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)

О т ч е т

по лабораторной работе № 4

Название лабораторной работы: Массивы, курсоры, триггеры, роли

Дисциплина: Базы данных

Вариант 18

Студент гр. ИУ6-34Б _____ А. И. Мокшина
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

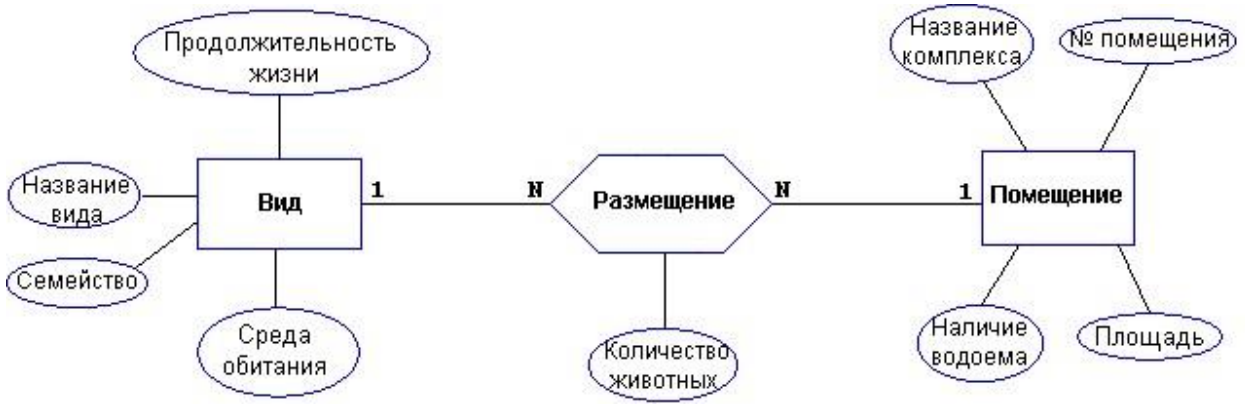
Цель:

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.

  Предметная область для практических заданий: **Зоопарк**



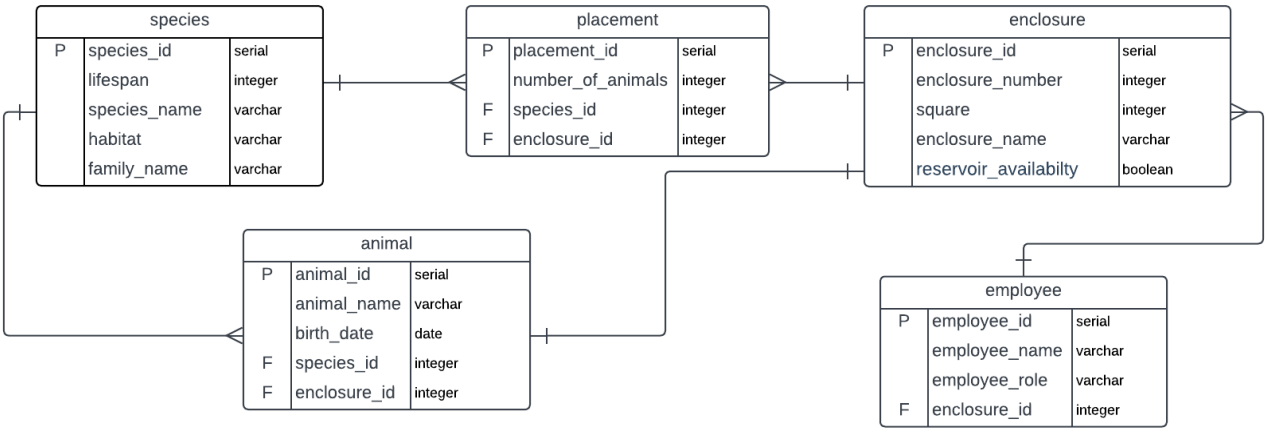


Рис. 1 – спроектированная БД

Написать и протестировать триггеры, выполняющие следующие действия для своей предметной области.

- Контроль наличия водоема для животных, обитающих в водной среде

```
CREATE OR REPLACE FUNCTION check_water_animal()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF NEW.species_id IN (SELECT species_id
                          FROM species
                          WHERE habitat IN ('ocean', 'freshwater'))
       AND NEW.enclosure_id NOT IN (SELECT enclosure_id
                                    FROM enclosure
                                    WHERE reservoir_availability = true) THEN
        RAISE EXCEPTION 'Нельзя поселить животное, обитающее в водной среде, в вольер
без водоема';
        RETURN NULL;
    END IF;

    RETURN NEW;
END;
$$;

CREATE TRIGGER water_animal_trigger
BEFORE INSERT OR UPDATE
ON placement
FOR EACH ROW
EXECUTE FUNCTION check_water_animal();
```

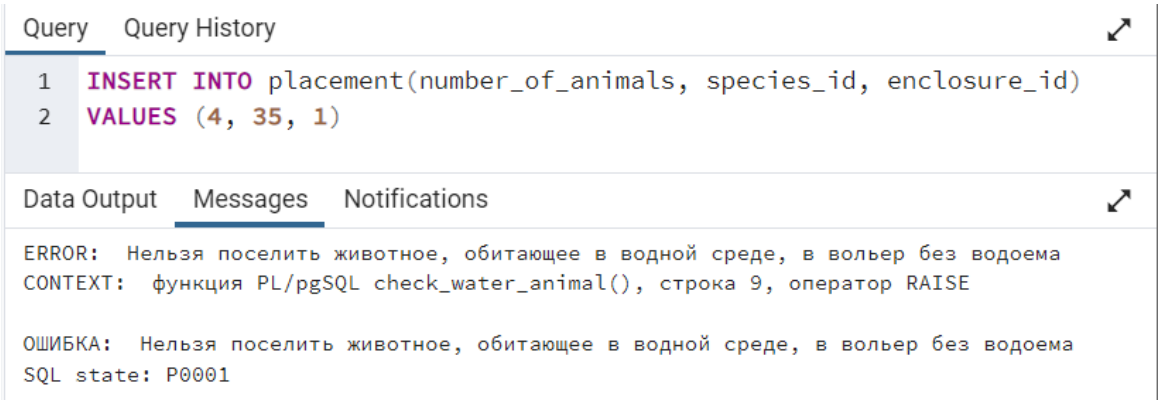


Рис. 2 – работа триггера

- Контроль дублирования животного, сотрудника и вида.

```
CREATE OR REPLACE FUNCTION check_double_animal()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM animal
        WHERE
            animal_name = NEW.animal_name
            AND birth_date = NEW.birth_date
            AND species_id = NEW.species_id
            AND enclosure_id = NEW.enclosure_id
    ) THEN
        RAISE EXCEPTION 'Дублирование животного';
        RETURN NULL;
    END IF;

    RETURN NEW;
END;
$$;

CREATE OR REPLACE TRIGGER double_animal_trigger
BEFORE INSERT OR UPDATE
ON animal
FOR EACH ROW
EXECUTE FUNCTION check_double_animal();
```

Рис. 3 – работа триггера

```sql
CREATE OR REPLACE FUNCTION check_double_employee()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM employee
        WHERE
            employee_name = NEW.employee_name
            AND employee_role = NEW.employee_role
            AND enclosure_id = NEW.enclosure_id) THEN
        RAISE EXCEPTION 'Дублирование сотрудника';
        RETURN NULL;
    END IF;

    RETURN NEW;
END;
$$;

CREATE OR REPLACE TRIGGER double_employee_trigger
BEFORE INSERT OR UPDATE
ON employee
FOR EACH ROW
EXECUTE FUNCTION check_double_employee();
```

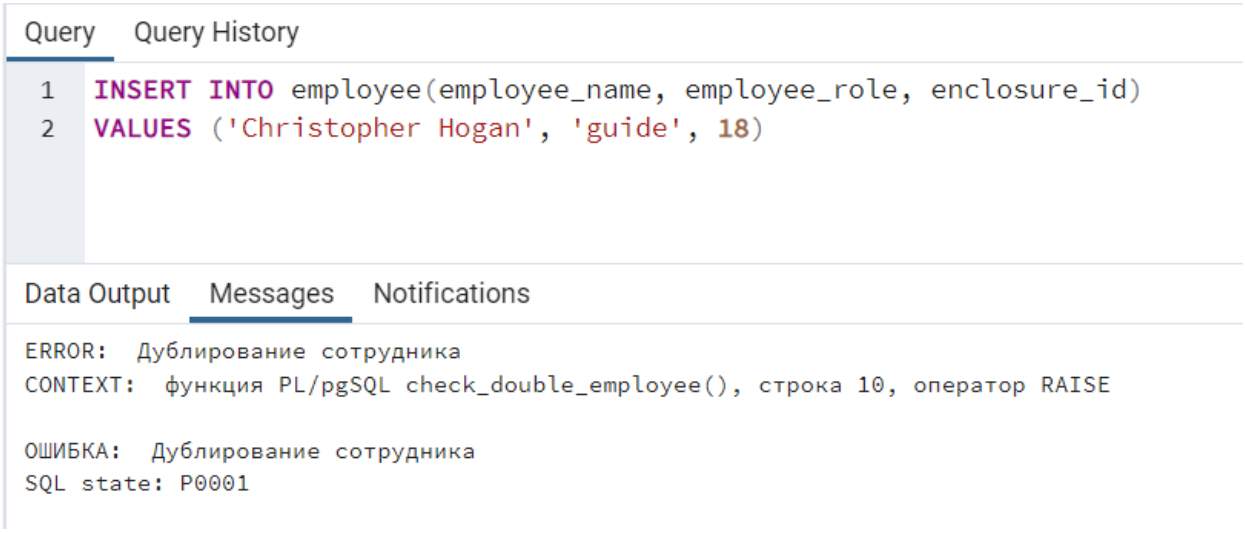


Рис. 4 – работа триггера

```
CREATE OR REPLACE FUNCTION check_double_species()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM species
        WHERE
            lifespan = NEW.lifespan
            AND species_name = NEW.species_name
            AND habitat = NEW.habitat
            AND family_name = NEW.family_name) THEN
        RAISE EXCEPTION 'Дублирование вида животного';
        RETURN NULL;
    END IF;

    RETURN NEW;
END;
$$;

CREATE OR REPLACE TRIGGER double_species_trigger
BEFORE INSERT OR UPDATE of range
ON species
FOR EACH ROW
EXECUTE FUNCTION check_double_species();
```

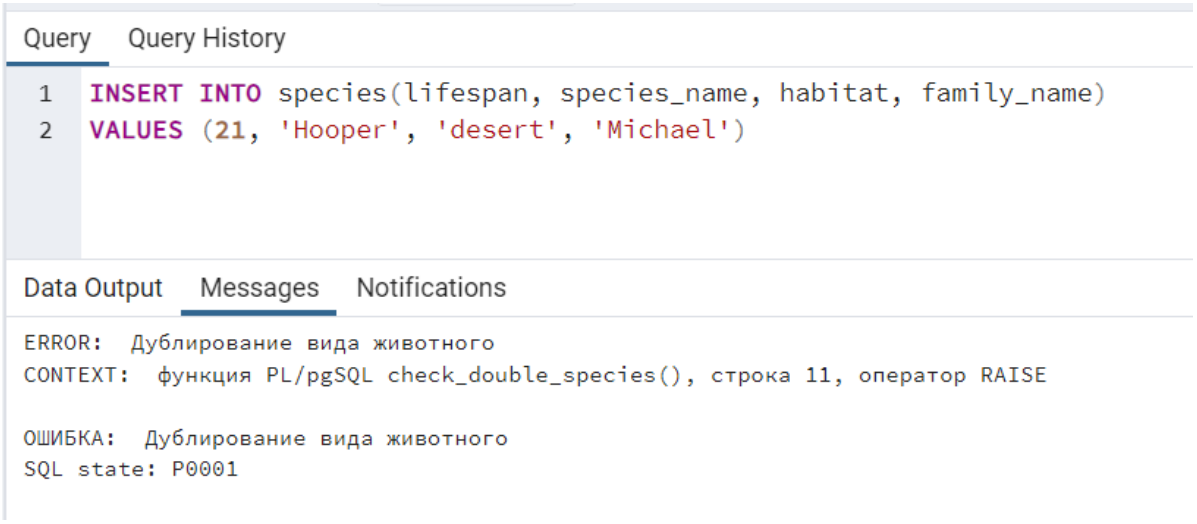


Рис. 5 – работа триггера

- Запрет на удаления помещения, если в этом помещении живет животное или работает сотрудник.

```
CREATE OR REPLACE FUNCTION check_delete_enclosure()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM animal
        WHERE enclosure_id = OLD.enclosure_id
    ) THEN
        RAISE EXCEPTION 'Нельзя удалить помещение, в котором живет животное';
        RETURN NULL;
    END IF;

    IF EXISTS (
        SELECT 1
        FROM employee
        WHERE enclosure_id = OLD.enclosure_id
    ) THEN
        RAISE EXCEPTION 'Нельзя удалить помещение, в котором работает сотрудник';
        RETURN NULL;
    END IF;

    RETURN OLD;
END;
$$;

CREATE OR REPLACE TRIGGER delete_enclosure_trigger
BEFORE DELETE
ON enclosure
FOR EACH ROW
EXECUTE FUNCTION check_delete_enclosure();
```

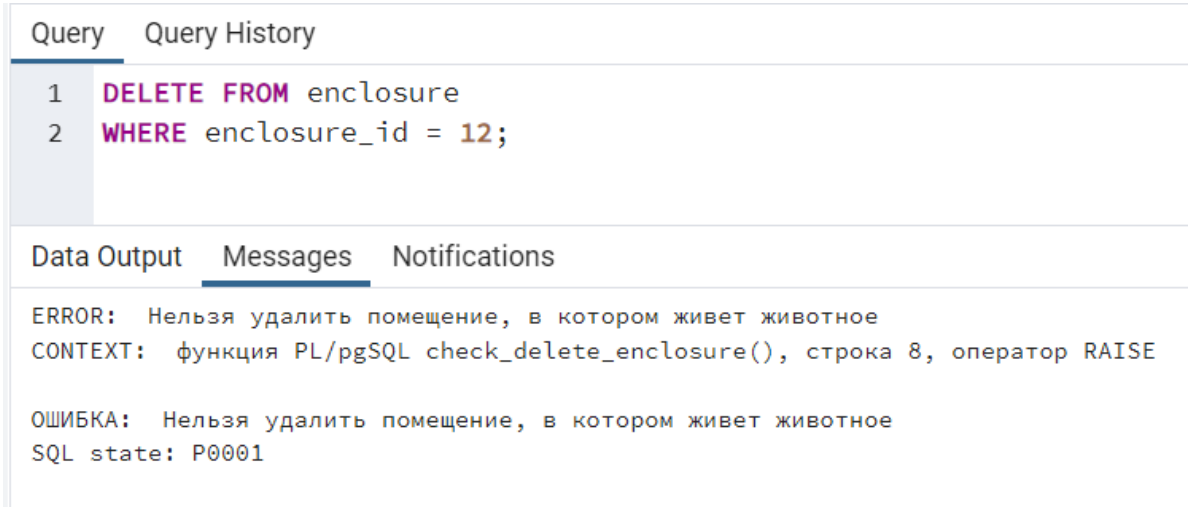


Рис. 6 – работа триггера

- Создать таблицу, состоящую из двух целочисленных полей и содержащую одну запись, для хранения количества животных и сотрудников. Написать триггеры для таблиц *животные* и *сотрудники*, подсчитывающие при добавлении и удалении общее количество животных и сотрудников и, сохраняющие итоги в созданной таблице.

```
CREATE TABLE animal_employee_counts (
    animal_count INTEGER DEFAULT 0,
    employee_count INTEGER DEFAULT 0
);

INSERT INTO animal_employee_counts DEFAULT VALUES;

UPDATE animal_employee_counts
SET
    animal_count = (SELECT COUNT(*) FROM animal),
    employee_count = (SELECT COUNT(*) FROM employee);
```

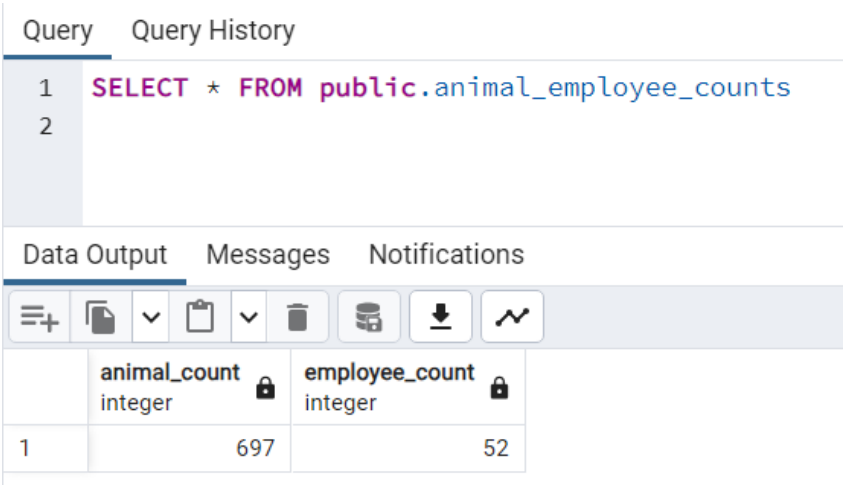


Рис. 7 – созданная таблица

```
CREATE OR REPLACE FUNCTION update_counts_animals()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        UPDATE animal_employee_counts
        SET animal_count = animal_count + 1;
        RETURN NEW;
    END IF;


    IF TG_OP = 'DELETE' THEN
        UPDATE animal_employee_counts
        SET animal_count = animal_count - 1;
        RETURN NULL;
    END IF;

END;
$$;
CREATE OR REPLACE TRIGGER animal_count_trigger
AFTER INSERT OR DELETE
ON animal
FOR EACH ROW
EXECUTE FUNCTION update_counts_animals();

CREATE OR REPLACE FUNCTION update_counts_employee()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        UPDATE animal_employee_counts
        SET employee_count = employee_count + 1;
        RETURN NEW;
    END IF;


    IF TG_OP = 'DELETE' THEN
        UPDATE animal_employee_counts
        SET employee_count = employee_count - 1;
        RETURN NULL;
    END IF;

END;
$$;

CREATE OR REPLACE TRIGGER employee_count_trigger
AFTER INSERT OR DELETE
ON employee
FOR EACH ROW
EXECUTE FUNCTION update_counts_employee() ;
```

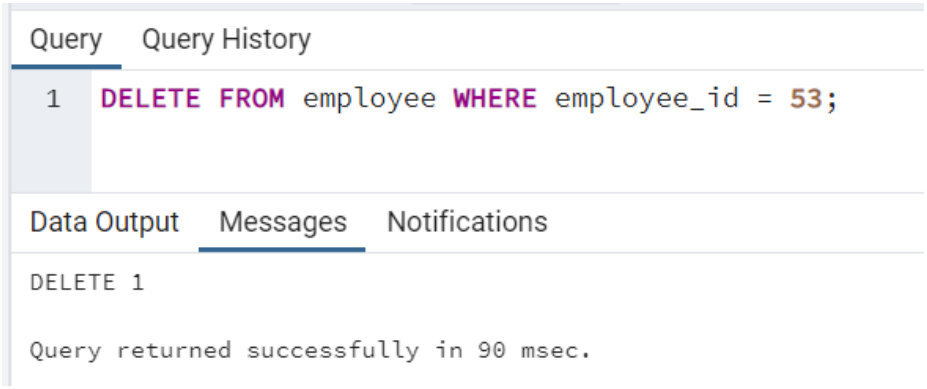


Рис. 8 – удаление записи

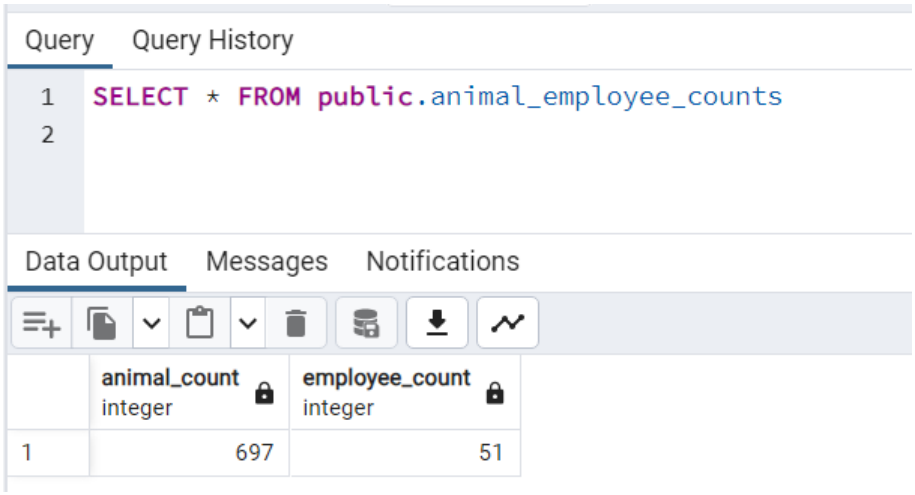
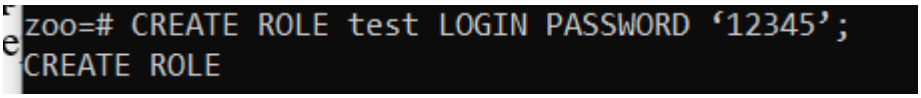
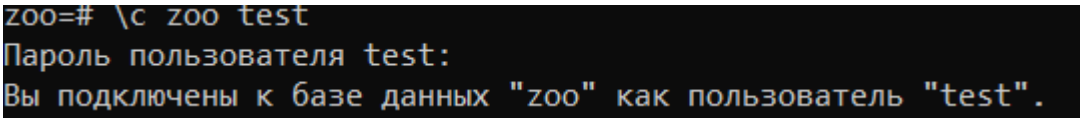


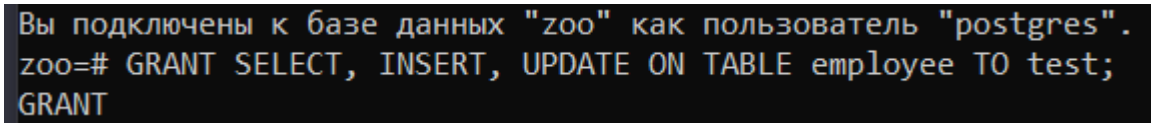
Рис. 9 – проверка работы триггера

- 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.

```
zoo=# GRANT SELECT (enclosure_number, square, enclosure_name) ON TABLE enclosure TO test;
GRANT
zoo=# GRANT UPDATE (animal_id, enclosure_id) ON TABLE animal TO test;
GRANT
```

- По крайней мере, для одной таблицы новому пользователю присваивается только право SELECT.

```
zoo=# GRANT SELECT ON TABLE placement TO test;
GRANT
```

```
zoo=# \c zoo test
Пароль пользователя test:
Вы подключены к базе данных "zoo" как пользователь "test".
zoo=> select * from placement limit 10
zoo-> ;
 placement_id | number_of_animals | species_id | enclosure_id
--------------+-------------------+------------+--------------
            1 |                 2 |         35 |            3
            2 |                 2 |         32 |           28
            3 |                 4 |         21 |           25
            4 |                 2 |         34 |           28
            5 |                 5 |         19 |           14
            6 |                 1 |         33 |           22
            7 |                 3 |         25 |           13
            8 |                 2 |          4 |           14
            9 |                 1 |         20 |           16
           10 |                 5 |         11 |           27
(10 ёЄЁюъ)
```

Вывод: были ознакомлены с назначением массивов, курсоров, триггеров и ролей, освоены навыки их написания и использования.