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 Машковцева Марина Алексеевна
Факультет прикладной информатики
Группа K3240
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ 3
2. ПРАКТИЧЕСКОЕ ЗАДАНИЕ 4
3. ВЫПОЛНЕНИЕ 5
3.1 Создание хранимых процедур и функций 5
3.2 Создание триггеров 7
4. ВЫВОДЫ 14

1. ЦЕЛЬ РАБОТЫ

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

2. ПРАКТИЧЕСКОЕ ЗАДАНИЕ

1. Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры.

2. Создать триггеры для индивидуальной БД согласно варианту: Вариант 2.2. 7 оригинальных триггеров.

3. ВЫПОЛНЕНИЕ

3.1 Создание хранимых процедур и функций

1) Добавить данные о новом штрафе водителя.

```
GIBDD=# CREATE OR REPLACE PROCEDURE "GIBDD".add_driver_violation(
GIBDD(#     IN p_violation_code INT,
GIBDD(#     IN p_datetime TIMESTAMP,
GIBDD(#     IN p_latitude DECIMAL,
GIBDD(#     IN p_payment_date DATE,
GIBDD(#     IN p_payment_status VARCHAR(13),
GIBDD(#     IN p_license_plate VARCHAR(9),
GIBDD(#     IN p_accident_id INT,
GIBDD(#     IN p_violation_id INT,
GIBDD(#     IN p_participant_id INT
GIBDD(# )
GIBDD-# LANGUAGE plpgsql
GIBDD-# AS $$
GIBDD$# BEGIN
GIBDD$#     INSERT INTO "GIBDD"."Violation"(
GIBDD$#         violation_code, datetime, latitude, payment_date, payment_status,
GIBDD$#         license_plate, accident_id, violation_id, participant_id
GIBDD$#     )
GIBDD$#     VALUES (
GIBDD$#         p_violation_code, p_datetime, p_latitude, p_payment_date, p_payment_status,
GIBDD$#         p_license_plate, p_accident_id, p_violation_id, p_participant_id
GIBDD$#     );
GIBDD$# END;
GIBDD$# $$;
CREATE PROCEDURE
```

Здесь и далее в процедурах и триггерах схема ГИБДД в кавычках, потому что иначе возникает ошибка schema "gibdd" does not exist.

Проверка:

```
GIBDD=# CALL "GIBDD".add_driver_violation(
GIBDD(# 55,
GIBDD(# '2025-05-20 12:00:00',
GIBDD(# 55.755800,
GIBDD(# NULL,
GIBDD(# 'не оплачено',
GIBDD(# 'Е222ОП34',
GIBDD(# 1,
GIBDD(# 1,
GIBDD(# 1
GIBDD(# );
CALL
```

```
GIBDD=# SELECT * FROM "GIBDD"."Violation"
GIBDD-# WHERE license_plate = 'Е222ОП34';
 violation_code |      datetime       | latitude  | payment_date | payment_status | license_plate | violation_id | accident_id | participant_id
----------------+---------------------+-----------+--------------+----------------+---------------+--------------+-------------+----------------
             30 | 2023-04-15 13:30:00 | 55.796100 |              | не оплачено    | Е222ОП34      |            5 |             |
             55 | 2025-05-20 12:00:00 | 55.755800 |              | не оплачено    | Е222ОП34      |            1 |           1 |              1
(2 rows)
```

2) Вывести данные инспектора, оштрафовавшего одного и того же водителя более одного раза. (Функция)

```
GIBDD=# CREATE OR REPLACE FUNCTION "GIBDD".get_inspectors_fined_driver_more_than_once(
GIBDD(#     p_driver_id INT
GIBDD(# )
GIBDD-# RETURNS TABLE (
GIBDD(#     inspector_id INT,
GIBDD(#     full_name TEXT,
GIBDD(#     num_violations BIGINT
GIBDD(# )
GIBDD-# LANGUAGE plpgsql
GIBDD-# AS $$
GIBDD$# BEGIN
GIBDD$#     RETURN QUERY
GIBDD$#     SELECT i.inspector_id, i.full_name, COUNT(*)::BIGINT
GIBDD$#     FROM "GIBDD"."Violation" v
GIBDD$#     JOIN "GIBDD"."Accident participant" ap ON v.participant_id = ap.participant_id
GIBDD$#     JOIN "GIBDD"."Accident" a ON v.accident_id = a.accident_id
GIBDD$#     JOIN "GIBDD"."Inspector" i ON a.inspector_id = i.inspector_id
GIBDD$#     WHERE ap.driver_id = p_driver_id
GIBDD$#     GROUP BY i.inspector_id, i.full_name
GIBDD$#     HAVING COUNT(*) > 1;
GIBDD$# END;
GIBDD$# $$;
CREATE FUNCTION
```

Проверка:

```
GIBDD=# SELECT * FROM "GIBDD".get_inspectors_fined_driver_more_than_once(1);
 inspector_id | full_name | violation_count
--------------+-----------+-----------------
(0 rows)
```

Проверка вручную для точности результата:

```
GIBDD=# SELECT a.inspector_id, COUNT(*)
GIBDD-# FROM "GIBDD"."Violation" v
GIBDD-# JOIN "GIBDD"."Accident participant" ap ON v.participant_id = ap.participant_id
GIBDD-# JOIN "GIBDD"."Accident" a ON v.accident_id = a.accident_id
GIBDD-# WHERE ap.driver_id = 1
GIBDD-# GROUP BY a.inspector_id
GIBDD-# HAVING COUNT(*) > 1;
 inspector_id | count
--------------+-------
(0 rows)
```

3) Вывести количество нарушений, повлекших лишение прав в заданном, как параметр районе. (Функция)

```
GIBDD=# CREATE OR REPLACE FUNCTION "GIBDD".get_license_suspensions_count(
GIBDD(#     p_department INT
GIBDD(# )
GIBDD-# RETURNS BIGINT
GIBDD-# LANGUAGE plpgsql
GIBDD-# AS $$
GIBDD$# BEGIN
GIBDD$#     RETURN (
GIBDD$#         SELECT COUNT(*)
GIBDD$#         FROM "GIBDD"."Violation" v
GIBDD$#         JOIN "GIBDD"."Violation type" vt ON v.violation_code = vt.violation_id
GIBDD$#         JOIN "GIBDD"."Accident" a ON v.accident_id = a.accident_id
GIBDD$#         JOIN "GIBDD"."Inspector" i ON a.inspector_id = i.inspector_id
GIBDD$#         WHERE i.department_number = p_department
GIBDD$#           AND vt.license_suspension_period <> ''
GIBDD$#     );
GIBDD$# END;
GIBDD$# $$;
CREATE FUNCTION
```

Проверка:

```
GIBDD=# SELECT "GIBDD".get_license_suspensions_count(1);
 get_license_suspensions_count
-------------------------------
                             2
(1 row)
```

Проверка вручную:

```
GIBDD=# SELECT COUNT(*) AS manual_count
GIBDD-# FROM "GIBDD"."Violation" v
GIBDD-# JOIN "GIBDD"."Violation type" vt ON v.violation_code = vt.violation_id
GIBDD-# JOIN "GIBDD"."Accident" a ON v.accident_id = a.accident_id
GIBDD-# JOIN "GIBDD"."Inspector" i ON a.inspector_id = i.inspector_id
GIBDD-# WHERE i.department_number = 1
GIBDD-#   AND vt.license_suspension_period <> '';
 manual_count
--------------
            2
(1 row)
```

3.2 Создание триггеров

1) Логирование вставки в таблицу Violation

```
GIBDD=# CREATE TABLE "GIBDD".violation_log (
GIBDD(# log_id SERIAL PRIMARY KEY,
GIBDD(#     violation_code INT,
GIBDD(#     insert_time TIMESTAMP DEFAULT now()
GIBDD(# );
CREATE TABLE
GIBDD=# CREATE OR REPLACE FUNCTION "GIBDD".log_violation_insert()
GIBDD-# RETURNS TRIGGER AS $$
GIBDD$# BEGIN
GIBDD$#     INSERT INTO "GIBDD".violation_log (violation_code)
GIBDD$#     VALUES (NEW.violation_code);
GIBDD$#     RETURN NEW;
GIBDD$# END;
GIBDD$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
GIBDD=# CREATE TRIGGER trg_log_violation_insert
GIBDD-# AFTER INSERT ON "GIBDD"."Violation"
GIBDD-# FOR EACH ROW
GIBDD-# EXECUTE FUNCTION "GIBDD".log_violation_insert();
CREATE TRIGGER
```

Проверка:

```
GIBDD=# INSERT INTO "GIBDD"."Violation" (
GIBDD(#     violation_code, datetime, latitude,
GIBDD(#     payment_date, payment_status, license_plate,
GIBDD(#     accident_id, violation_id, participant_id
GIBDD(# )
GIBDD-# VALUES (
GIBDD(#     56,
GIBDD(#     '2025-05-21 10:00:00',
GIBDD(#     55.755800,
GIBDD(#     NULL,
GIBDD(#     'не оплачено',
GIBDD(#     'Е222ОП34',
GIBDD(#     1,
GIBDD(#     1,
GIBDD(#     1
GIBDD(# );
INSERT 0 1
```

```
GIBDD=# SELECT * FROM "GIBDD".violation_log;
 log_id | violation_code |        insert_time
--------+----------------+----------------------------
      1 |             56 | 2025-05-21 12:05:20.897179
(1 row)
```

2) Проверка, чтобы штрафы не создавались без номера машины

```
GIBDD=# CREATE OR REPLACE FUNCTION "GIBDD".check_license_plate_not_null()
GIBDD-# RETURNS TRIGGER AS $$
GIBDD$# BEGIN
GIBDD$#     IF NEW.license_plate IS NULL THEN
GIBDD$#         RAISE EXCEPTION 'Номер машины не может быть пустым!';
GIBDD$#     END IF;
GIBDD$#     RETURN NEW;
GIBDD$# END;
GIBDD$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
GIBDD=#
GIBDD=# CREATE TRIGGER trg_check_license_plate
GIBDD-# BEFORE INSERT OR UPDATE ON "GIBDD"."Violation"
GIBDD-# FOR EACH ROW
GIBDD-# EXECUTE FUNCTION "GIBDD".check_license_plate_not_null();
CREATE TRIGGER
```

Проверка:

```
GIBDD=# INSERT INTO "GIBDD"."Violation" (violation_code, license_plate)
GIBDD-# VALUES (101, NULL);
ERROR:  Номер машины не может быть пустым!
```

3) Обновление даты последнего штрафа в таблице Driver

```
GIBDD=# CREATE OR REPLACE FUNCTION "GIBDD".update_last_violation_date()
GIBDD-# RETURNS TRIGGER AS $$
GIBDD$# BEGIN
GIBDD$#     UPDATE "GIBDD"."Driver"
GIBDD$# SET last_violation_date = NEW.payment_date
GIBDD$#     WHERE license_number = (
GIBDD$#         SELECT license_number FROM "GIBDD"."Car" WHERE car_number = NEW.license_plate
GIBDD$# );
GIBDD$#     RETURN NEW;
GIBDD$# END;
GIBDD$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
GIBDD=# CREATE TRIGGER trg_update_last_violation_date
GIBDD-# AFTER INSERT ON "GIBDD"."Violation"
GIBDD-# FOR EACH ROW
GIBDD-# EXECUTE FUNCTION "GIBDD".update_last_violation_date();
CREATE TRIGGER
```

Проверка:

```
GIBDD=# INSERT INTO "GIBDD"."Driver" (
GIBDD(#     phone_number,
GIBDD(#     license_number,
GIBDD(#     address,
GIBDD(#     full_name,
GIBDD(#     last_violation_date
GIBDD(# ) VALUES (
GIBDD(# '79990001122',
GIBDD(# 123456,
GIBDD(# 'г. Москва, ул. Тестовая, 1',
GIBDD(#     'Тестовый Водитель',
GIBDD(#     NULL
GIBDD(# );
INSERT 0 1
```

```
GIBDD=# INSERT INTO "GIBDD"."Car" (
GIBDD(#     car_number,
GIBDD(#     insurance_type,
GIBDD(#     color,
GIBDD(#     registration_date,
GIBDD(#     manufacture_year,
GIBDD(#     model,
GIBDD(#     license_number
GIBDD(# ) VALUES (
GIBDD(#     'X999XX999',
GIBDD(#     'ocaro',
GIBDD(#     'черный',
GIBDD(#     '2015-03-05',
GIBDD(#     2020,
GIBDD(#     'Solaris',
GIBDD(#     123456
GIBDD(# );
INSERT 0 1
GIBDD=# SELECT * FROM "GIBDD"."Car" WHERE license_number = 123456;
 car_number | insurance_type | color  | registration_date | manufacture_year |  model  | license_number
------------+----------------+--------+-------------------+------------------+---------+----------------
 X999XX999  | ocaro          | черный | 2015-03-05        |             2020 | Solaris |         123456
(1 row)
```

```
GIBDD=# SELECT * FROM "GIBDD"."Driver" WHERE license_number = 123456;
 phone_number | license_number |          address           |     full_name
| last_violation_date
--------------+----------------+----------------------------+------------------
+---------------------
 79990001122  |         123456 | г. Москва, ул. Тестовая, 1 | Тестовый Водитель
|
(1 row)
```

4) Запрет удаления инспектора, если он участвует в авариях

```
GIBDD=# CREATE OR REPLACE FUNCTION "GIBDD".prevent_inspector_delete()
GIBDD-# RETURNS TRIGGER AS $$
GIBDD$# BEGIN
GIBDD$#     IF EXISTS (
GIBDD$#         SELECT 1 FROM "GIBDD".Accident WHERE inspector_id = OLD.inspector_id
GIBDD$#     ) THEN
GIBDD$#         RAISE EXCEPTION 'Нельзя удалить инспектора, участвующего в авариях.';
GIBDD$#     END IF;
GIBDD$#     RETURN OLD;
GIBDD$# END;
GIBDD$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
GIBDD=# CREATE TRIGGER trg_prevent_inspector_delete
GIBDD-# BEFORE DELETE ON "GIBDD"."Inspector"
GIBDD-# FOR EACH ROW
GIBDD-# EXECUTE FUNCTION "GIBDD".prevent_inspector_delete();
CREATE TRIGGER
```

Проверка:

Берем существующего инспектора из таблицы.

```
GIBDD=# SELECT * FROM "GIBDD"."Inspector";
 inspector_id |           full_name           | department_number | phone_number
--------------+-------------------------------+-------------------+--------------
         2001 | Иванов Алексей Владимирович   |                 1 | 79261001122
         2002 | Петрова Елена Сергеевна       |                 1 | 79262112233
GIBDD=# DELETE FROM "GIBDD"."Inspector"
GIBDD-# WHERE inspector_id = 2002;
ERROR:  Нельзя удалить инспектора, участвующего в авариях.
```

5) Логирование изменения статуса аварии

```
GIBDD=# CREATE TABLE "GIBDD".accident_status_log (
GIBDD(#     log_id SERIAL PRIMARY KEY,
GIBDD(#     accident_id INT,
GIBDD(#     old_status VARCHAR,
GIBDD(#     new_status VARCHAR,
GIBDD(#     change_time TIMESTAMP DEFAULT now()
GIBDD(# );
CREATE TABLE
```

```
GIBDD=# CREATE OR REPLACE FUNCTION "GIBDD".log_accident_status_change()
GIBDD-# RETURNS TRIGGER AS $$
GIBDD$# BEGIN
GIBDD$#     IF OLD.status IS DISTINCT FROM NEW.status THEN
GIBDD$#         INSERT INTO "GIBDD".accident_status_log(accident_id, old_status, new_status)

GIBDD$#         VALUES (OLD.accident_id, OLD.status, NEW.status);
GIBDD$#     END IF;
GIBDD$#     RETURN NEW;
GIBDD$# END;
GIBDD$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
GIBDD=# CREATE TRIGGER trg_log_accident_status_change
GIBDD-# AFTER UPDATE ON "GIBDD"."Accident"
GIBDD-# FOR EACH ROW  -- Ключевое изменение!
GIBDD-# EXECUTE FUNCTION "GIBDD".log_accident_status_change();
CREATE TRIGGER
```

Проверка:

```
GIBDD=# UPDATE "GIBDD"."Accident"
GIBDD-# SET status = 'завершено'
GIBDD-# WHERE accident_id = 1;
UPDATE 1
GIBDD=# SELECT * FROM "GIBDD".accident_status_log;
 log_id | accident_id |    old_status    | new_status |        change_time
--------+-------------+------------------+------------+----------------------------
      1 |           1 | зарегистрировано | завершено  | 2025-05-21 15:05:32.788053
(1 row)
```

6) Автоматическое заполнение даты регистрации машины, если она не указана изначально

```
GIBDD=# CREATE OR REPLACE FUNCTION "GIBDD".set_default_registration_date()
GIBDD-# RETURNS TRIGGER AS $$
GIBDD$# BEGIN
GIBDD$#     IF NEW.registration_date IS NULL THEN
GIBDD$#         NEW.registration_date := CURRENT_DATE;
GIBDD$#     END IF;
GIBDD$#     RETURN NEW;
GIBDD$# END;
GIBDD$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
GIBDD=# CREATE TRIGGER trg_set_registration_date
GIBDD-# BEFORE INSERT ON "GIBDD"."Car"
GIBDD-# FOR EACH ROW
GIBDD-# EXECUTE FUNCTION "GIBDD".set_default_registration_date();
CREATE TRIGGER
```

Проверка:

```
GIBDD=# INSERT INTO "GIBDD"."Car" (
GIBDD(#     car_number,
GIBDD(#     insurance_type,
GIBDD(#     color,
GIBDD(#     manufacture_year,
GIBDD(#     model,
GIBDD(#     license_number
GIBDD(# ) VALUES (
GIBDD(#     'A111AA777',
GIBDD(#     'осаго',
GIBDD(#     'Красный',
GIBDD(#     2020,
GIBDD(#     'Vesta',
GIBDD(#     123456
GIBDD(# );
INSERT 0 1
GIBDD=# SELECT car_number, registration_date
GIBDD-# FROM "GIBDD"."Car"
GIBDD-# WHERE car_number = 'A111AA777';
 car_number | registration_date
------------+-------------------
 A111AA777  | 2025-05-21
(1 row)


GIBDD=# INSERT INTO "GIBDD"."Car" (
GIBDD(#     car_number,
GIBDD(#     insurance_type,
GIBDD(#     color,
GIBDD(#     registration_date,
GIBDD(#     manufacture_year,
GIBDD(#     model,
GIBDD(#     license_number
GIBDD(# ) VALUES (
GIBDD(#     'B222BB777',
GIBDD(#     'каско',
GIBDD(#     'Синий',
GIBDD(#     '2020-01-15',
GIBDD(#     2019,
GIBDD(#     'Vesta',
GIBDD(# 123456
GIBDD(# );
INSERT 0 1
GIBDD=# SELECT car_number, registration_date
GIBDD-# FROM "GIBDD"."Car"
GIBDD-# WHERE car_number = 'B222BB777';
 car_number | registration_date
------------+-------------------
 B222BB777  | 2020-01-15
(1 row)
```

7) Запрет добавления водителя с существующим номером прав

```
GIBDD=# CREATE OR REPLACE FUNCTION "GIBDD".unique_license_check()
GIBDD-# RETURNS TRIGGER AS $$
GIBDD$# BEGIN
GIBDD$#     IF EXISTS (SELECT 1 FROM "GIBDD"."Driver" WHERE license_number = NEW.license_number) THEN
GIBDD$#         RAISE EXCEPTION 'Водитель с таким номером прав уже существует!';
GIBDD$#     END IF;
GIBDD$#     RETURN NEW;
GIBDD$# END;
GIBDD$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
GIBDD=# CREATE TRIGGER trg_unique_license
GIBDD-# BEFORE INSERT ON "GIBDD"."Driver"
GIBDD-# FOR EACH ROW
GIBDD-# EXECUTE FUNCTION "GIBDD".unique_license_check();
CREATE TRIGGER
```

Проверка:

```
GIBDD=# INSERT INTO "GIBDD"."Driver" (
GIBDD(#     phone_number,
GIBDD(#     license_number,
GIBDD(#     address,
GIBDD(#     full_name
GIBDD(# ) VALUES (
GIBDD(#     '79110001122',
GIBDD(#     999999,
GIBDD(#     'г. Москва, ул. Тестовая, 1',
GIBDD(#     'Абв Абв Абв'
GIBDD(# );
INSERT 0 1
GIBDD=# INSERT INTO "GIBDD"."Driver" (
GIBDD(#     phone_number,
GIBDD(#     license_number,
GIBDD(#     address,
GIBDD(#     full_name
GIBDD(# ) VALUES (
GIBDD(#     '79120002233',
GIBDD(#     999999,
GIBDD(#     'г. Москва, ул. Тестовая, 2',
GIBDD(#     'Ваб Ваб Ваб'
GIBDD(# );
ERROR:  Водитель с таким номером прав уже существует!
```

4. ВЫВОДЫ

В ходе выполнения лабораторной работы были освоены практические навыки создания и использования хранимых процедур, функций и триггеров в PostgreSQL. Были успешно реализованы три процедуры и семь триггеров для индивидуальной базы данных, что позволило автоматизировать выполнение типовых операций, обеспечить целостность данных и логирование изменений.

Работа показала, что использование хранимых процедур и триггеров значительно упрощает управление базой данных, повышает её надежность и сокращает количество ручных операций. Полученные навыки могут быть применены в дальнейшем для проектирования и оптимизации более сложных систем управления базами данных.