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»

Обучающиеся Захматов Юрий Дмитриевич

Факультет прикладной информатики

Группа K3241

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, pgadmin 4, SQL Shell (psql).

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.
   Вариант 2.2. 7 оригинальных триггеров - 7 баллов (max)

.**Вариант 18. БД «ГИБДД»**

Создать хранимые процедуры:

1) Добавить данные о новом штрафе водителя.
2) Вывести данные инспектора, оштрафовавшего одного и того же водителя более одного раза.
3) Вывести количество нарушений, повлекших лишение прав в заданном, как параметр районе.

Создать триггеры.

Ход работы:

Создадим хранимые процедуры:

1. Добавить данные о новом штрафе водителя
   Код:

```
CREATE OR REPLACE PROCEDURE "GIBDD"."add_driver_fine"(
    p_driver_id INTEGER,
    p_inspector_id INTEGER,
    p_vehicle_id INTEGER,
    p_violation_code INTEGER,
    p_fine_amount INTEGER,
    p_license_suspension INTEGER DEFAULT NULL,
    p_latitude NUMERIC(9,6),
    p_longitude NUMERIC(9,6),
    p_violation_date TIMESTAMP DEFAULT NOW(),
    p_resolution_number VARCHAR(50)
)
LANGUAGE plpgsql
AS $$
DECLARE
    v_fine_id INTEGER;
BEGIN
```

```
    -- Проверка существования водителя, инспектора, ТС и нарушения
    IF NOT EXISTS (SELECT 1 FROM "GIBDD"."Водитель" WHERE "id_водителя" =
p_driver_id) THEN
        RAISE EXCEPTION 'Водитель с ID % не найден', p_driver_id;
    END IF;

    IF NOT EXISTS (SELECT 1 FROM "GIBDD"."Инспектор" WHERE "id_инспектора"
= p_inspector_id) THEN
        RAISE EXCEPTION 'Инспектор с ID % не найден', p_inspector_id;
    END IF;

    IF NOT EXISTS (SELECT 1 FROM "GIBDD"."Транспортное_средство" WHERE
"id_ТС" = p_vehicle_id) THEN
        RAISE EXCEPTION 'Транспортное средство с ID % не найдено',
p_vehicle_id;
    END IF;

    IF NOT EXISTS (SELECT 1 FROM "GIBDD"."Нарушения_ПДД" WHERE
"Код_нарушения" = p_violation_code) THEN
        RAISE EXCEPTION 'Нарушение с кодом % не найдено', p_violation_code;
    END IF;

    -- Создание штрафа
    INSERT INTO "GIBDD"."Штраф" (
        "Сумма_штрафа",
        "Размер_оплаты",
        "Дата_и_время_оплаты",
        "Статус_оплаты",
        "Номер_постановления_о_штрафе"
    ) VALUES (
        p_fine_amount,
        0,
        NULL,
        'Не оплачен',
        p_resolution_number
    ) RETURNING "id_штрафа" INTO v_fine_id;

    -- Регистрация нарушения
    INSERT INTO "GIBDD"."Регистрация_нарушения" (
        "id_водителя",
        "id_инспектора",
        "id_штрафа",
        "id_ТС",
        "Код_нарушения",
        "Срок_лишения_прав_управления_ТС",
        "Координаты_нарушения_широта",
        "Координаты_нарушения_долгота",
        "Дата_и_время_нарушения"
    ) VALUES (
        p_driver_id,
        p_inspector_id,
        v_fine_id,
        p_vehicle_id,
        p_violation_code,
        p_license_suspension,
        p_latitude,
        p_longitude,
        p_violation_date
    );

    COMMIT;
```

```
    RAISE NOTICE 'Штраф успешно добавлен. ID штрафа: %', v_fine_id;
END;
$$;
```

Пример работы:

Query | Query History

```
CALL "GIBDD"."add_driver_fine"(
    p_driver_id := 1,
    p_inspector_id := 3,
    p_vehicle_id := 1,
    p_violation_code := 5,
    p_fine_amount := 30000,
    p_license_suspension := 180,
    p_latitude := 55.755825,
    p_longitude := 37.617298,
    p_violation_date := '2025-05-25 14:30:00',
    p_resolution_number := '78-20250525-1'
);
```

Data Output | Messages | Notifications

```
ЗАМЕЧАНИЕ:  Штраф успешно добавлен. ID штрафа: 57
CALL

Query returned successfully in 92 msec.
```

Query | Query History

```
SELECT * FROM "GIBDD"."Штраф"
WHERE id_штрафа = 57;
```

Data Output | Messages | Notifications

	id_штрафа [PK] integer	Сумма_штрафа integer	Размер_оплаты integer	Дата_и_время_оплаты timestamp without time zone	Статус_оплаты "GIBDD"."размер_оплаты"	Номер_постановления_о_штрафе character varying (50)
1	57	30000	0	[null]	Не оплачен	78-20250525-1

Процедура работает.

2. Вывести данные инспектора, оштрафовавшего одного и того же водителя более одного раза.
   Код:

```
CREATE OR REPLACE PROCEDURE "GIBDD"."get_repeat_inspectors_for_driver"(
    p_driver_id INTEGER
)
LANGUAGE plpgsql
AS $$
DECLARE
    rec RECORD;
BEGIN
    -- Проверка существования водителя
    IF NOT EXISTS (SELECT 1 FROM "GIBDD"."Водитель" WHERE "id_водителя" =
p_driver_id) THEN
        RAISE EXCEPTION 'Водитель с ID % не найден', p_driver_id;
    END IF;

    -- Вывод результатов
    RAISE NOTICE 'Инспекторы, которые оштрафовали водителя ID % более
одного раза:', p_driver_id;

    -- Создаем временную таблицу для результатов
    CREATE TEMP TABLE IF NOT EXISTS temp_results (
        inspector_id INTEGER,
        inspector_name VARCHAR(255),
        rank VARCHAR(100),
        violations_count INTEGER
    );

    -- Заполняем таблицу данными
    INSERT INTO temp_results
    SELECT
        i."id_инспектора" AS inspector_id,
        i."ФИО" AS inspector_name,
        i."Звание" AS rank,
        COUNT(r."id_нарушения") AS violations_count
    FROM
        "GIBDD"."Регистрация_нарушения" r
        JOIN "GIBDD"."Инспектор" i ON r."id_инспектора" = i."id_инспектора"
    WHERE
        r."id_водителя" = p_driver_id
    GROUP BY
        i."id_инспектора", i."ФИО", i."Звание"
    HAVING
        COUNT(r."id_нарушения") > 1
    ORDER BY
        violations_count DESC;

    -- Вывод заголовка таблицы
    RAISE NOTICE '| % | % | % | % |',
        lpad('ID инспектора', 15),
        lpad('ФИО инспектора', 30),
        lpad('Звание', 20),
        lpad('Количество нарушений', 20);
    RAISE NOTICE '|%|%|%|%|',
        repeat('-', 17),
        repeat('-', 32),
        repeat('-', 22),
        repeat('-', 22);
```

```
    -- Вывод данных
    FOR rec IN SELECT * FROM temp_results LOOP
        RAISE NOTICE '| % | % | % | % |',
            lpad(rec.inspector_id::TEXT, 15),
            lpad(rec.inspector_name, 30),
            lpad(rec.rank, 20),
            lpad(rec.violations_count::TEXT, 20);
    END LOOP;

    -- Удаляем временную таблицу
    DROP TABLE IF EXISTS temp_results;
END;
$$;
```

Пример работы:

```
1   CALL "GIBDD"."get_repeat_inspectors_for_driver"(1);
```

Data Output | Messages | Notifications

```
ЗАМЕЧАНИЕ:  Инспекторы, которые оштрафовали водителя ID 1 более одного раза:
ЗАМЕЧАНИЕ:  |   ID инспектора |                ФИО инспектора |               Звание | Количество нарушений |
ЗАМЕЧАНИЕ:  |-----------------|-------------------------------|----------------------|----------------------|
ЗАМЕЧАНИЕ:  |               3 |       Лебедев Михаил Сергеевич | Старший лейтенант по |                    8 |
ЗАМЕЧАНИЕ:  |               1 |   Соколов Александр Васильевич |      Капитан полиции |                    2 |
CALL

Query returned successfully in 39 msec.
```

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

Код:

```
CREATE OR REPLACE PROCEDURE "GIBDD"."get_license_suspensions_in_area"(
    p_min_lat NUMERIC(9,6),
    p_max_lat NUMERIC(9,6),
    p_min_lon NUMERIC(9,6),
    p_max_lon NUMERIC(9,6))
LANGUAGE plpgsql
AS $$
DECLARE
    v_count INTEGER;
BEGIN
    -- Проверка корректности координат
    IF p_min_lat >= p_max_lat OR p_min_lon >= p_max_lon THEN
        RAISE EXCEPTION 'Некорректные координаты района: минимальные
значения должны быть меньше максимальных';
    END IF;

    -- Подсчет нарушений
    SELECT COUNT(*) INTO v_count
    FROM "GIBDD"."Регистрация_нарушения"
    WHERE
        "Координаты_нарушения_широта" BETWEEN p_min_lat AND p_max_lat AND
        "Координаты_нарушения_долгота" BETWEEN p_min_lon AND p_max_lon AND
        "Срок_лишения_прав_управления_ТС" > 0;

    -- Вывод результатов
    RAISE NOTICE 'В районе с координатами (широта: %-%, долгота: %-%)',
        p_min_lat, p_max_lat, p_min_lon, p_max_lon;
    RAISE NOTICE 'было зафиксировано % нарушений, повлекших лишение прав',
v_count;
```

```
END;
$$;
```

Пример работы:

```
31
32  CALL "GIBDD"."get_license_suspensions_in_area"(
33      p_min_lat := 40.74,
34      p_max_lat := 60.77,
35      p_min_lon := 30.60,
36      p_max_lon := 40.63
37  );
```

Data Output | Messages | Notifications

```
ЗАМЕЧАНИЕ:  В районе с координатами (широта: 40.74-60.77, долгота: 30.60-40.63)
ЗАМЕЧАНИЕ:  было зафиксировано 6 нарушений, повлекших лишение прав
CALL

Query returned successfully in 62 msec.
```

Создание триггеров:

1. Триггер для проверки даты регистрации ТС (не раньше даты выпуска)

Код:

```
      CREATE OR REPLACE FUNCTION
"GIBDD"."check_vehicle_registration_date"()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW."Дата_регистрации_в_ГИБДД" IS NOT NULL AND
       NEW."Дата_регистрации_в_ГИБДД" < NEW."Дата_выпуска" THEN
        RAISE EXCEPTION 'Дата регистрации не может быть раньше даты выпуска
ТС';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "trg_check_vehicle_registration"
BEFORE INSERT OR UPDATE ON "GIBDD"."Транспортное_средство"
FOR EACH ROW EXECUTE FUNCTION "GIBDD"."check_vehicle_registration_date"();
```

Пример работы:

Query | Query History

```
1 INSERT INTO "GIBDD"."Транспортное_средство" (
2     "Дата_регистрации_в_ГИБДД", "Дата_выпуска", "Номер_ТС", "Категория_ТС"
3 ) VALUES (
4     '2020-01-01', '2021-01-01', 'TEST001', 'B'
5 );
```

Data Output | Messages | Notifications

```
ERROR:  Дата регистрации не может быть раньше даты выпуска ТС
CONTEXT:  функция PL/pgSQL "GIBDD".check_vehicle_registration_date(), строка 5, оператор RAISE

ОШИБКА:  Дата регистрации не может быть раньше даты выпуска ТС
SQL state: P0001
```

2. Триггер обновления статуса штрафа

Код:

```
CREATE OR REPLACE FUNCTION "GIBDD"."update_fine_status_on_payment"()
RETURNS TRIGGER AS $$
BEGIN
    -- Если произошла оплата (дата оплаты изменилась с NULL на конкретную
дату)
    IF NEW."Дата_и_время_оплаты" IS NOT NULL AND OLD."Дата_и_время_оплаты"
IS NULL THEN

        -- Обновляем статус в зависимости от суммы оплаты
        IF NEW."Размер_оплаты" >= NEW."Сумма_штрафа" THEN
            NEW."Статус_оплаты" = 'Полный';
        ELSIF NEW."Размер_оплаты" > 0 THEN
            NEW."Статус_оплаты" = 'Частичный';
        ELSE
            NEW."Статус_оплаты" = 'Не оплачен';
        END IF;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "trg_update_fine_status"
BEFORE UPDATE ON "GIBDD"."Штраф"
FOR EACH ROW EXECUTE FUNCTION "GIBDD"."update_fine_status_on_payment"();
```

Пример работы:

3. Триггер на проверку лишения прав при регистрации нарушения

   Код:

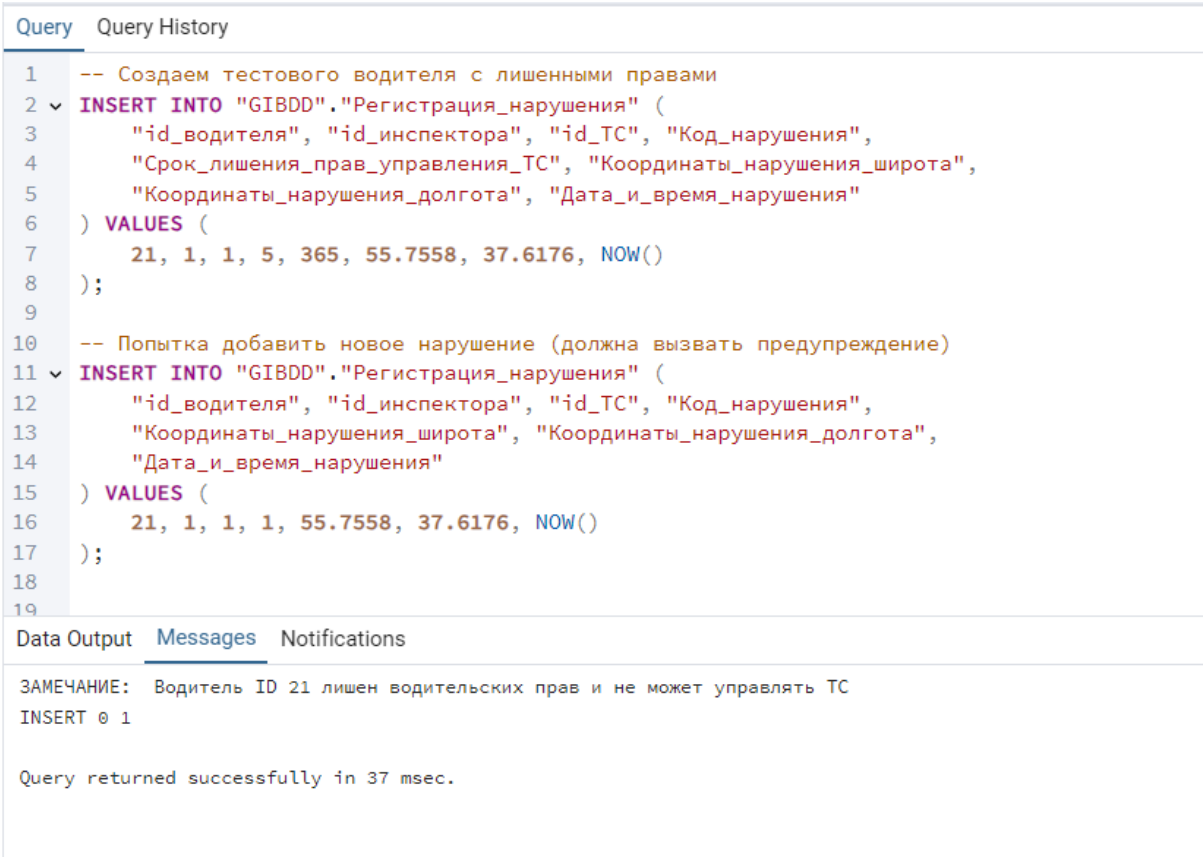
```
CREATE OR REPLACE FUNCTION "GIBDD"."check_driver_license_validity"()
RETURNS TRIGGER AS $$
DECLARE
    v_license_suspended BOOLEAN;
BEGIN
    -- Проверяем, не лишен ли водитель прав
    SELECT EXISTS (
        SELECT 1 FROM "GIBDD"."Регистрация_нарушения"
        WHERE "id_водителя" = NEW."id_водителя"
        AND "Срок_лишения_прав_управления_ТС" > 0
        AND "Дата_и_время_нарушения" + ("Срок_лишения_прав_управления_ТС" *
INTERVAL '1 day') > NOW()
    ) INTO v_license_suspended;

    IF v_license_suspended THEN
        RAISE NOTICE 'Водитель ID % лишен водительских прав и не может
управлять ТС', NEW."id_водителя";
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "trg_check_driver_license_validity"
BEFORE INSERT ON "GIBDD"."Регистрация_нарушения"
FOR EACH ROW EXECUTE FUNCTION "GIBDD"."check_driver_license_validity"();
```

   Пример работы:



4. Триггер для проверки уникальности VIN
   Код:

```
CREATE OR REPLACE FUNCTION "GIBDD"."check_vin_uniqueness"()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW."vin_код_ТС" IS NOT NULL AND EXISTS (
        SELECT 1 FROM "GIBDD"."Транспортное_средство"
        WHERE "vin_код_ТС" = NEW."vin_код_ТС"
        AND "id_ТС" != COALESCE(NEW."id_ТС", 0)
    THEN
        RAISE EXCEPTION 'Транспортное средство с таким VIN-кодом уже
зарегистрировано';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "trg_check_vin_uniqueness"
BEFORE INSERT OR UPDATE ON "GIBDD"."Транспортное_средство"
FOR EACH ROW EXECUTE FUNCTION "GIBDD"."check_vin_uniqueness"();
```

5. Триггер для логгирования изменений в таблице ДТП
   Код:

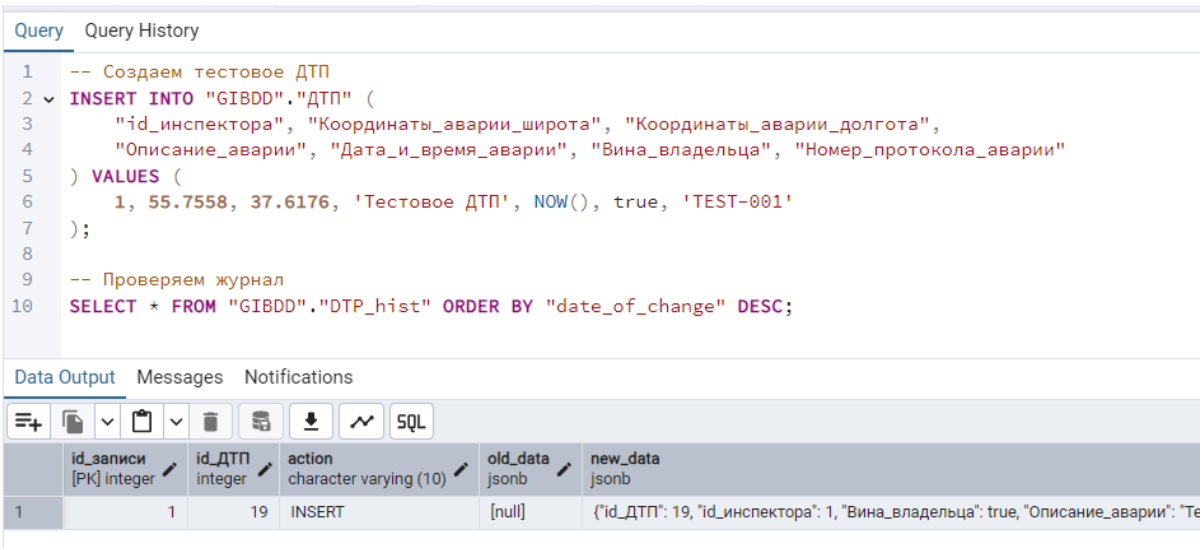
```
CREATE TABLE IF NOT EXISTS "GIBDD"."DTP_hist" (
    "id_записи" SERIAL PRIMARY KEY,
    "id_ДТП" INTEGER,
    "action" VARCHAR(10),
    "old_data" JSONB,
    "new_data" JSONB,
    "date_of_change" TIMESTAMP DEFAULT NOW(),
```

```
    "user" VARCHAR(100) DEFAULT CURRENT_USER
);

CREATE OR REPLACE FUNCTION "GIBDD"."log_dtp_changes"()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO "GIBDD"."DTP_hist" (
            "id_ДТП", "action", "new_data"
        ) VALUES (
            NEW."id_ДТП", 'INSERT', to_jsonb(NEW)
        );
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO "GIBDD"."DTP_hist" (
            "id_ДТП", "action", "old_data", "new_data"
        ) VALUES (
            NEW."id_ДТП", 'UPDATE', to_jsonb(OLD), to_jsonb(NEW)
        );
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO "GIBDD"."DTP_hist" (
            "id_ДТП", "action", "old_data"
        ) VALUES (
            OLD."id_ДТП", 'DELETE', to_jsonb(OLD)
        );
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "trg_log_dtp_changes"
AFTER INSERT OR UPDATE OR DELETE ON "GIBDD"."ДТП"
FOR EACH ROW EXECUTE FUNCTION "GIBDD"."log_dtp_changes"();
```

Пример работы:



6. Триггер для логгирования изменений в таблице Водитель

Код:

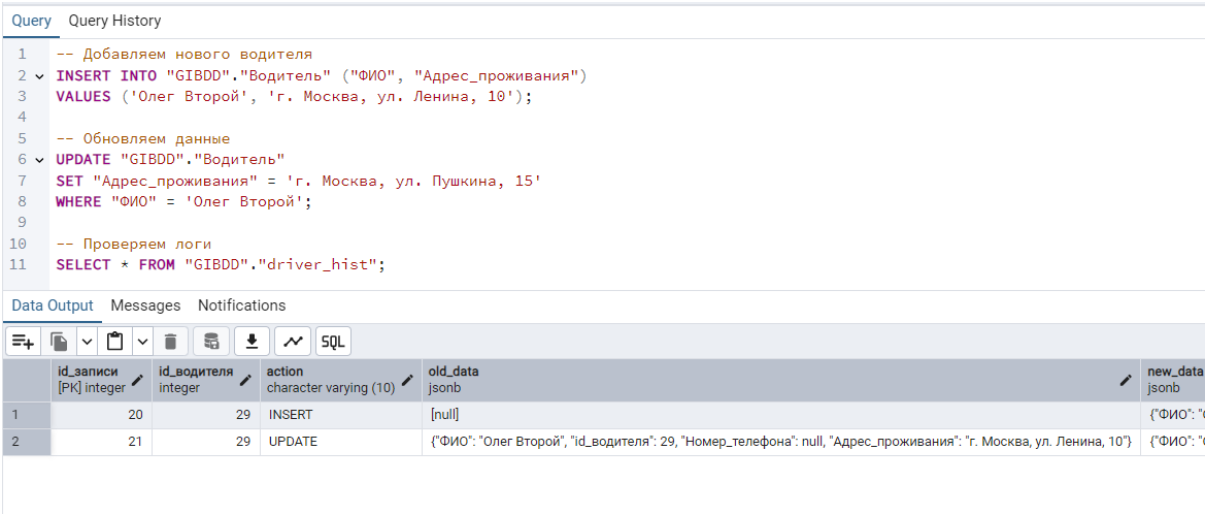
```
-- Создаем таблицу для логов водителей
CREATE TABLE IF NOT EXISTS "GIBDD"."driver_hist" (
    "id_записи" SERIAL PRIMARY KEY,
    "id_водителя" INTEGER NOT NULL,
```

```
    "action" VARCHAR(10) NOT NULL,
    "old_data" JSONB,
    "new_data" JSONB,
    "changed at" TIMESTAMP DEFAULT NOW(),
    "user" VARCHAR(100) DEFAULT CURRENT_USER
);

-- Функция для логирования
CREATE OR REPLACE FUNCTION "GIBDD"."log_driver_changes"()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO "GIBDD"."driver_hist" (
            "id_водителя", "action", "new_data"
        ) VALUES (
            NEW."id_водителя", 'INSERT', to_jsonb(NEW)
        );
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO "GIBDD"."driver_hist" (
            "id_водителя", "action", "old_data", "new_data"
        ) VALUES (
            OLD."id_водителя", 'UPDATE', to_jsonb(OLD), to_jsonb(NEW)
        );
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO "GIBDD"."driver_hist" (
            "id_водителя", "action", "old_data"
        ) VALUES (
            OLD."id_водителя", 'DELETE', to_jsonb(OLD)
        );
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- Создаем триггер
CREATE OR REPLACE TRIGGER "trg_log_driver_changes"
AFTER INSERT OR UPDATE OR DELETE ON "GIBDD"."Водитель"
FOR EACH ROW EXECUTE FUNCTION "GIBDD"."log_driver_changes"();
```

Пример работы:



7. Триггер для логгирования изменений в таблице Транспортное средство

   Код:

```
```

```sql
-- Создаем таблицу для логов ТС
CREATE TABLE IF NOT EXISTS "GIBDD"."TS_hist" (
    "id_записи" SERIAL PRIMARY KEY,
    "id_TC" INTEGER NOT NULL,
    "action" VARCHAR(10) NOT NULL,
    "old_data" JSONB,
    "new_data" JSONB,
    "changed_at" TIMESTAMP DEFAULT NOW(),
    "user" VARCHAR(100) DEFAULT CURRENT_USER
);

-- Функция для логирования
CREATE OR REPLACE FUNCTION "GIBDD"."log_vehicle_changes"()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO "GIBDD"."TS_hist" (
            "id_TC", "action", "new_data"
        ) VALUES (
            NEW."id_TC", 'INSERT', to_jsonb(NEW)
        );
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO "GIBDD"."TS_hist" (
            "id_TC", "action", "old_data", "new_data"
        ) VALUES (
            NEW."id_TC", 'UPDATE', to_jsonb(OLD), to_jsonb(NEW)
        );
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO "GIBDD"."TS_hist" (
            "id_TC", "action", "old_data"
        ) VALUES (
            OLD."id_TC", 'DELETE', to_jsonb(OLD)
        );
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- Создаем триггер
CREATE TRIGGER "trg_log_vehicle_changes"
AFTER INSERT OR UPDATE OR DELETE ON "GIBDD"."Транспортное_средство"
FOR EACH ROW EXECUTE FUNCTION "GIBDD"."log_vehicle_changes"();
```

Пример работы:

Query | Query History

```sql
INSERT INTO "GIBDD"."Транспортное_средство" (
    "Дата_выпуска", "Номер_ТС", "Категория_ТС"
) VALUES ('2022-01-01', 'A111AA777', 'B');

-- Обновляем данные
UPDATE "GIBDD"."Транспортное_средство"
SET "Номер_ТС" = 'A111AA178'
WHERE "Номер_ТС" = 'A111AA777';

-- Проверяем логи
SELECT * FROM "GIBDD"."TS_hist" ORDER BY "changed_at" DESC;
```

Data Output | Messages | Notifications

SQL

	id_записи [PK] integer	id_TC integer	action character varying (10)	old_data jsonb
1	1	28	INSERT	[null]
2	2	28	UPDATE	{"id_TC": 28, "vin_код_ТС": null, "Марка_ТС": null, "Номер_ТС": "A111AA777", "Модель_ТС": null, "Дата_вы