

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

**Отчет**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**  
**«ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ,  
ПРЕДСТАВЛЕНИЯ И ИНДЕКСЫ В POSTGRESQL»**

Автор: Кононов Степан Владимирович

Факультет: ИКТ

Группа: K32392

Преподаватель: Говорова М. М.

Дата: 02.05.2023

**ИТМО**

Санкт-Петербург 2023

# Лабораторная работа №3

## Процедуры/функции

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

```
select i.id_employee, salary::numeric
from task
      join implementation i on task.id = i.id_task
      join staff_member sm on sm.id = i.id_employee
      join employee_position on sm.id_job_title = employee_position.id
where EXTRACT(DAY FROM (DATE_TRUNC('day', task.due_date_to) - DATE_TRUNC('day', actual_completion_date))) >= 3
```

	id_employee ÷	salary ÷
1	983	302.4
2	949	302.4

Зарплаты сотрудников до повышения.

```
CREATE OR REPLACE FUNCTION increase_salaries_for_employees_ahead_of_schedule(increase_percent FLOAT) RETURNS VOID AS
$$
BEGIN
    UPDATE employee_position
    SET salary = ep.salary * (1 + increase_percent)
    FROM employee_position AS ep
         JOIN staff_member sm ON ep.id = sm.id_job_title
    WHERE id_employee IN (SELECT sm.id_employee
                        FROM task
                             JOIN implementation i ON task.id = i.id_task
                             JOIN staff_member sm ON sm.id = i.id_employee
                        WHERE EXTRACT(DAY FROM (DATE_TRUNC('day', task.due_date_to) -
                                                DATE_TRUNC('day', actual_completion_date))) >= 3);
END;
$$ LANGUAGE plpgsql;
select increase_salaries_for_employees_ahead_of_schedule(0.5);
```

	id_employee ÷	salary ÷
1	983	453.6
2	949	453.6

Зарплаты после повышения на 50%

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

```
CREATE OR REPLACE FUNCTION get_the_number_of_employee_projects(employee_id INT)
    RETURNS INT AS
$$
DECLARE
    count INT;
BEGIN
    SELECT COUNT(Distinct(p.id))::INT
    INTO count
    FROM employee
        JOIN staff_member sm ON employee.id = sm.id_employee
        JOIN implementation i ON sm.id = i.id_employee
        JOIN task t ON i.id_task = t.id
        JOIN project p ON p.id = t.id_project
    WHERE employee.id = employee_id
    GROUP BY employee.id;
    RETURN count;
END;
$$ LANGUAGE plpgsql;
select get_the_number_of_employee_projects(81) as project_number;
```

	project_number ÷
1	5

Сотрудник с id = 81 работает над 5 проектами

- Для поиска номера телефона сотрудника (телефон установлен в каждом отделе). Поскольку при составлении базы данных телефон указывался сразу для сотрудника, то мы ищем номер отдела в котором работает сотрудник.

```
CREATE OR REPLACE FUNCTION get_dep_number_for_employee(employee_id INT)
    RETURNS VARCHAR(12) AS
```

```

$$
DECLARE
    dep_number VARCHAR(12);
BEGIN
    SELECT d.telephone
    INTO dep_number
    FROM employee
        JOIN staff_member sm ON employee.id = sm.id_employee
        JOIN department d ON d.id = sm.id_department
    WHERE employee.id = employee_id;
    RETURN dep_number;
END;
$$
LANGUAGE plpgsql;
select get_dep_number_for_employee(81) as department_number;

```

department_number	
1	+33285528638

Номер отдела в котором работает сотрудник с id = 81

## Триггер

- Проверяем корректность постановки задачи.
  1. Задачу можно поставить для конкретного проекта, только если проект еще не завершен.
  2. Сроки выполнения задачи должны укладываться в сроки выполнения проекта.

```

CREATE OR REPLACE FUNCTION check_task_insert()
RETURNS TRIGGER AS
$$
BEGIN
    IF NOT EXISTS (
        SELECT 1
        FROM project
        WHERE id = NEW.id_project
        AND execution_status = 'In progress'
    ) THEN
        RAISE EXCEPTION 'Cannot insert task to project that is not in progress';
    END IF;

    IF EXISTS (

```

```

        SELECT 1
        FROM project
        WHERE id = NEW.id_project
        AND (NEW.due_date_from < due_date_from OR NEW.due_date_to > due_date_to)
    ) THEN
        RAISE EXCEPTION 'Task dates must be between project dates';
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER task_insert_trigger
BEFORE INSERT ON task
FOR EACH ROW
EXECUTE FUNCTION check_task_insert();

```

## Пример работы триггера

Возьмем проект id = 5. Он имеет статус “Completed successfully”

```

select id, title, execution_status
from project
where id = 55

```

	id	title	execution_status
1	55	Adipisci dolore modi magnam ut quiquia.	Completed successfully

Попытаемся добавить к нему задачу.

```

insert into task(id_project, comment, price, due_date_from, due_date_to, execution_status,
actual_completion_date)
values (55, 'Eius.', 58043.00, '2002-08-31', '2019-09-28', 'Not started', NULL)

```

Получаем ошибку

**[P0001] ОШИБКА: Cannot insert task to project that is not in progress**  
 Где: функция PL/pgSQL check\_task\_insert(), строка 9, оператор RAISE

Возьмем прок id = 10.

```
select id, execution_status, due_date_from, due_date_to
from project
where id = 10;
```

	id	execution_status	due_date_from	due_date_to
1	10	In progress	2009-01-24	2009-05-10

Попытаемся добавить к нему задачу, которая не вписывается во временные рамки проекта.

```
insert into task(id_project, comment, price, due_date_from, due_date_to, execution_status,
actual_completion_date)
values (10, 'Eius.', 58043.00, '2002-08-31', '2019-09-28', 'Not started', NULL)
```

Получаем ошибку

[P0001] ОШИБКА: Task dates must be between project dates  
Где: функция PL/pgSQL check\_task\_insert(), строка 18, оператор RAISE

## Вывод

В результате выполнения лабораторной работы были достигнуты следующие цели:

- Овладение практическими навыками создания процедур, функций и триггеров в базе данных PostgreSQL
- Создание процедур и функций в соответствии с индивидуальным заданием, что позволило эффективно реализовать определенную логику обработки данных
- Создание триггеров для логирования событий вставки, что позволило улучшить контроль за изменениями в базе данных и обеспечить более точную отчетность.

Таким образом, выполнение лабораторной работы позволило успешно освоить необходимые навыки работы с базой данных PostgreSQL и использовать их для решения задач по обработке, хранению и анализу данных.