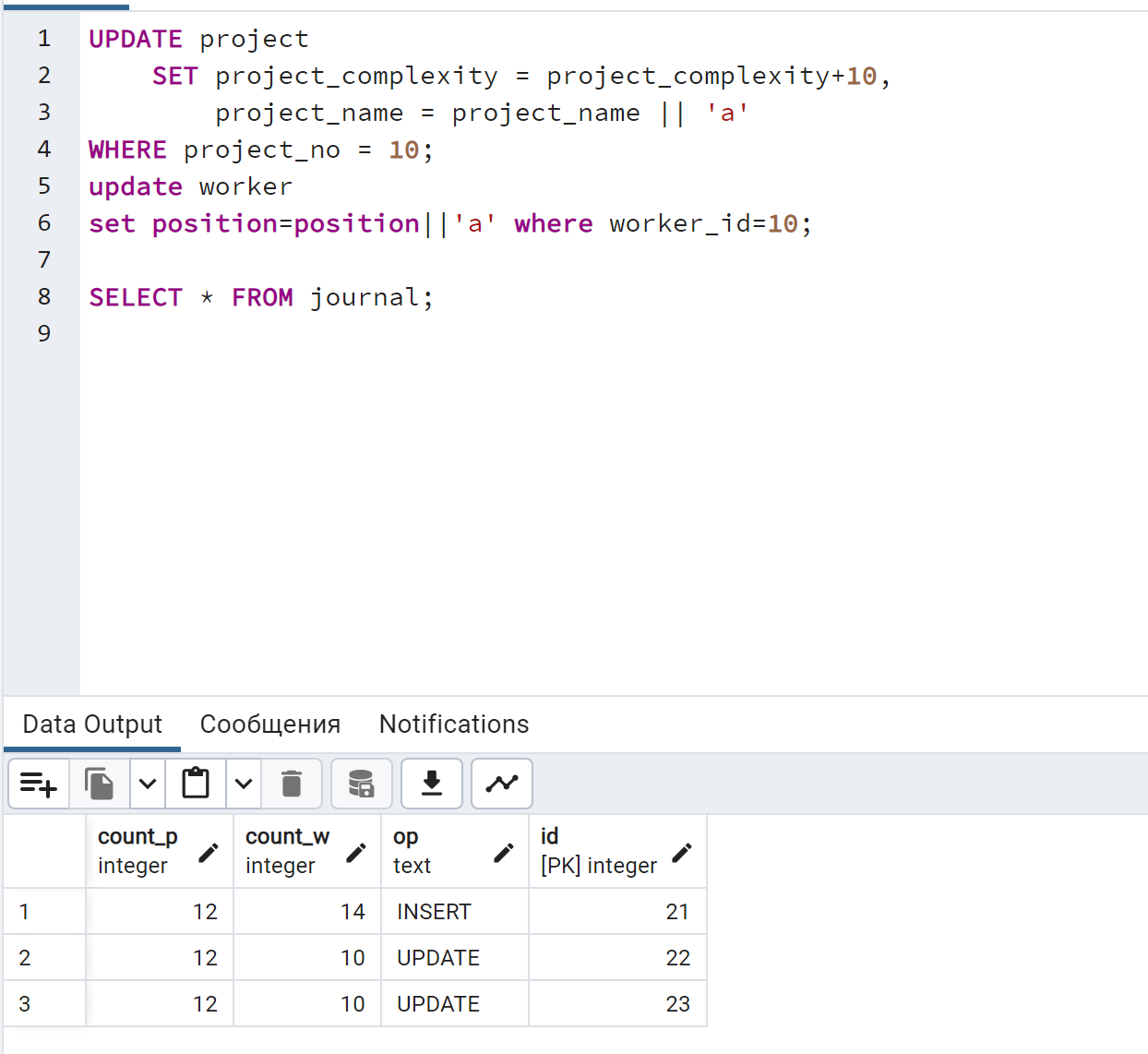


Рисунок 4.5 – проверка UPDATE для project и worker

**Создадим триггеры DELETE** для таблиц project и worker:

CREATE TRIGGER tr\_del\_project

AFTER DELETE ON project

REFERENCING OLD TABLE AS old\_tab

FOR EACH STATEMENT

EXECUTE FUNCTION fun\_journal();

CREATE TRIGGER tr\_del\_worker

AFTER DELETE ON worker

REFERENCING OLD TABLE AS old\_tab

FOR EACH STATEMENT

EXECUTE FUNCTION fun\_journal();

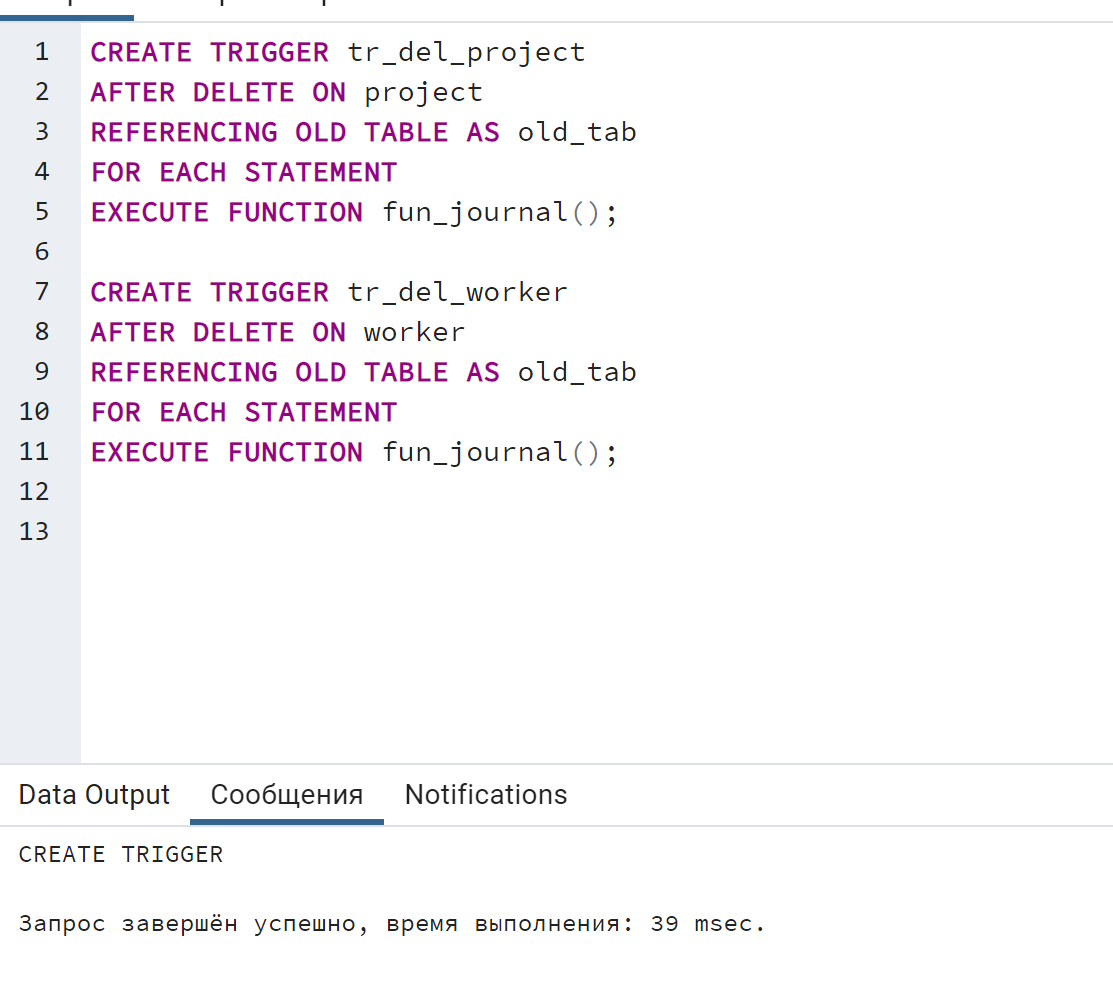


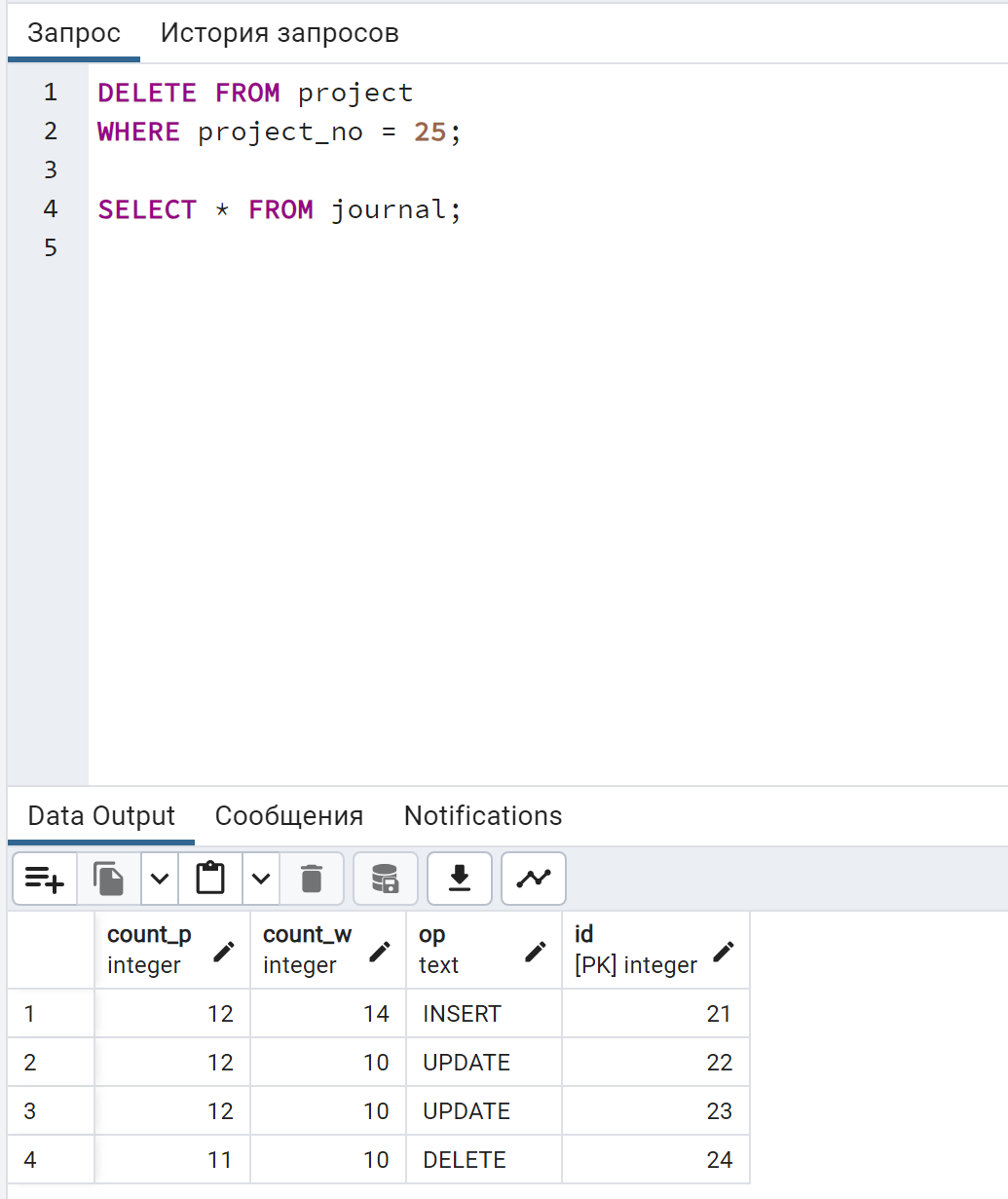
Рисунок 4.6 – создание триггера DELETE для project и worker

**Проверка:**

DELETE FROM project

WHERE project\_no = 25;

SELECT \* FROM journal;

  
Рисунок 4.7 – проверка DELETE для project

**Задание 5**

Создать пользователя test и выдать ему доступ к базе данных.

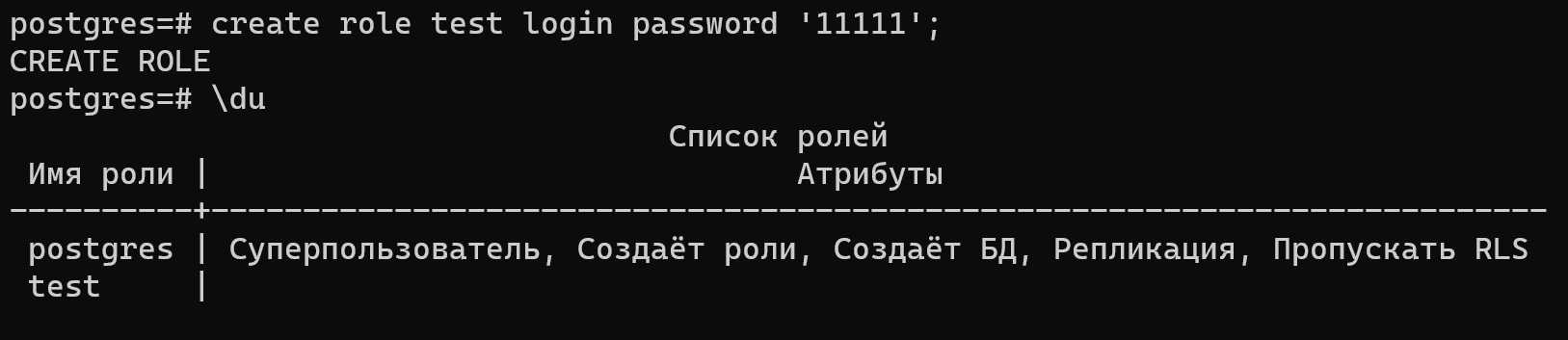


Рисунок 5.1 – создание роли test

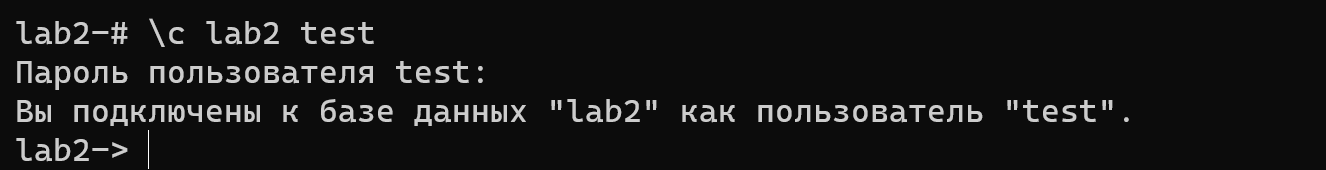


Рисунок 5.2 – подключение test к базе данных lab2

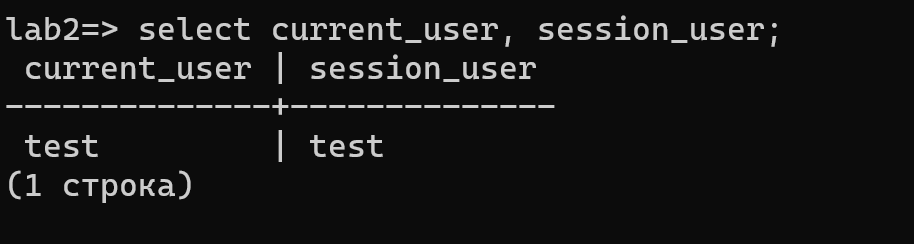


Рисунок 5.3 – проверка текущих пользователей

**Задание 6**

Составить и выполнить скрипты присвоения новому пользователю прав доступа к таблицам, созданным в практическом задании 1. При этом права доступа к различным таблицам должны быть различными, а именно:

* По крайней мере, для одной таблицы новому пользователю присваиваются права SELECT, INSERT, UPDATE в полном объеме.
* По крайней мере, для одной таблицы новому пользователю присваиваются права SELECT и UPDATE только избранных столбцов.
* По крайней мере, для одной таблицы новому пользователю присваивается только право SELECT.

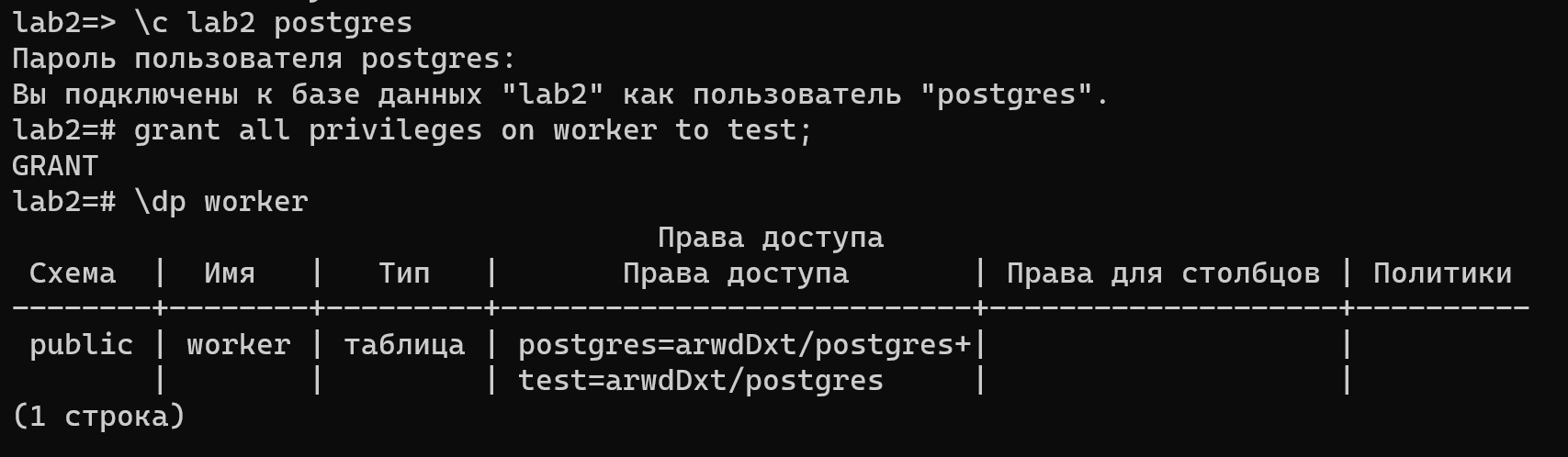


Рисунок 6.1 – предоставление всех привилегий от postgres test-у на таблицу worker

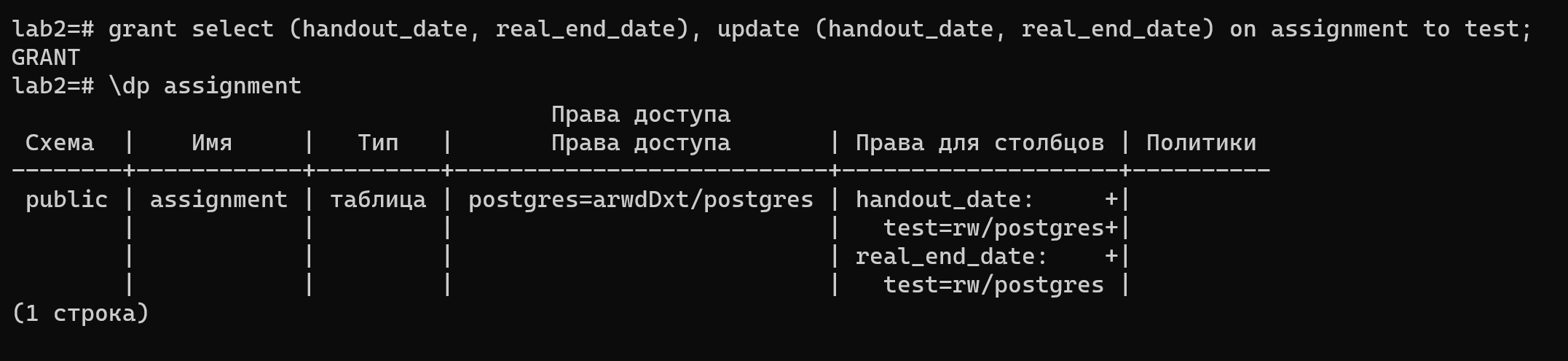


Рисунок 6.2 – предоставление привилегий select, update от postgres test-у на некоторые столбцы таблицы assignment

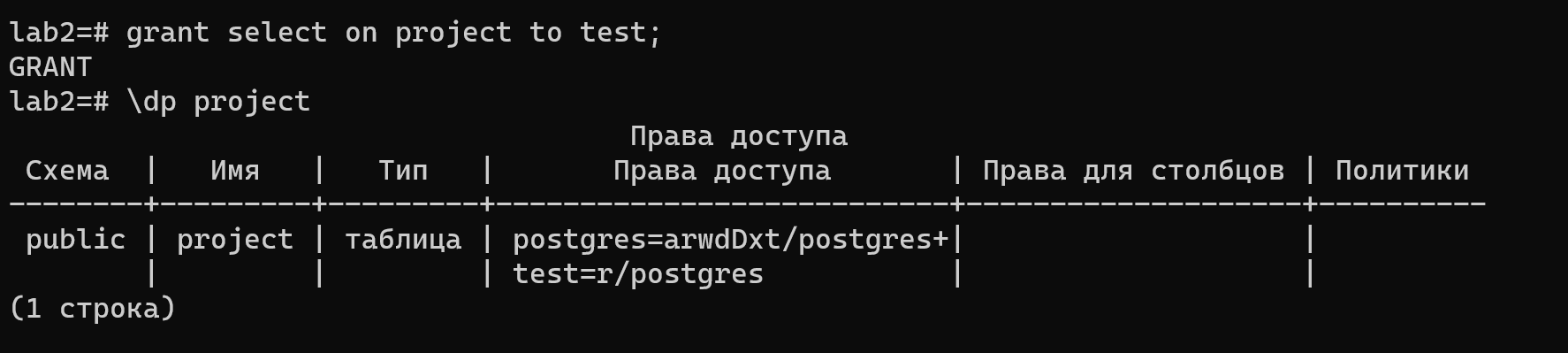


Рисунок 6.3 – предоставление привилегий select от postgres test-у на таблицу project

Привилегии отображаются в формате: роль=привилегии/кем\_предоставлены.

Каждая привилегия кодируется одним символом:

* a = insert
* r = select
* w = update
* d = delete
* D = truncate
* x = reference
* t = trigger

**Вывод**: мы научились работать с триггерами, курсорами, массивами, получили знания о ролях и предоставлении пользователям разного доступа к данным