

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

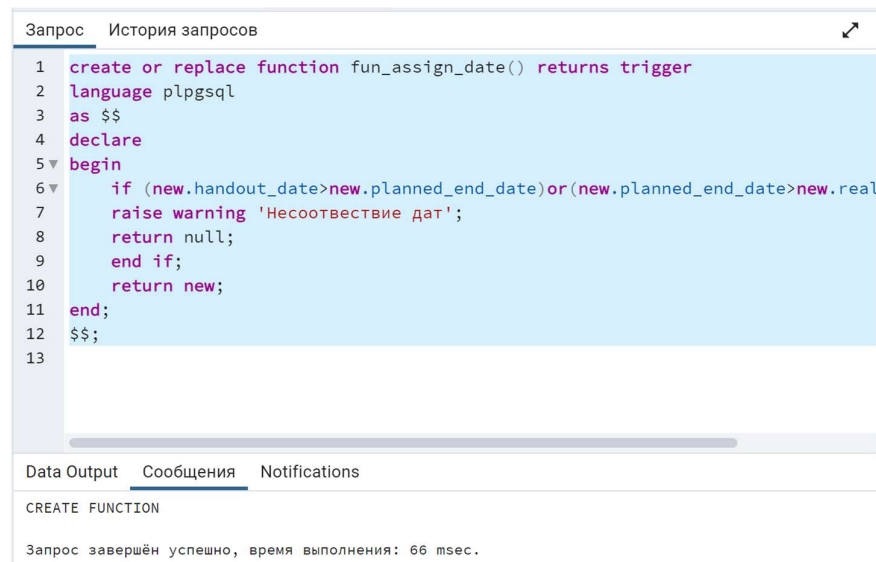
Для таблицы assignments напомним триггер, который не даст завести поручение с датой выдачи больше плановой даты окончания, большей реальной датой окончания

- 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_d
    ate) then
    raise warning 'Несоответствие дат';
    return null;
  end if;
  return new;
```

```
end;  
$$;
```



The screenshot shows a SQL IDE window with a tab labeled 'Запрос' (Query). The main editor area contains the following SQL code:

```
1 create or replace function fun_assign_date() returns trigger  
2 language plpgsql  
3 as $$  
4 declare  
5 begin  
6     if (new.handout_date > new.planned_end_date) or (new.planned_end_date > new.real_end_date)  
7     then  
8         raise warning 'Несоответствие дат!';  
9     end if;  
10    return new;  
11 end;  
12 $$;
```

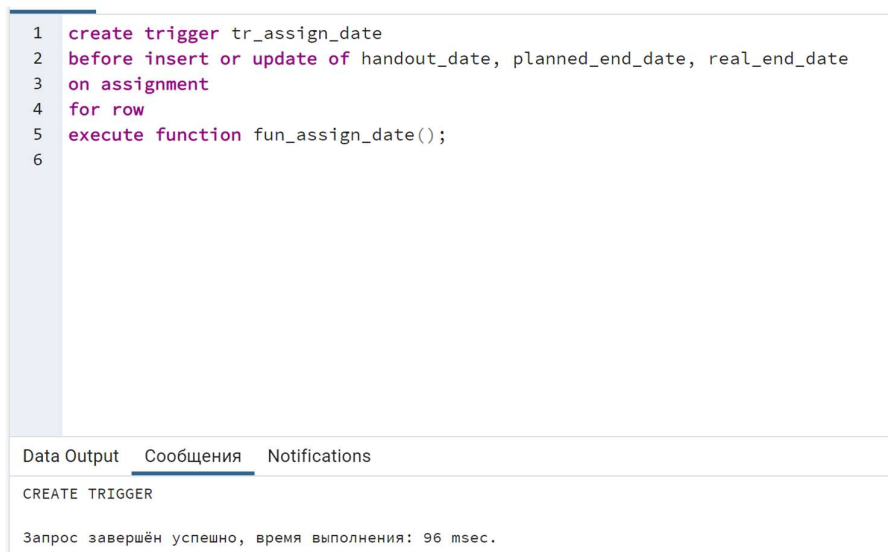
Below the editor, there are three tabs: 'Data Output', 'Сообщения' (Messages), and 'Notifications'. The 'Сообщения' tab is active, showing the following output:

```
CREATE FUNCTION  
Запрос завершён успешно, время выполнения: 66 msec.
```

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();
```



The screenshot shows a SQL IDE window with a tab labeled 'Запрос' (Query). The main editor area contains the following SQL code:

```
1 create trigger tr_assign_date  
2 before insert or update of handout_date, planned_end_date, real_end_date  
3 on assignment  
4 for row  
5 execute function fun_assign_date();  
6
```

Below the editor, there are three tabs: 'Data Output', 'Сообщения' (Messages), and 'Notifications'. The 'Сообщения' tab is active, showing the following output:

```
CREATE TRIGGER  
Запрос завершён успешно, время выполнения: 96 msec.
```

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);

```

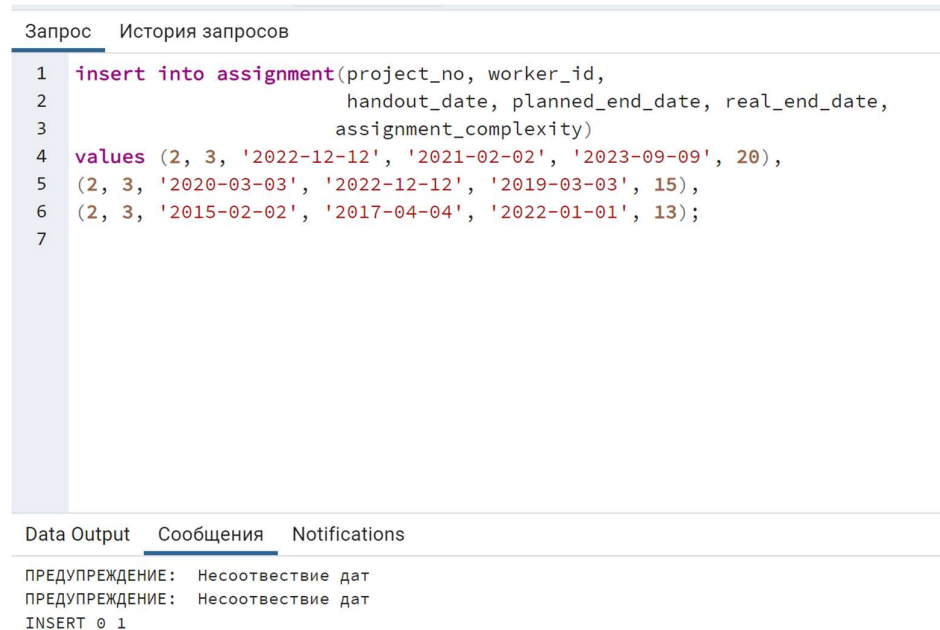


Рисунок 1.3 – проверка для insert

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;

```

```
1 update assignment a
2 set handout_date='2022-12-12',
3 planned_end_date='2021-02-02',
4 real_end_date='2023-09-09'
5 where a.assignment_id=1;
6
```

Data Output	Сообщения	Notifications
ПРЕДУПРЕЖДЕНИЕ: Несоответствие дат		
UPDATE 0		

Рисунок 1.4 – проверка для update

Задание 2

Для таблицы projects напомним 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;
```

\$\$;

```
1 create or replace function fun_proj_name() returns trigger
2 language plpgsql
3 as $$
4 declare
5 begin
6     if exists(select 1 from project where new.project_name=project_name) then
7         raise warning 'Проект с таким названием уже существует';
8         return null;
9     end if;
10    return new;
11 end;
12 $$;
```

Data Output Сообщения Notifications

CREATE FUNCTION

Запрос завершён успешно, время выполнения: 55 msec.

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();

```
1 create trigger tr_proj_name
2 before insert or update of project_name
3 on project
4 for row
5 execute function fun_proj_name();
6
```

Data Output Сообщения Notifications

CREATE TRIGGER

Запрос завершён успешно, время выполнения: 75 msec.

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');

```
1 insert into project (project_name, project_complexity, deadline)
2 values ('Бойцовский клуб', 13, '2020-03-20'),
3 ('Властелин колец', 72, '2010-10-10');
4
```

Data Output Сообщения Notifications

ПРЕДУПРЕЖДЕНИЕ: Проект с таким названием уже существует
INSERT 0 1

Запрос завершён успешно, время выполнения: 55 msec.

Рисунок 2.3 – проверка для insert

Проверка для update:

update project

set project_name='Top' where project_no=10;

```
1 update project
2 set project_name='Top' where project_no=10;
3
```

Data Output Сообщения Notifications

ПРЕДУПРЕЖДЕНИЕ: Проект с таким названием уже существует
UPDATE 0

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;
$$;
```



```
1 create or replace function fun_worker_del() returns trigger
2 language plpgsql
3 as $$
4
5 begin
6
7     if exists(select 1 from assignment as a
8               where a.worker_id=old.worker_id and
9                     real_end_date>'2023-11-12') then
10        raise warning 'Работник имеет поручение';
11        return null;
12    end if;
13    delete from assignment a where a.worker_id=old.worker_id;
14    raise warning 'успешно';
15    return old;
16 end;
17 $$;
```

Data Output **Сообщения** Notifications

CREATE FUNCTION

Запрос завершён успешно, время выполнения: 72 msec.

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();

```
1 create trigger tr_work_del
2 before delete
3 on worker
4 for row
5 execute function fun_worker_del();
6
```

Data Output **Сообщения** Notifications

CREATE TRIGGER

Запрос завершён успешно, время выполнения: 100 msec.

Рисунок 3.2 – создание триггера DELETE для worker

Проверка для delete:

delete from worker where worker_name='Андрей';

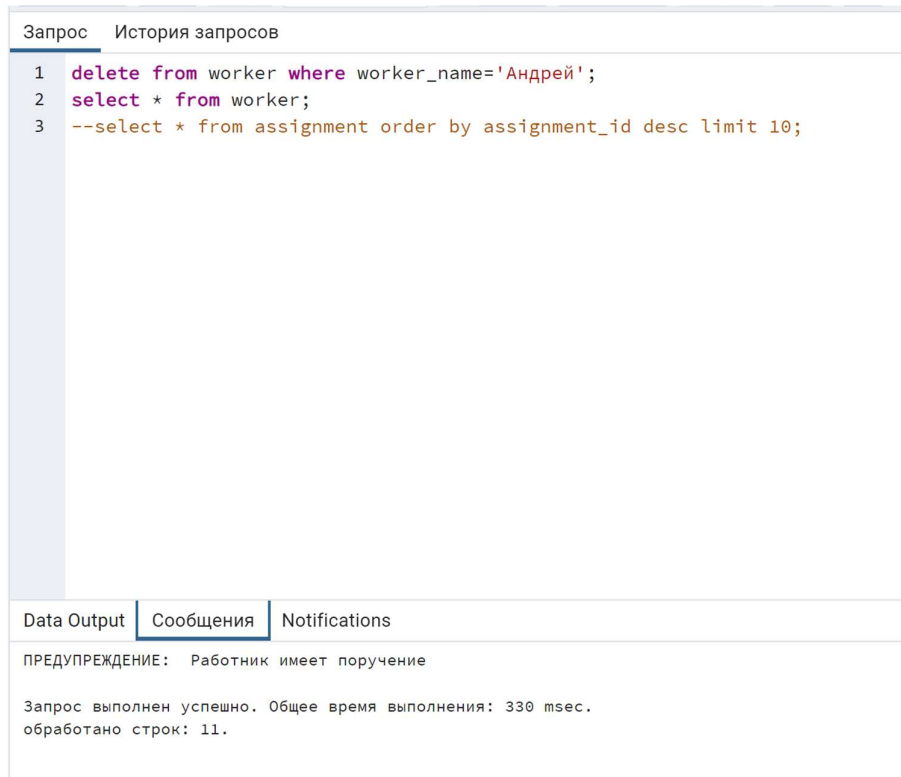


Рисунок 3.3 – проверка на неудачное удаление

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';

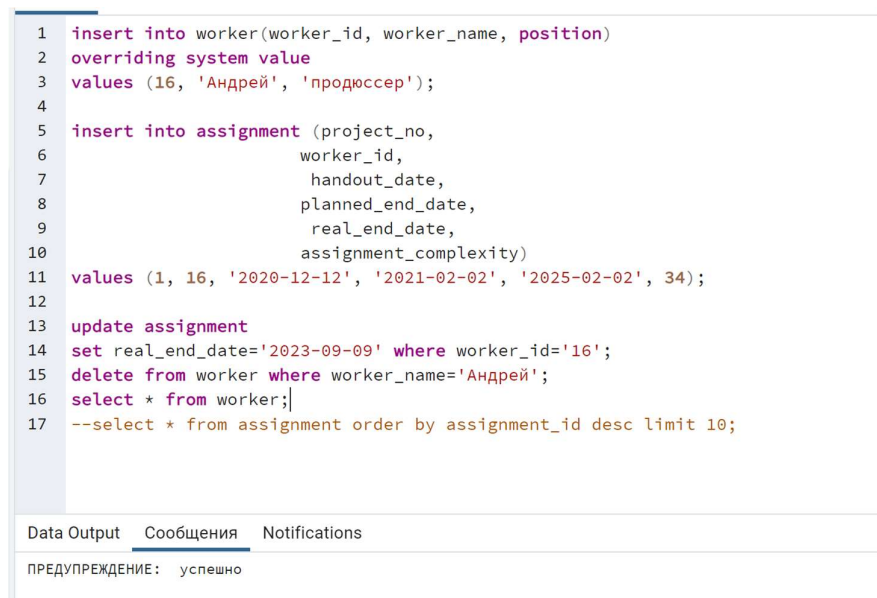


Рисунок 3.4 – проверка на удачное удаление

```

13 update assignment
14 set real_end_date='2023-09-09' where worker_id='16';
15 delete from worker where worker_name='Андрей';
16 select * from worker;
17

```

Data Output

Сообщения

Notifications

≡+

▼

▼

	worker_id [PK] integer	worker_name text	position text
6	6	Кузнецов Алексей Дмитриевич	декоратор
7	7	Квашонкина Ксения Андреевна	композитор
8	8	Айрапетова Елена Дмитриевна	звукорежиссёр
9	9	Светлаков Степан Егорович	актёр
10	10	Стрюкова Ольга Сергеевна	актрисаа

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:

```

CREATE 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;
$$;

```

```

4 BEGIN
5 IF TG_LEVEL != 'STATEMENT' OR TG_WHEN != 'AFTER' THEN
6     RAISE EXCEPTION 'Ошибка в установке триггера';
7     RETURN NULL;
8 END IF;
9
10 IF TG_OP = 'INSERT' OR TG_OP = 'UPDATE' THEN
11     INSERT INTO journal
12         SELECT (select Count(*) from project),
13             (select count(*) from worker), (select TG_OP FROM new_tab);
14 ELSE
15     INSERT INTO journal
16         SELECT (select Count(*) from project),
17             (select count(*) from worker), (select TG_OP FROM old_tab);
18 END IF;
19 RETURN NULL;
20 END;
21 $$;
22
23

```

CREATE FUNCTION

Запрос завершён успешно, время выполнения: 84 мсек.

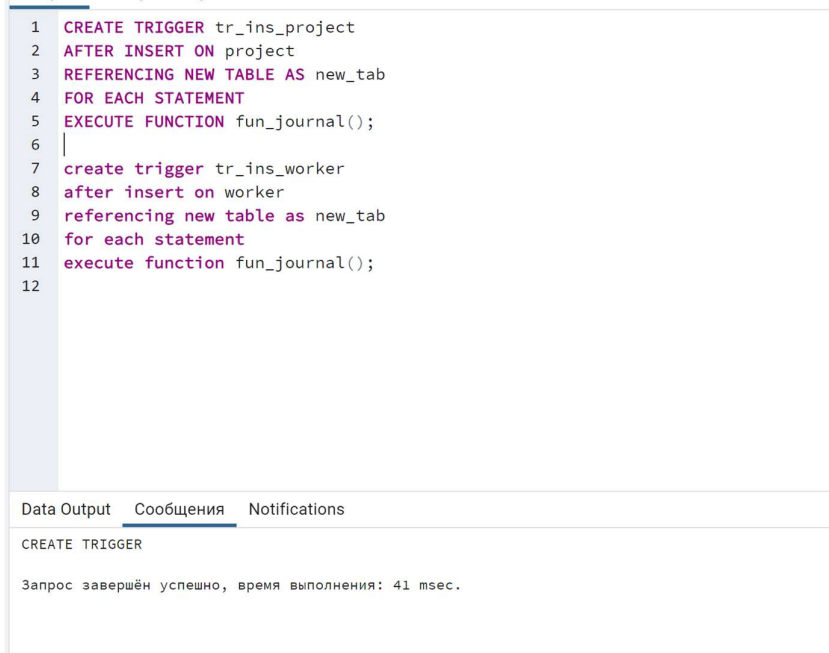
Рисунок 4.1 – создание триггерной функции (INSERT/UPDATE/DELETE) для
journal

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();
```



```
1 CREATE TRIGGER tr_ins_project
2 AFTER INSERT ON project
3 REFERENCING NEW TABLE AS new_tab
4 FOR EACH STATEMENT
5 EXECUTE FUNCTION fun_journal();
6
7 create trigger tr_ins_worker
8 after insert on worker
9 referencing new table as new_tab
10 for each statement
11 execute function fun_journal();
12
```

Data Output **Сообщения** Notifications

CREATE TRIGGER

Запрос завершен успешно, время выполнения: 41 мсек.

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;
```

```

1  --insert into worker(worker_name, position)
2  --values ('Илья', 'режиссёр');
3  insert into project(project_name, project_complexity, deadline)
4  values ('Хоббит', 32, '2020-02-02');
5  --delete from worker where worker_name='Илья';
6  --delete from project where project_name='Хоббит';
7  --delete from journal;
8  select * from journal;
9

```

Data Output Сообщения Notifications

	count_p integer	count_w integer	op text	id [PK] integer
1	12	14	INSERT	21

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();

```
1 CREATE TRIGGER tr_upd_project
2 AFTER UPDATE ON project
3 REFERENCING NEW TABLE AS new_tab
4 FOR EACH STATEMENT
5 EXECUTE FUNCTION fun_journal();
6
7 CREATE TRIGGER tr_upd_worker
8 AFTER UPDATE ON worker
9 REFERENCING NEW TABLE AS new_tab
10 FOR EACH STATEMENT
11 EXECUTE FUNCTION fun_journal();
12
13
```

CREATE TRIGGER

Запрос завершён успешно, время выполнения: 39 msec.

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;

```

1  UPDATE project
2      SET project_complexity = project_complexity+10,
3          project_name = project_name || 'a'
4  WHERE project_no = 10;
5  update worker
6  set position=position||'a' where worker_id=10;
7
8  SELECT * FROM journal;
9

```

Data Output Сообщения Notifications

	count_p integer	count_w integer	op text	id [PK] integer
1	12	14	INSERT	21
2	12	10	UPDATE	22
3	12	10	UPDATE	23

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();

```

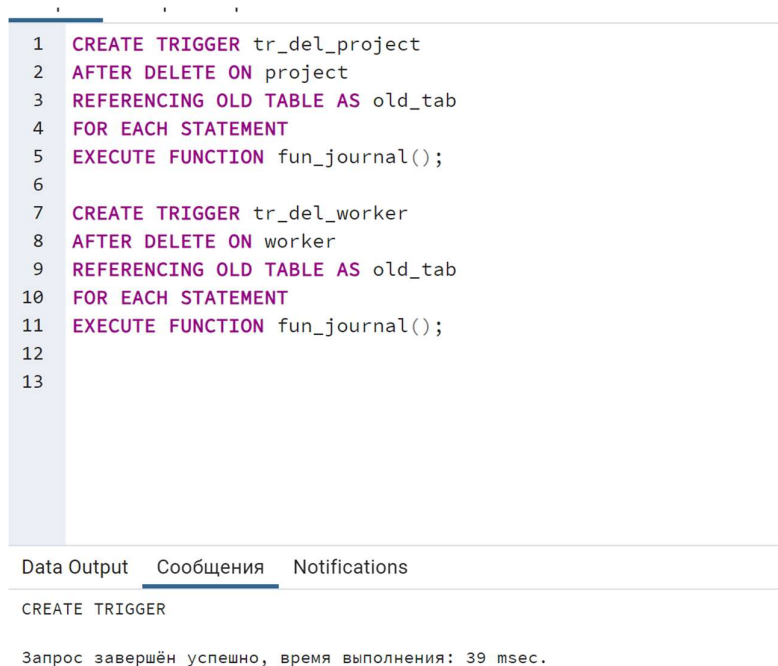



Рисунок 4.6 – создание триггера DELETE для project и worker

Проверка:

DELETE FROM project

WHERE project_no = 25;

SELECT * FROM journal;

Запрос		История запросов			
1	DELETE FROM project				
2	WHERE project_no = 25;				
3					
4	SELECT * FROM journal;				
5					

Data Output		Сообщения		Notifications	
count_p	count_w	op	id		
integer	integer	text	[PK] integer		
1	12	14	INSERT		21
2	12	10	UPDATE		22
3	12	10	UPDATE		23
4	11	10	DELETE		24

Рисунок 4.7 – проверка DELETE для project

Задание 5

Создать пользователя test и выдать ему доступ к базе данных.

```
postgres=# create role test login password '11111';
CREATE ROLE
postgres=# \du
```

Имя роли	Список ролей Атрибуты
postgres	Суперпользователь, Создаёт роли, Создаёт БД, Репликация, Пропускать RLS
test	

Рисунок 5.1 – создание роли test

```
lab2-# \c lab2 test
Пароль пользователя test:
Вы подключены к базе данных "lab2" как пользователь "test".
lab2->
```

Рисунок 5.2 – подключение test к базе данных lab2

```
lab2=> select current_user, session_user;
current_user | session_user
-----+-----
test         | test
(1 строка)
```

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.

```
lab2=> \c lab2 postgres
Пароль пользователя postgres:
Вы подключены к базе данных "lab2" как пользователь "postgres".
lab2=# grant all privileges on worker to test;
GRANT
lab2=# \dp worker
```

Схема	Имя	Тип	Права доступа	Права для столбцов	Политики
public	worker	таблица	postgres=arwdDxt/postgres+ test=arwdDxt/postgres		

(1 строка)

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

```
lab2=# grant select (handout_date, real_end_date), update (handout_date, real_end_date) on assignment to test;
GRANT
lab2=# \dp assignment
```

Схема	Имя	Тип	Права доступа Права доступа	Права для столбцов	Политики
public	assignment	таблица	postgres=arwdDxt/postgres	handout_date: + test=rw/postgres+ real_end_date: + test=rw/postgres	

(1 строка)

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

```
lab2=# grant select on project to test;
GRANT
lab2=# \dp project
```

Схема	Имя	Тип	Права доступа Права доступа	Права для столбцов	Политики
public	project	таблица	postgres=arwdDxt/postgres+ test=r/postgres		

(1 строка)

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