

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «Процедуры, функции, триггеры в PostgreSQL»
по дисциплине «Проектирование и реализация баз данных»

Автор: Берулава Л.А.

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы:.....	3
Практическое задание.....	3
Выполнение.....	3
Вывод.....	8

Цель работы: овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

Вариант 2 (max - 8 баллов)

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
2. Создать авторский триггер по варианту индивидуального задания.

Выполнение

Создайте хранимые процедуры:

1. Для повышения оклада сотрудников, выполнивших задания с трехдневным опережением графика на заданный процент.

```
CREATE OR REPLACE PROCEDURE up_oklad(percentage integer)
```

```
LANGUAGE plpgsql as
```

```
$$
```

```
BEGIN
```

```
    UPDATE Сотрудник_Отдела
```

```
    SET dolya_stavki = dolya_stavki + percentage / 100
```

```
    WHERE id_sotrudnika IN (
```

```
        SELECT sp.id_sotrudnika
```

```
        FROM Сотрудник_Проекта AS sp
```

```
        INNER JOIN Задания AS t ON sp.id_project = t.id_project
```

```
        INNER JOIN Выплата AS v ON sp.id_sotrudnika_project = v.id_sotrudnika_project
```

```
        WHERE t.end_date < CURRENT_DATE + 3 AND t.status = 'Finished'
```

```
    );
```

```
END;
```

```
$$;
```

```

CREATE PROCEDURE
berulaa=# CREATE OR REPLACE PROCEDURE up_oklad(precentage integer)
berulaa=# LANGUAGE plpgsql as
berulaa=# $$
berulaa=# BEGIN
berulaa=# UPDATE Сотрудник_Отдела
berulaa=#     SET dolya_stavki = dolya_stavki + percentage / 100
berulaa=#     WHERE id_sotrudnika IN (
berulaa=#         SELECT sp.id_sotrudnika
berulaa=#         FROM Сотрудник_Проекта AS sp
berulaa=#         INNER JOIN Задания AS t ON sp.id_project = t.id_project
berulaa=#     INNER JOIN Выплата AS v ON sp.id_sotrudnika_project = v.id_sotrudnika_project
berulaa=#         WHERE t.end_date < date_vyplaty + 3 AND t.status = 'Finished'
berulaa=#     );
berulaa=# END;
berulaa=# $$;
CREATE PROCEDURE

```

```

berulaa=# CREATE OR REPLACE PROCEDURE up_oklad(precentage integer)
berulaa=# LANGUAGE plpgsql as
berulaa=# $$
berulaa=# BEGIN
berulaa=# UPDATE Сотрудник_Отдела
berulaa=#     SET dolya_stavki = dolya_stavki + round(precentage/100.0, 2)
berulaa=#     WHERE id_sotrudnika_otdela IN (
berulaa=#         SELECT sp.id_sotrudnika FROM Сотрудник_Проекта AS sp
berulaa=#         INNER JOIN Задания AS t ON sp.id_project = t.id_project
berulaa=#         WHERE t.end_date - 3 >= CURRENT_DATE AND t.status = 'Finished'
berulaa=#     );
berulaa=# END;
berulaa=# $$;
CREATE PROCEDURE
berulaa=# SELECT * FROM Сотрудник_Отдела;

```

id_otdela	id_dolzhnosti	id_sotrudnika	dolya_stavki	hours_of_work	status	id_sotrudnika_otdela
4	1	4	1.00	8	Active	4
1	2	5	1.00	8	Active	5
2	3	6	1.00	8	Active	6
3	1	7	1.00	8	Active	7
4	2	8	1.00	8	Active	8
1	3	9	1.00	8	Active	9
1	1	1	1.00	8	Active	1
2	1	10	1.00	8	Active	10
2	2	2	1.00	8	Active	2
3	3	3	1.04	8	Active	3

```

(10 rows)

berulaa=# CALL up_oklad(4);
CALL
berulaa=# SELECT * FROM Сотрудник_Отдела;

```

id_otdela	id_dolzhnosti	id_sotrudnika	dolya_stavki	hours_of_work	status	id_sotrudnika_otdela
4	1	4	1.00	8	Active	4
1	2	5	1.00	8	Active	5
2	3	6	1.00	8	Active	6
3	1	7	1.00	8	Active	7
4	2	8	1.00	8	Active	8
1	3	9	1.00	8	Active	9
1	1	1	1.00	8	Active	1
2	1	10	1.00	8	Active	10
2	2	2	1.00	8	Active	2
3	3	3	1.08	8	Active	3

```

(10 rows)

```

2. Для вычисления количества проектов, в выполнении которых участвует сотрудник.

```

CREATE OR REPLACE FUNCTION employee_count_project_fun1(employee_id bigint)
RETURNS TABLE(
    id_sotrudnika_project bigint,
    project_count integer
) AS
$$
BEGIN

```

```

RETURN QUERY
SELECT sp.id_sotrudnika_project, COUNT(DISTINCT id_project) as project_count
FROM Сотрудник_Проекта sp
WHERE id_sotrudnika = employee_id
GROUP BY sp.id_sotrudnika_project;
END;
$$
LANGUAGE plpgsql;

```

```

berulaa=# CREATE OR REPLACE FUNCTION employee_count_project_fun4(employee_id bigint)
berulaa=# RETURNS TABLE(
berulaa(#      id_sotrudnika bigint,
berulaa(#      project_count bigint
berulaa(# ) AS
berulaa=# $$
berulaa$# BEGIN
berulaa$#     RETURN QUERY
berulaa$#     SELECT sp.id_sotrudnika, COUNT(DISTINCT id_project) as project_count
berulaa$#     FROM Сотрудник_Проекта sp
berulaa$#     WHERE sp.id_sotrudnika = employee_id
berulaa$# GROUP BY sp.id_sotrudnika;
berulaa$# END;
berulaa$# $$
berulaa=# LANGUAGE plpgsql;
CREATE FUNCTION

```

```

[berulaa=# SELECT * FROM employee_count_project_fun4(3)
  id_sotrudnika | project_count
-----+-----
              3 |              3
(1 row)

```

```

berulaa=# CREATE OR REPLACE PROCEDURE employee_count_project(IN employee_id bigint, OUT project_count bigint)
berulaa=# LANGUAGE plpgsql as
berulaa=# $$
berulaa$# BEGIN
berulaa$# SELECT COUNT(DISTINCT id_project) INTO project_count
berulaa$#      FROM Сотрудник_Проекта
berulaa$#      WHERE id_sotrudnika = employee_id;
berulaa$# END;
berulaa$# $$;
CREATE PROCEDURE
berulaa=# CALL employee_count_project(3, null);
project_count
-----
3
(1 row)

```

3. Для поиска номера телефона сотрудника (телефон установлен в каждом отделе).

```

CREATE OR REPLACE PROCEDURE find_phone_number(IN employee_id bigint, OUT
phone_number VARCHAR(20))
LANGUAGE plpgsql as
$$
BEGIN
    SELECT contacts INTO phone_number FROM Сотрудник
    WHERE Сотрудник.id_sotrudnika = employee_id;
END;
$$;

```

```

berulaa=# CREATE OR REPLACE PROCEDURE find_phone_number(IN employee_id bigint, OUT phone_number VARCHAR(20))
berulaa=# LANGUAGE plpgsql as
berulaa=# $$
berulaa$# BEGIN
berulaa$# SELECT contacts INTO phone_number FROM Сотрудник
berulaa$#      WHERE Сотрудник.id_sotrudnika = employee_id;
berulaa$# END;
berulaa$# $$;
CREATE PROCEDURE
berulaa=# CALL find_phone_number(7, null);
phone_number
-----
+7 (565) 759-02-51

```

Создайте необходимые триггеры:

- Триггер для создания выплаты сотрудникам проекта за выполненное задание

```

CREATE OR REPLACE FUNCTION process_finished_task()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.status = 'Finished' THEN
        INSERT INTO Выплата (id_sotrudnika_project, razmer_vyplaty, date_vyplaty)
        VALUES (NEW.id_sotrudnika_project, NEW.payment, CURRENT_DATE);
    END IF;
    RETURN NEW;
END;
$$ language plpgsql;

```

```
CREATE OR REPLACE TRIGGER task_finished_trigger
AFTER UPDATE OF status ON Задания
FOR EACH ROW
EXECUTE FUNCTION process_finished_task();
```

```
berulaa=# CREATE OR REPLACE FUNCTION process_finished_task()
berulaa=# RETURNS TRIGGER AS $$
berulaa=# BEGIN
berulaa=#     IF NEW.status = 'Finished' THEN
berulaa=#         INSERT INTO Выплата (id_sotrudnika_project, razmer_vyplaty, date_vyplaty)
berulaa=#         VALUES (NEW.id_sotrudnika_project, NEW.payment, CURRENT_DATE);
berulaa=#     END IF;
berulaa=#     RETURN NEW;
berulaa=# END;
berulaa=# $$ language plpgsql;
CREATE FUNCTION
berulaa=#
berulaa=# CREATE OR REPLACE TRIGGER task_finished_trigger
berulaa=# AFTER UPDATE OF status ON Задания
berulaa=# FOR EACH ROW
berulaa=# EXECUTE FUNCTION process_finished_task();
CREATE TRIGGER
berulaa=#
berulaa=# SELECT * FROM Выплата;
  id_payment | id_sotrudnika_project | razmer_vyplaty | date_vyplaty
-----+-----+-----+-----
(0 rows)
```

```
berulaa=# UPDATE Задания SET status = 'Finished' WHERE id_task = 5;
UPDATE 1
berulaa=# SELECT * FROM Выплата;
  id_payment | id_sotrudnika_project | razmer_vyplaty | date_vyplaty
-----+-----+-----+-----
          1 |                    4 |         11000 | 2023-12-24
          2 |                    3 |         11000 | 2023-12-24
          3 |                    2 |         11000 | 2023-12-24
          4 |                    1 |         11000 | 2023-12-24
(4 rows)

berulaa=#
```

Вывод

В ходе лабораторной работы была освоена работа с процедурами и триггерами.