**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

ПО ПРАКТИЧЕСКОМУ ЗАДАНИЮ  
 «Лабораторная работа №3»

Автор: Митурский Богдан Антонович

Факультет: ИКТ

Группа: K32392

Преподаватель: Говорова М. М.



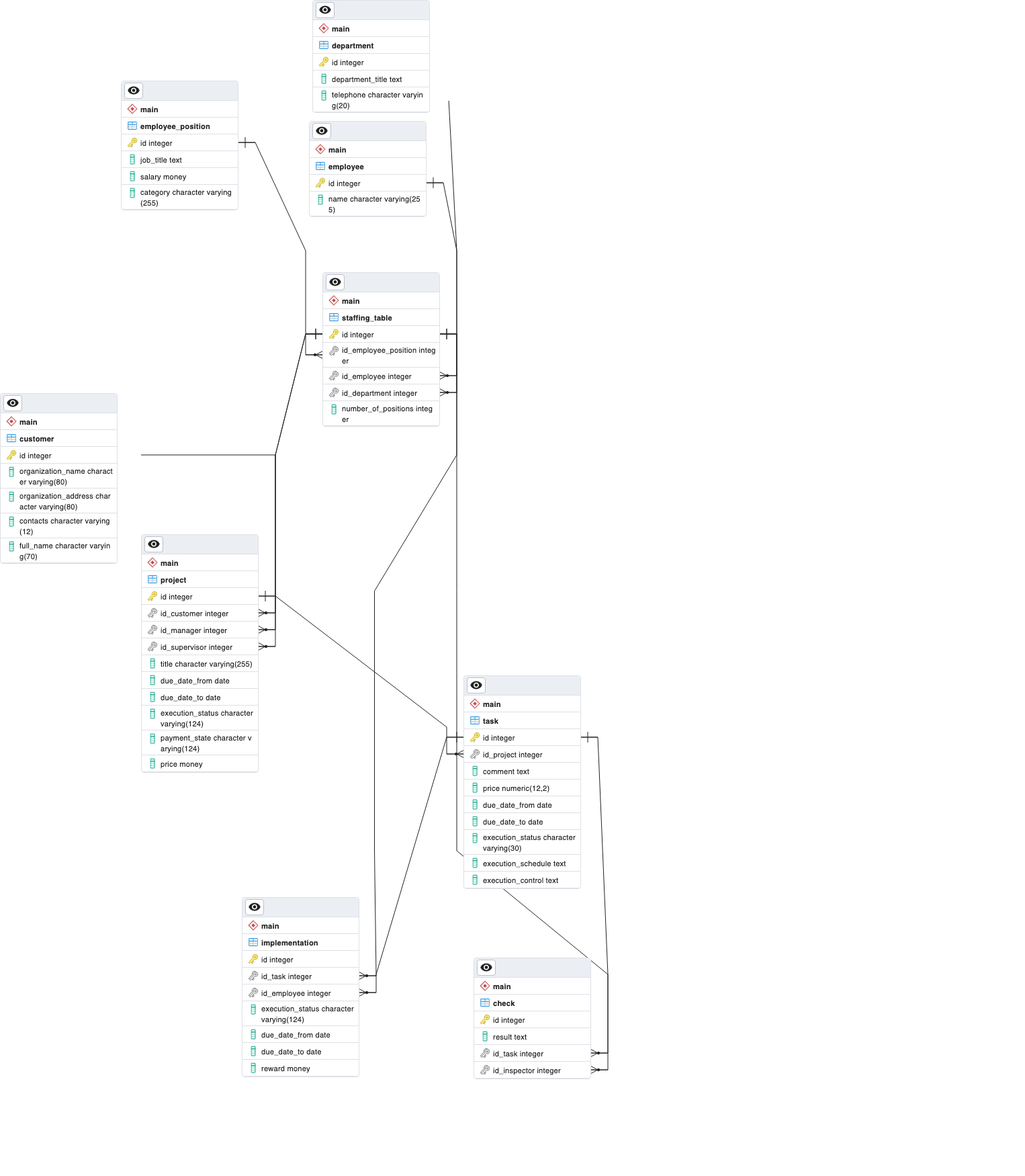
Санкт-Петербург 2023

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

**Практическое задание (Вариант 1):**

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

*Схема логической модели базы данных, сгенерированная в Generate ERD:*

**

**Выполнение (Процедуры/Функции):**

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

**SELECT** st.id\_employee\_position, e.name, ep.salary, ep.job\_title

**FROM** main.task

**JOIN** main.implementation **AS** im **ON** main.task.**id** = im.id\_task

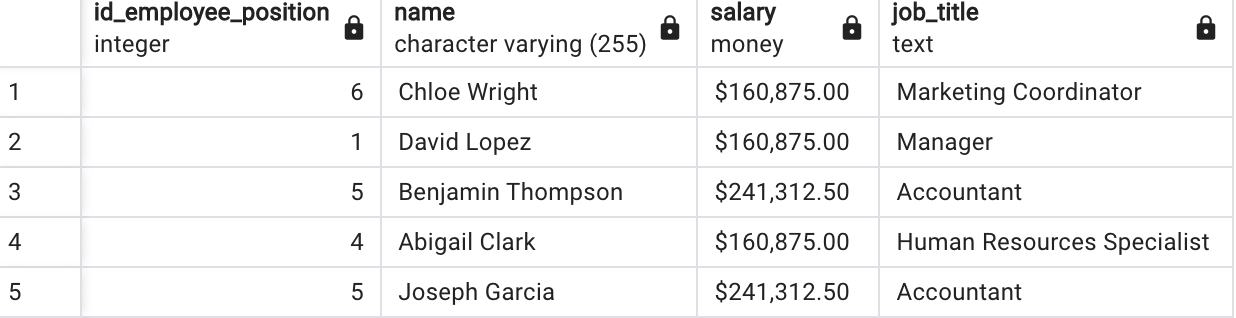
**JOIN** main.staffing\_table **AS** st **ON** st.**id** = im.id\_employee

**JOIN** main.employee **AS** e **ON** st.id\_employee = e.**id**

**JOIN** main.employee\_position **AS** ep **ON** ep.**id** = st.id\_employee\_position

**WHERE** im.execution\_status **LIKE** '%Completed%'

**AND** **EXTRACT**(**DAY** **FROM** AGE(main.task.due\_date\_to, im.due\_date\_to)) > 3



**CREATE** **OR** **REPLACE** **FUNCTION** increase\_salaries\_for\_employees\_ahead\_of\_schedule(increase\_percent **float**)

**RETURNS** VOID **AS** $$

**BEGIN**

**UPDATE** main.employee\_position

**SET** salary = salary \* (1 + increase\_percent)

**WHERE** **id** **IN** (

**SELECT** st.id\_employee\_position

**FROM** main.task

**JOIN** main.implementation **AS** im **ON** main.task.**id** = im.id\_task

**JOIN** main.staffing\_table **AS** st **ON** st.**id** = im.id\_employee

**JOIN** main.employee **AS** e **ON** st.id\_employee = e.**id**

**JOIN** main.employee\_position **AS** ep **ON** ep.**id** = st.id\_employee\_position

**WHERE** im.execution\_status **LIKE** '%Completed%'

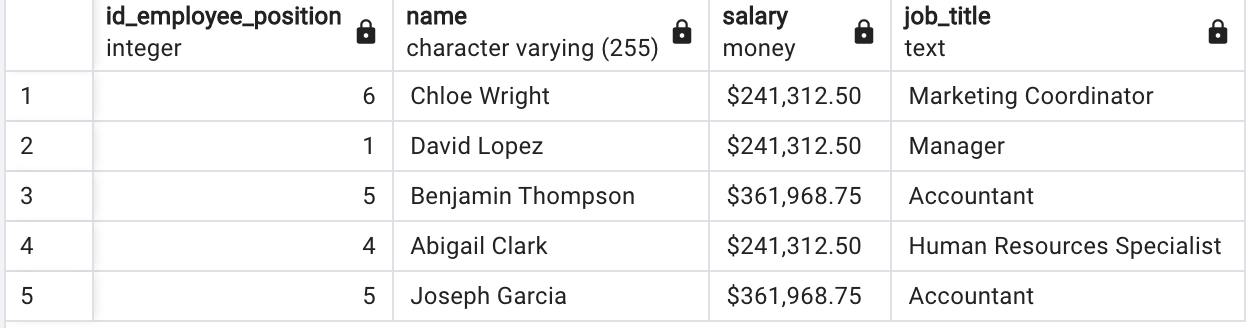
**AND** **EXTRACT**(**DAY** **FROM** AGE(main.task.due\_date\_to, im.due\_date\_to)) > 3

);

**END**

$$ **LANGUAGE** plpgsql;

**SELECT** increase\_salaries\_for\_employees\_ahead\_of\_schedule(0.5);

****

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

**CREATE** **OR** **REPLACE** **FUNCTION** main.get\_the\_number\_of\_employee\_projects(employee\_id **INT**)

**RETURNS** **INT** **AS** $$

**BEGIN**

**RETURN** (

**SELECT** **COUNT**(**DISTINCT** main.task.id\_project)

**FROM** main.implementation

**JOIN** main.task **ON** main.task.**id** = main.implementation.id\_task

**JOIN** main.staffing\_table **ON** main.staffing\_table.**id** = main.implementation.id\_employee

**WHERE** main.staffing\_table.id\_employee = employee\_id

);

**END**

$$ **LANGUAGE** plpgsql;

**SELECT** get\_the\_number\_of\_employee\_projects(13) **AS** project\_number;

****

1. Для поиска номера телефона сотрудника (телефон находится в каждом отделе)

**CREATE** **OR** **REPLACE** **FUNCTION** get\_employee\_phone(employee\_id **INT**)

**RETURNS** **VARCHAR** **AS** $$

**DECLARE**

phone **VARCHAR**;

**BEGIN**

**SELECT** **INTO** phone main.department.telephone

**FROM** main.department

**JOIN** main.staffing\_table **ON** main.staffing\_table.id\_department = main.department.**id**

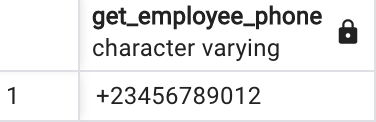
**WHERE** main.staffing\_table.**id** = employee\_id;

**RETURN** phone;

**END**

$$ **LANGUAGE** plpgsql;

**SELECT** get\_employee\_phone(2);

****

**Выполнение (Триггеры на логирование):**

1. Создадим триггеры для логирования действий в таблицах. Для начала создадим таблицу, которая будет выступать в роли журнала.

**CREATE** **TABLE** main.db\_log (

**id** **SERIAL** **PRIMARY** **KEY**,

operation **VARCHAR**(50) **NOT** **NULL**,

**timestamp** **TIMESTAMP** **WITHOUT** **TIME** **ZONE** **DEFAULT** (now() **AT** **TIME** **ZONE** 'utc'),

tablename **VARCHAR**(255),

old\_data TEXT,

new\_data TEXT

);

1. Создадим функцию записи в которую тригеры будут передавать данные.

**CREATE** **OR** **REPLACE** **FUNCTION** main.log\_changes() **RETURNS** **TRIGGER** **AS** $$

**BEGIN**

**IF** (TG\_OP = 'DELETE') **THEN**

**INSERT** **INTO** main.db\_log(operation, tablename, old\_data) **VALUES** (TG\_OP, TG\_TABLE\_NAME, **old**::text);

**RETURN** **old**;

**ELSIF** (TG\_OP = 'UPDATE') **THEN**

**INSERT** **INTO** main.db\_log(operation, tablename, old\_data, new\_data) **VALUES** (TG\_OP, TG\_TABLE\_NAME, **old**::text, **new**::text);

**RETURN** **new**;

**ELSIF** (TG\_OP = 'INSERT') **THEN**

**INSERT** **INTO** main.db\_log(operation, tablename, new\_data) **VALUES** (TG\_OP, TG\_TABLE\_NAME, **new**::text);

**RETURN** **new**;

**END IF**;

**RETURN** **NULL**;

**END**;

$$ **LANGUAGE** plpgsql;

1. Инициализируем тригеры для каждой таблицы в БД.

**CREATE** **TRIGGER** log\_task\_changes

**AFTER** **INSERT** **OR** **UPDATE** **OR** **DELETE** **ON** main.task

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** main.log\_changes();

**CREATE** **TRIGGER** log\_project\_changes

**AFTER** **INSERT** **OR** **UPDATE** **OR** **DELETE** **ON** main.project

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** main.log\_changes();

**CREATE** **TRIGGER** log\_employee\_changes

**AFTER** **INSERT** **OR** **UPDATE** **OR** **DELETE** **ON** main.employee

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** main.log\_changes();

**CREATE** **TRIGGER** log\_employee\_position\_changes

**AFTER** **INSERT** **OR** **UPDATE** **OR** **DELETE** **ON** main.employee\_position

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** main.log\_changes();

**CREATE** **TRIGGER** log\_implementation\_changes

**AFTER** **INSERT** **OR** **UPDATE** **OR** **DELETE** **ON** main.implementation

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** main.log\_changes();

**CREATE** **TRIGGER** log\_staffing\_table\_changes

**AFTER** **INSERT** **OR** **UPDATE** **OR** **DELETE** **ON** main.staffing\_table

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** main.log\_changes();

**CREATE** **TRIGGER** log\_department\_changes

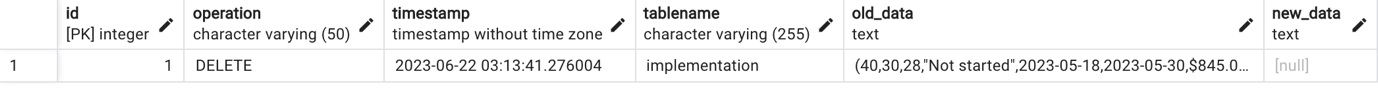
**AFTER** **INSERT** **OR** **UPDATE** **OR** **DELETE** **ON** main.department

**FOR** **EACH** **ROW** **EXECUTE** **PROCEDURE** main.log\_changes();

1. Для проверки удалим выполнение с id 40 из main.implements.

**DELETE** **FROM** main.implements **WHERE** **id** = 40

1. Проверим триггер с помощью SELECT \* FROM main.db\_log



Триггер работает, данные логируются.

**Вывод**

В результате выполнения лабороторной работы были реализованы процедуры/функции, что позволило ознакомиться с данным функционалом и глубже погрузиться в написание сложных запросов и работу с pgAdmin. А также был реализован триггер логирования, что повлекло за собой понимание необходимости и важности триггеров. Реализация триггера логирования вовлекла в процесс и заставила задуматься об интересе реализации других триггеров. 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.