# Отчет по лабораторной работе 6.

Выполнил: Золотухин Андрей Александрович КС-36
Преподаватель: Семёнов Геннадий Николаевич

# ЗАДАНИЕ

## Лаб№6 Функции, Хранимые Процедуры и Триггеры 2023

### Оформление отчета:

1. Сформулировать на русском языке (в виде текста) описание объектов вашей БД в соответствии с вашим вариантом задания.
2. Написать код создающий данные объекты БД на языке pgsql или SQL.
3. Приложить результаты выполнения функций, процедур, триггеров в виде картинки.

### Задания:

### Функции и Хранимые процедуры

1. Создать функцию **Period(…, …)** с двумя входными параметрами типа date, которая выберет строки из дочерней таблицы в диапазоне дат, указанных первым и вторым аргументами при вызове функции **Period**(…, …).
2. Создать функцию **Sum_object(…)**(с одним параметром), которая возвращает список имен **объектов** из родительской таблицы на основании данных дочерней таблицы. Список **объектов** определяется значением параметра, исходя из условия, что суммарное количество **объектов** должно быть больше, чем заданное значение в параметре.
3. Создать функцию **row_count(…, …)**, которая подсчитывает количество строк дочерней таблицы, даты которых находятся между параметрами **date_from** и **date_to**.
4. Создать хранимую процедуру **object_stat(…)**, которая подсчитывает минимальное, максимальное и среднее значение **атрибута** в дочерней таблице, входным параметром является **имя объекта родительской таблицы**.
5. Создать хранимую процедуру **objects_stat(…)**, которая подсчитывает минимальное, максимальное и среднее значение каждого объекта в дочерней таблице и выводит имя объекта, входным параметром является **имя объекта** из родительской таблицы.
6. Создать хранимую процедуру **Itog**(text) с одним входным параметром (**имя объекта**), которая выводит наименование **объекта** по суммарному количеству **объектов и оценку**:
   a) **Оценка** «**Незначительный объект**», если число объектов меньше 2
   б) **Оценка** «**Обычный объект**», если число объектов больше или равно 2 и меньше или равно 3
   c) **Оценка** «**Значительный объект**», если число объектов больше 3

### Триггеры

7. Создать триггер **After_Delete**, который при удалении записи из родительской таблицы удалял бы все связанные записи из дочерней таблицы. Показать результат работы триггера.
8. Создать триггер **Before_Delete**, который при удалении записи из дочерней таблицы выводил бы имя **объекта** родительской таблицы. Показать результат работы триггера.
9. Создать триггер **ins_sum**, который связывает триггер с таблицей для инструкций INSERT. Это действует как сумматор, чтобы суммировать значения, вставленные в один из столбцов дочерней таблицы. Триггер должен активироваться перед каждой строкой, вставленной в таблицу. Показать результат работы триггера.
10. Создать триггер **Before_Update_Value** на событие UPDATE, который увеличивает значение числового поля дочерней таблицы на 10%. Показать результат работы триггера.

**Результат создания таблиц лабораторной работы 1**

```
Var7=# SELECT * FROM groups;
 group_id | group_name
----------+------------
        1 | CS-10
        2 | CS-14
(2 rows)
```

```
Var7=# SELECT * FROM rooms;
 room_id | room_number
---------+-------------
       1 |          14
       2 |          37
       3 |          25
(3 rows)
```

```
Var7=# SELECT * FROM students;
 student_id | student_name | scholarship | group_name
------------+--------------+-------------+------------
          1 | Vasilyev     |        2000 |          1
          2 | Petrova      |        2570 |          2
          3 | Sidorov      |        2000 |          2
          4 | Ivanov       |        2240 |          1
          5 | Sidorova     |        4500 |          1
          6 | Grishin      |        4000 |          2
(6 rows)
```

```
Var7=# SELECT * FROM accommodations;
 accommodation_id | accommodation_date | distance | room_number | student_name | neighbour_name
------------------+--------------------+----------+-------------+--------------+----------------
                1 | 2005-08-03         |      200 |           1 |            1 |              6
                2 | 2005-08-15         |      435 |           2 |            2 |              5
                3 | 2005-08-02         |      112 |           3 |            3 |              4
                4 | 2005-08-02         |      240 |           3 |            4 |              3
                5 | 2005-08-14         |     1200 |           2 |            5 |              2
                6 | 2005-08-04         |      780 |           1 |            6 |              1
(6 rows)
```

## Выполнение задания
## ФУНКЦИИ И ХРАНИМЫЕ ПРОЦЕДУРЫ

1) **Создать функцию Period(..., …) с двумя входными параметрами типа date, которая выберет строки из дочерней таблицы в диапазоне дат, указанных первым и вторым аргументами при вызове функции Period(..., …):**

*Формирование списка размещений студентов в заданном диапазоне дат*:

```
CREATE OR REPLACE FUNCTION Period(start_date DATE, end_date
DATE)
RETURNS TABLE(accommodation_id INT, accommodation_date DATE,
distance INT, room_number INT, student_name INT,
neighbour_name INT) AS $$
BEGIN
     RETURN QUERY
     SELECT a.*
     FROM public.accommodations a
     WHERE a.accommodation_date BETWEEN start_date AND
end_date;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM Period('2005-08-01', '2005-08-10');
```

```
CREATE FUNCTION
 accommodation_id | accommodation_date | distance | room_number | student_name | neighbour_name
------------------+--------------------+----------+-------------+--------------+----------------
                1 | 2005-08-03         |      200 |           1 |            1 |              6
                3 | 2005-08-02         |      112 |           3 |            3 |              4
                4 | 2005-08-02         |      240 |           3 |            4 |              3
                6 | 2005-08-04         |      780 |           1 |            6 |              1
(4 rows)
```

**2) Создать функцию Sum_object(…)(с одним параметром), которая возвращает список имен объектов из родительской таблицы на основании данных дочерней таблицы. Список объектов определяется значением параметра, исходя из условия, что суммарное количество объектов должно быть больше, чем заданное значение в параметре:**

*Формирование списка групп, суммарная стипендия студентов в которых превышает заданное значение*:

```
CREATE OR REPLACE FUNCTION Sum_object(min_sum INTEGER)
RETURNS TABLE(group_name VARCHAR) AS $$
BEGIN
    RETURN QUERY
    SELECT g.group_name
    FROM public.groups g
    INNER JOIN public.students s ON g.group_id = s.group_name
```

```
    GROUP BY g.group_name
    HAVING SUM(s.scholarship) > min_sum;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM Sum_object(8000);
```

```
CREATE FUNCTION
 group_name
------------
 CS-10
 CS-14
(2 rows)
```

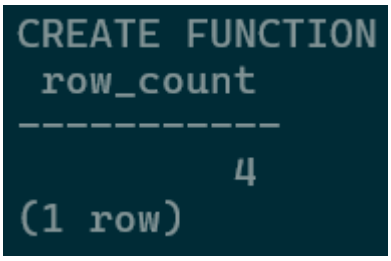
**3) Создать функцию row_count(…, …), которая подсчитывает количество строк дочерней таблицы, даты которых находятся между параметрами date_from и date_to:**

*Формирование количества размещений студентов, которые произошли в заданный период*:

```
CREATE OR REPLACE FUNCTION row_count(date_from DATE, date_to
DATE)
RETURNS INT AS $$
DECLARE
    count_rows INT;
BEGIN
    SELECT COUNT(*)
    INTO count_rows
    FROM public.accommodations
    WHERE accommodation_date BETWEEN date_from AND date_to;

    RETURN count_rows;
END;
$$ LANGUAGE plpgsql;

SELECT row_count('2005-08-01', '2005-08-10');
```




**4) Создать хранимую процедуру object_stat(…), которая подсчитывает минимальное, максимальное и среднее значение атрибута в дочерней таблице, входным параметром является имя объекта родительской таблицы:**

*Формирование статистики (минимальное, максимальное и среднее значение) для различных объектов (студенты, комнаты, размещения)*:

```
CREATE OR REPLACE PROCEDURE object_stat(object_name VARCHAR)
LANGUAGE plpgsql
AS $$
DECLARE
    min_value INTEGER;
    max_value INTEGER;
    avg_value FLOAT;
BEGIN
    IF object_name = 'students' THEN
        SELECT MIN(scholarship), MAX(scholarship), AVG(scholarship)
        INTO min_value, max_value, avg_value
        FROM students;
    ELSIF object_name = 'rooms' THEN
        SELECT MIN(room_number), MAX(room_number), AVG(room_number)
        INTO min_value, max_value, avg_value
        FROM rooms;
    ELSIF object_name = 'accommodations' THEN
        SELECT MIN(distance), MAX(distance), AVG(distance)
        INTO min_value, max_value, avg_value
        FROM accommodations;
    ELSE
        RAISE EXCEPTION 'Unknown object name: %', object_name;
    END IF;

    RAISE NOTICE 'Минимальное значение: %, Максимальное значение:
%, Среднее значение: %', min_value, max_value, avg_value;
END;
```

```
$$;

CALL object_stat('students');
CALL object_stat('rooms');
CALL object_stat('accommodations');
```

```
CREATE PROCEDURE
psql:select.sql:75: NOTICE:  Минимальное значение: 2000, Максимальное значение: 4500, Среднее значение: 2885
CALL
psql:select.sql:76: NOTICE:  Минимальное значение: 14, Максимальное значение: 37, Среднее значение: 25.333333333333332
CALL
psql:select.sql:77: NOTICE:  Минимальное значение: 112, Максимальное значение: 1200, Среднее значение: 494.5
CALL
```

**5) Создать хранимую процедуру objects_stat(…), которая подсчитывает минимальное, максимальное и среднее значение каждого объекта в дочерней таблице и выводит имя объекта, входным параметром является имя объекта из родительской таблицы:**

*Формирование статистики (минимальное, максимальное и среднее значение) для различных объектов (группы, комнаты)*:

```
CREATE OR REPLACE PROCEDURE objects_stat(object_name VARCHAR)
LANGUAGE plpgsql
AS $$
DECLARE
    rec RECORD;
BEGIN
    IF object_name = 'groups' THEN
        FOR rec IN
            SELECT g.group_name, MIN(s.scholarship) AS
min_scholarship,
                   MAX(s.scholarship) AS max_scholarship,
                   AVG(s.scholarship) AS avg_scholarship
            FROM public.groups g
            INNER JOIN public.students s ON g.group_id =
s.group_name
            GROUP BY g.group_name
        LOOP
            RAISE NOTICE 'Группа: %, Минимальная стипендия: %,
Максимальная стипендия: %, Средняя стипендия: %',
                         rec.group_name, rec.min_scholarship,
rec.max_scholarship, rec.avg_scholarship;
        END LOOP;
    ELSIF object_name = 'rooms' THEN
        FOR rec IN
```

```
            SELECT r.room_number, MIN(a.distance) AS min_distance,
                   MAX(a.distance) AS max_distance,
                   AVG(a.distance) AS avg_distance
            FROM public.rooms r
            INNER JOIN public.accommodations a ON r.room_id =
a.room_number
            GROUP BY r.room_number
        LOOP
            RAISE NOTICE 'Комната: %, Минимальное расстояние: %,
Максимальное расстояние: %, Среднее расстояние: %',
                         rec.room_number, rec.min_distance,
rec.max_distance, rec.avg_distance;
        END LOOP;
    END IF;
END;
$$;

CALL objects_stat('groups');
CALL objects_stat('rooms');
```

```
CREATE PROCEDURE
psql:select.sql:114: NOTICE:  Группа: CS-10, Минимальная стипендия: 2000, Максимальная стипендия: 4500, Средняя стипендия: 2913.3333333333333333
psql:select.sql:114: NOTICE:  Группа: CS-14, Минимальная стипендия: 2000, Максимальная стипендия: 4000, Средняя стипендия: 2856.6666666666666667
CALL
psql:select.sql:115: NOTICE:  Комната: 37, Минимальное расстояние: 435, Максимальное расстояние: 1200, Среднее расстояние: 817.5000000000000000
psql:select.sql:115: NOTICE:  Комната: 14, Минимальное расстояние: 200, Максимальное расстояние: 780, Среднее расстояние: 490.0000000000000000
psql:select.sql:115: NOTICE:  Комната: 25, Минимальное расстояние: 112, Максимальное расстояние: 240, Среднее расстояние: 176.0000000000000000
CALL
```

6) **Создать хранимую процедуру Itog(text) с одним входным параметром (имя объекта), которая выводит наименование объекта по суммарному количеству объектов и оценку:**
   a) **Оценка «Незначительный объект», если число объектов меньше 2**
   b) **Оценка «Обычный объект», если число объектов больше или равно 2 и меньше или равно 3**
   c) **Оценка «Значительный объект», если число объектов больше 3:**

*Формирование количества объектов и их оценки в зависимости от этого количества*:

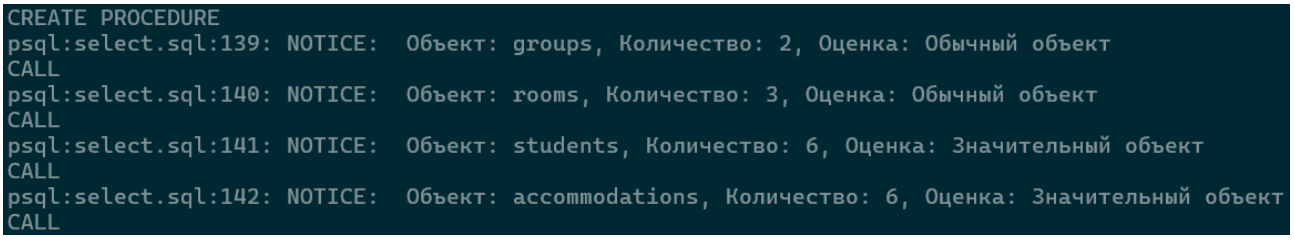
```
CREATE OR REPLACE PROCEDURE Itog(IN object_name TEXT)
LANGUAGE plpgsql
AS $$
DECLARE
    object_count INTEGER;
```

```
    evaluation TEXT;
BEGIN
    EXECUTE format('SELECT COUNT(*) FROM %I', object_name)
INTO object_count;

    IF object_count < 2 THEN
        evaluation := 'Незначительный объект';
    ELSIF object_count >= 2 AND object_count <= 3 THEN
        evaluation := 'Обычный объект';
    ELSE
        evaluation := 'Значительный объект';
    END IF;

    RAISE NOTICE 'Объект: %, Количество: %, Оценка: %',
object_name, object_count, evaluation;
END;
$$;

CALL Itog('groups');
CALL Itog('rooms');
CALL Itog('students');
CALL Itog('accommodations');
```




## ТРИГГЕРЫ

**7) Создать триггер After_Delete, который при удалении записи из родительской таблицы удалял бы все связанные записи из дочерней таблицы. Показать результат работы триггера:**

*Формирование удалений размещений студентов, связанных с удаляемой комнатой*:

Примечание: поскольку в 1-ой лаб. работе я поставил на ограничителе внешних ключей таблицы “размещений” флаг NO ACTION на ON DELETE, т.е. не производить никаких действий при этой операции, то я исправлю это значение на CASCADE,

чтобы я мог удалять записи из родительской и дочерней таблиц одновременно.

```
ALTER TABLE public.accommodations
     DROP CONSTRAINT room_room_number_fkey;
ALTER TABLE public.accommodations
     DROP CONSTRAINT student_student_name_fkey;
ALTER TABLE public.accommodations
     DROP CONSTRAINT neighbour_neighbour_name_fkey;

ALTER TABLE public.accommodations
     ADD CONSTRAINT room_room_number_fkey FOREIGN KEY
(room_number)
               REFERENCES public.rooms(room_id) MATCH SIMPLE
                    ON UPDATE NO ACTION
                    ON DELETE CASCADE;
ALTER TABLE public.accommodations
     ADD CONSTRAINT student_student_name_fkey FOREIGN KEY
(student_name)
               REFERENCES public.students(student_id) MATCH
SIMPLE
                    ON UPDATE NO ACTION
                    ON DELETE CASCADE;
ALTER TABLE public.accommodations
     ADD CONSTRAINT neighbour_neighbour_name_fkey FOREIGN KEY
(neighbour_name)
               REFERENCES public.students(student_id) MATCH
SIMPLE
                    ON UPDATE NO ACTION
                    ON DELETE CASCADE;

CREATE OR REPLACE FUNCTION delete_related_accommodations()
RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM accommodations
    WHERE room_number = OLD.room_id;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER after_delete_room
AFTER DELETE ON rooms
FOR EACH ROW EXECUTE FUNCTION
delete_related_accommodations();

SELECT * FROM accommodations;
DELETE FROM rooms WHERE room_id = 3;
SELECT * FROM accommodations;
```

```
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
CREATE FUNCTION
CREATE TRIGGER
 accommodation_id | accommodation_date | distance | room_number | student_name | neighbour_name
------------------+--------------------+----------+-------------+--------------+----------------
                1 | 2005-08-03         |      200 |           1 |            1 |              6
                2 | 2005-08-15         |      435 |           2 |            2 |              5
                3 | 2005-08-02         |      112 |           3 |            3 |              4
                4 | 2005-08-02         |      240 |           3 |            4 |              3
                5 | 2005-08-14         |     1200 |           2 |            5 |              2
                6 | 2005-08-04         |      780 |           1 |            6 |              1
(6 rows)

DELETE 1
 accommodation_id | accommodation_date | distance | room_number | student_name | neighbour_name
------------------+--------------------+----------+-------------+--------------+----------------
                1 | 2005-08-03         |      200 |           1 |            1 |              6
                2 | 2005-08-15         |      435 |           2 |            2 |              5
                5 | 2005-08-14         |     1200 |           2 |            5 |              2
                6 | 2005-08-04         |      780 |           1 |            6 |              1
(4 rows)
```

**8) Создать триггер Before_Delete, который при удалении записи из дочерней таблицы выводил бы имя объекта родительской таблицы. Показать результат работы триггера:**

*Формирование уведомлений перед удалением размещения студента*:

```
CREATE OR REPLACE FUNCTION
notify_before_delete_accommodation()
RETURNS TRIGGER AS $$
BEGIN
    RAISE NOTICE 'Удаляется размещение студента: %',
OLD.accommodation_id;

    IF EXISTS (SELECT 1 FROM rooms WHERE room_id =
OLD.room_number) THEN
```

```
        RAISE NOTICE 'Удаляется комната с номером: %',
OLD.room_number;
    END IF;

    IF EXISTS (SELECT 1 FROM students WHERE student_id =
OLD.student_name) THEN
        RAISE NOTICE 'Удаляется студент с ID: %',
OLD.student_name;
    END IF;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER before_delete_accommodation
BEFORE DELETE ON accommodations
FOR EACH ROW EXECUTE FUNCTION
notify_before_delete_accommodation();

SELECT * FROM accommodations;
DELETE FROM accommodations WHERE accommodation_id = 1;
SELECT * FROM accommodations;
```

```
CREATE FUNCTION
CREATE TRIGGER
 accommodation_id | accommodation_date | distance | room_number | student_name | neighbour_name
------------------+--------------------+----------+-------------+--------------+----------------
                1 | 2005-08-03         |      200 |           1 |            1 |              6
                2 | 2005-08-15         |      435 |           2 |            2 |              5
                5 | 2005-08-14         |     1200 |           2 |            5 |              2
                6 | 2005-08-04         |      780 |           1 |            6 |              1
(4 rows)

psql:select.sql:209: NOTICE:  Удаляется размещение студента: 1
psql:select.sql:209: NOTICE:  Удаляется комната с номером: 1
psql:select.sql:209: NOTICE:  Удаляется студент с ID: 1
DELETE 1
 accommodation_id | accommodation_date | distance | room_number | student_name | neighbour_name
------------------+--------------------+----------+-------------+--------------+----------------
                2 | 2005-08-15         |      435 |           2 |            2 |              5
                5 | 2005-08-14         |     1200 |           2 |            5 |              2
                6 | 2005-08-04         |      780 |           1 |            6 |              1
(3 rows)
```

**9) Создать триггер ins_sum, который связывает триггер с таблицей для инструкций INSERT. Это действует как сумматор, чтобы суммировать значения, вставленные в один из столбцов дочерней таблицы. Триггер должен**

**активироваться перед каждой строкой, вставленной в таблицу. Показать результат работы триггера:**

*Формирование суммы значений дистанции от общежития при добавлении новых размещений*:

```
CREATE OR REPLACE FUNCTION sum_distance()
RETURNS TRIGGER AS $$
BEGIN
     NEW.distance := NEW.distance + (SELECT
COALESCE(SUM(distance), 0)
     FROM accommodations);
     RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER ins_sum
BEFORE INSERT ON accommodations
FOR EACH ROW
EXECUTE FUNCTION sum_distance();

SELECT * FROM accommodations;
INSERT INTO accommodations (accommodation_date, distance,
room_number, student_name, neighbour_name) VALUES
        ('2005-08-20', 300, 1, 1, 2);
SELECT * FROM accommodations;
```

```
CREATE FUNCTION
CREATE TRIGGER
 accommodation_id | accommodation_date | distance | room_number | student_name | neighbour_name
------------------+--------------------+----------+-------------+--------------+----------------
                2 | 2005-08-15         |      435 |           2 |            2 |              5
                5 | 2005-08-14         |     1200 |           2 |            5 |              2
                6 | 2005-08-04         |      780 |           1 |            6 |              1
(3 rows)

INSERT 0 1
 accommodation_id | accommodation_date | distance | room_number | student_name | neighbour_name
------------------+--------------------+----------+-------------+--------------+----------------
                2 | 2005-08-15         |      435 |           2 |            2 |              5
                5 | 2005-08-14         |     1200 |           2 |            5 |              2
                6 | 2005-08-04         |      780 |           1 |            6 |              1
                7 | 2005-08-20         |     2715 |           1 |            1 |              2
(4 rows)
```

**10) Создать триггер Before_Update_Value на событие UPDATE, который увеличивает значение числового поля дочерней таблицы на 10%. Показать результат работы триггера:**

*Формирование увеличения стипендии студентов на 10% перед обновлением*:

```
CREATE OR REPLACE FUNCTION before_update_value()
RETURNS TRIGGER AS $$
BEGIN
    NEW.scholarship := NEW.scholarship * (1.0 + (10.0 /
100.0));
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER Before_Update_Value
BEFORE UPDATE ON students
FOR EACH ROW
EXECUTE FUNCTION before_update_value();

UPDATE students SET scholarship = 2000 WHERE student_id = 1;

SELECT * FROM students WHERE student_id = 1;
```

```
CREATE FUNCTION
CREATE TRIGGER
UPDATE 1
 student_id | student_name | scholarship | group_name
------------+--------------+-------------+------------
          1 | Vasilyev     |        2200 |          1
(1 row)
```