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»

Обучающийся Скобликов Кирилл Александрович
Факультет прикладной информатики
Группа К3239
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-
Петербург
2024/2025

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

Практическое задание:

1. Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)
2. Создать триггеры для индивидуальной БД согласно варианту:
   Вариант 2.1. 3 триггера - 3 балла (min). Допустимо использовать триггеры логирования из практического занятия по функциям и триггерам.
3. Модифицировать триггер (триггерную функцию) из практической работы на проверку корректности входа и выхода сотрудника

Выполнение:

Созданы 3 процедуры, согласно варианту:

```
Library=# CREATE OR REPLACE PROCEDURE find_book_count(book_name varchar(150), book_author varchar(150), OUT result int)
Library-# LANGUAGE plpgsql AS $$
Library$# BEGIN
Library$#       SELECT COUNT(*) INTO result FROM "Library"."Copy_book" AS CB LEFT JOIN "Library"."Book" AS B ON
Library$#               CB.book_id = B.book_id
Library$#       WHERE B.name = book_name AND B.author = book_author
Library$# ;
Library$# END;
Library$# $$;
CREATE PROCEDURE
```

```
Library=# CREATE OR REPLACE PROCEDURE new_book(book_author varchar(150), book_name varchar(150), volume_number int,
Library(# compiled_by varchar(200), original_language varchar(50), translator varchar(200), year_of_issue date)
Library-# LANGUAGE plpgsql AS $$
Library$# BEGIN
Library$#       INSERT INTO "Library"."Book" (author, name, volume_number, compiled_by, original_language, translator, year_of_issue)
Library$#       VALUES (book_author, book_name, volume_number, compiled_by, original_language, translator, year_of_issue);
Library$# END;
Library$# $$;
CREATE PROCEDURE
```

```
Library=# CREATE OR REPLACE PROCEDURE new_user(name varchar(50), surname varchar(50), lastname varchar(50), address varchar(50), phone_number varchar(50), education varchar(75), series varchar(4), number varchar
(6), date_from date, date_to date, library_card_number int)
Library-# LANGUAGE plpgsql AS $$
Library$# DECLARE
Library$#       person_exists bool;
Library$# BEGIN
Library$#       SELECT NOT EXISTS(SELECT * FROM "Library"."Passport" AS P WHERE series = P.series AND number = P.number) INTO person_exists;
Library$#       IF person_exists THEN
Library$#               INSERT INTO "Library"."People" (name, surname, lastname, address, phone_number, role, education)
Library$#                       VALUES (name, surname, lastname, address, phone_number, "reader", education);
Library$#               INSERT INTO "Library"."Passport" (people_id, series, number, date_from, date_to)
Library$#                       VALUES ((SELECT people_id FROM "Library"."People" AS P WHERE P.name = name AND P.surname = surname AND P.lastname = lastname AND P.address = address AND P.phone_number = phone_num
ber AND P.education = education), series, number, date_from, date_to);
Library$#               INSERT INTO "Library"."Reader" (library_card_number, people_id)
Library$#                       VALUES ((SELECT MAX(library_card_number) FROM "Library"."Reader") + 1, (SELECT people_id FROM "Library"."People" AS P WHERE P.name = name AND P.surname = surname AND P.lastname =
lastname AND P.address = address AND P.phone_number = phone_number AND P.education = education));
Library$# END IF;
Library$# END;
Library$# $$;
CREATE PROCEDURE
```

Созданы 3(4) триггера:

```
Library=# CREATE OR REPLACE FUNCTION fn_del_inf_of_passport() RETURNS TRIGGER AS $psql$
Library$# BEGIN
Library$#       DELETE FROM "Library"."Passport" WHERE people_id = OLD.people_id;
Library$#       RETURN OLD;
Library$# END;
Library$# $psql$ LANGUAGE plpgsql;
CREATE FUNCTION
Library=# CREATE TRIGGER trig_del_inf_of_passport BEFORE DELETE ON "Library"."People"
Library-# FOR EACH ROW EXECUTE PROCEDURE fn_del_inf_of_passport();
CREATE TRIGGER
Library=# CREATE OR REPLACE FUNCTION fn_del_inf_of_reader() RETURNS TRIGGER AS $psql$
Library$# BEGIN
Library$#       DELETE FROM "Library"."Reader" WHERE people_id = OLD.people_id;
Library$#       RETURN OLD;
Library$# END;
Library$# $psql$ LANGUAGE plpgsql;
CREATE FUNCTION
Library=# CREATE TRIGGER trig_del_inf_of_reader BEFORE DELETE ON "Library"."People"
Library-# FOR EACH ROW EXECUTE PROCEDURE fn_del_inf_of_reader();
CREATE TRIGGER
```

```
Library=# CREATE OR REPLACE FUNCTION check_limit() RETURNS TRIGGER AS $psql$
Library$# DECLARE
Library$#       cur_bk_cnt INT;
Library$# BEGIN
Library$#       SELECT COUNT(*) INTO cur_bk_cnt FROM "Library"."Reading" WHERE library_card_number = NEW.library_card_number AND returned = false;
Library$#       IF cur_bk_cnt = 20 THEN
Library$#               RAISE EXCEPTION 'Лимит на выдачу книг достигнут, новые выдачи запрещены';
Library$#       END IF;
Library$#       RETURN NEW;
Library$# END;
Library$# $psql$ LANGUAGE plpgsql;
CREATE FUNCTION
Library=# CREATE TRIGGER new_reading BEFORE INSERT ON "Library"."Reading"
Library-# FOR EACH ROW EXECUTE PROCEDURE check_limit();
CREATE TRIGGER
```

```
Library=# CREATE TABLE new_users_log(id INT PRIMARY KEY, people_id INT, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
CREATE TABLE
Library=# CREATE OR REPLACE FUNCTION new_log() RETURNS TRIGGER AS $psql$
Library$# BEGIN
Library$#       IF NOT EXISTS(Select people_id from "Library"."new_users_log" AS nl WHERE nl.people_id = NEW.people_id) THEN
Library$#               INSERT INTO "Library"."new_users_log" (id, people_id)
Library$#                       VALUES (COALESCE((SELECT MAX(id) FROM "Library"."new_users_log"), 0) + 1, NEW.people_id);
Library$#       END IF;
Library$#
Library$#       RETURN NEW;
Library$# END;
Library$# $psql$ LANGUAGE plpgsql;
CREATE FUNCTION
Library=# CREATE TRIGGER INSERT_NEW_USER_LOG AFTER INSERT ON "Library"."People"
Library-#       FOR EACH ROW EXECUTE PROCEDURE new_log();
CREATE TRIGGER
Library=#
```

Исправлен триггер из практической работы:

```
emp_time=# create or replace function fn_check_time_punch() returns trigger as $psql$
emp_time$#     begin
emp_time$#         if new.is_out_punch = (
emp_time$#           select tps.is_out_punch
emp_time$#           from time_punch tps
emp_time$#           where tps.employee_id = new.employee_id
emp_time$#           order by tps.id desc limit 1
emp_time$#         ) and new.punch_time > (select tps.punch_time from time_punch tps where tps.employee_id = new.employee_id order by tps.id desc
emp_time$# limit 1)then
emp_time$#             return null;
emp_time$#         end if;
emp_time$#         return new;
emp_time$#     end;
emp_time$# $psql$ language plpgsql;
CREATE FUNCTION
emp_time=# create trigger check_time_punch before insert on time_punch
emp_time-# for each row execute procedure fn_check_time_punch();
ОШИБКА:  триггер "check_time_punch" для отношения "time_punch" уже существует
emp_time=# create or replace trigger check_time_punch before insert on time_punch
emp_time-# for each row execute procedure fn_check_time_punch();
CREATE TRIGGER
emp_time=#
```

Вывод: в ходе выполнения лабораторной работы 5 получилось овладеть практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL, с помощью консоли SQL Shell.