

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: К32392

Преподаватель: Говорова М. М.

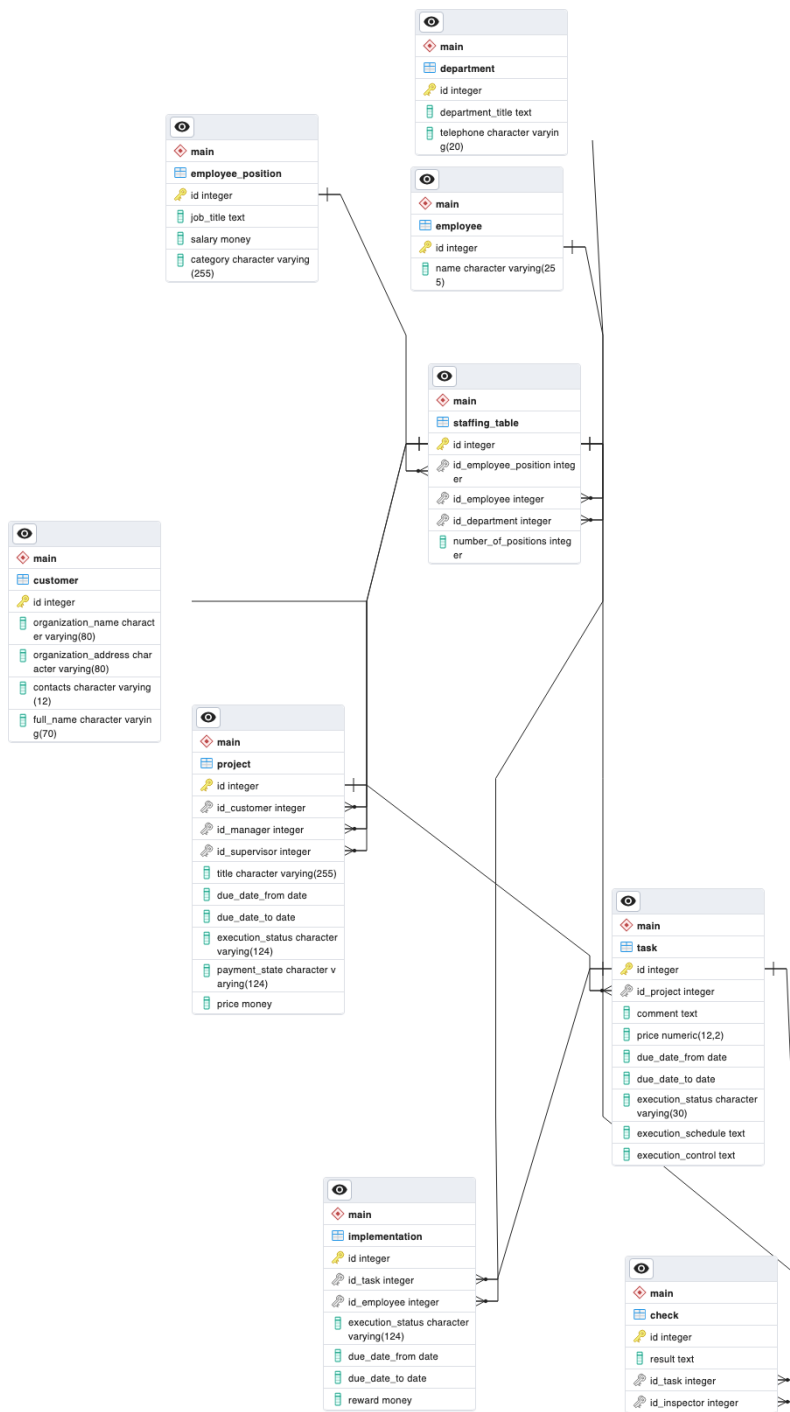
ИТМО
Санкт-Петербург 2023

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
```

	id_employee_position integer	name character varying (255)	salary money	job_title text
1	6	Chloe Wright	\$160,875.00	Marketing Coordinator
2	1	David Lopez	\$160,875.00	Manager
3	5	Benjamin Thompson	\$241,312.50	Accountant
4	4	Abigail Clark	\$160,875.00	Human Resources Specialist
5	5	Joseph Garcia	\$241,312.50	Accountant

```
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);
```

	id_employee_position integer	name character varying (255)	salary money	job_title text
1	6	Chloe Wright	\$241,312.50	Marketing Coordinator
2	1	David Lopez	\$241,312.50	Manager
3	5	Benjamin Thompson	\$361,968.75	Accountant
4	4	Abigail Clark	\$241,312.50	Human Resources Specialist
5	5	Joseph Garcia	\$361,968.75	Accountant

2. 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;

```

	project_number integer
1	3


3. 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);

```

	get_employee_phone character varying 
1	+23456789012

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
);
```

2. Создадим функцию записи в которую триггеры будут передавать данные.

```
CREATE OR REPLACE FUNCTION main.log_changes() RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO main.db_log(operation, tablename, old_data)
VALUES (TG_OP, TG_TABLE_NAME, old::text);
        RETURN old;
    ELIF (TG_OP = 'UPDATE') THEN
        INSERT INTO main.db_log(operation, tablename, old_data,
new_data) VALUES (TG_OP, TG_TABLE_NAME, old::text, new::text);
        RETURN new;
    ELIF (TG_OP = 'INSERT') THEN
        INSERT INTO main.db_log(operation, tablename, new_data)
VALUES (TG_OP, TG_TABLE_NAME, new::text);
        RETURN new;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

3. Инициализируем триггеры 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();

```

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

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

5. Проверим триггер с помощью `SELECT * FROM main.db_log`

	id [PK] integer	operation character varying (50)	timestamp timestamp without time zone	tablename character varying (255)	old_data text	new_data text
1	1	DELETE	2023-06-22 03:13:41.276004	implementation	(40,30,28,"Not started",2023-05-18,2023-05-30,\$845.0...	[null]

Триггер работает, данные логируются.

Вывод

В результате выполнения лабораторной 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.