Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5

«Создание БД PostgreSQL в pgAdmin. Резервное копирование и восстановление БД»

по дисциплине «**Проектирование и реализация баз данных**»

Вариант 17

Автор: Гордеев А. А.

Факультет: ПИН

Группа:К3239

Преподаватель: Говорова М. М.

Санкт-Петербург 2025

ОГЛАВЛЕНИЕ

Цель работы

Овладеть практическими 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.
2. Создать триггеры для индивидуальной БД согласно варианту: 7 оригинальных триггеров

Индивидуальное задание

Описание модели

Вариант 17. БД «Телефонный провайдер»

Описание предметной области: Информационная система служит для хранения информации об абонентах телефонной компании и для учета оплаты всех видов услуг абонентами.

Каждый абонент подключен к определенному тарифу. Тариф определяет базовое количество минут, ГБт, смс. Кроме того, он может подключить дополнительные услуги за отдельную плату. Необходимо знать текущий баланс клиента. У клиента могут быть подключены сторонние ресурсы, требующие оплаты, не зависящие от текущего тарифа. Клиент может менять тариф.

В системе должны храниться сведения о продолжительности разговоров каждого абонента, о стоимости внутренних и международных переговоров, о задолженности абонента.

БД должна содержать следующий минимальный набор сведений: ФИО абонента. Номер телефона. Адрес абонента. Город. Зона (город, республика, СНГ, дальнее зарубежье). Страна. Стоимость тарифа. Сроки действия тарифа. Продолжительность разговора в минутах. Дата звонка. Время звонка. Код зоны. Цена минуты. Сумма оплаты. Дата оплаты. Статус оплаты. Дата фактической оплаты.

Выполнение

Название БД: phone_provider

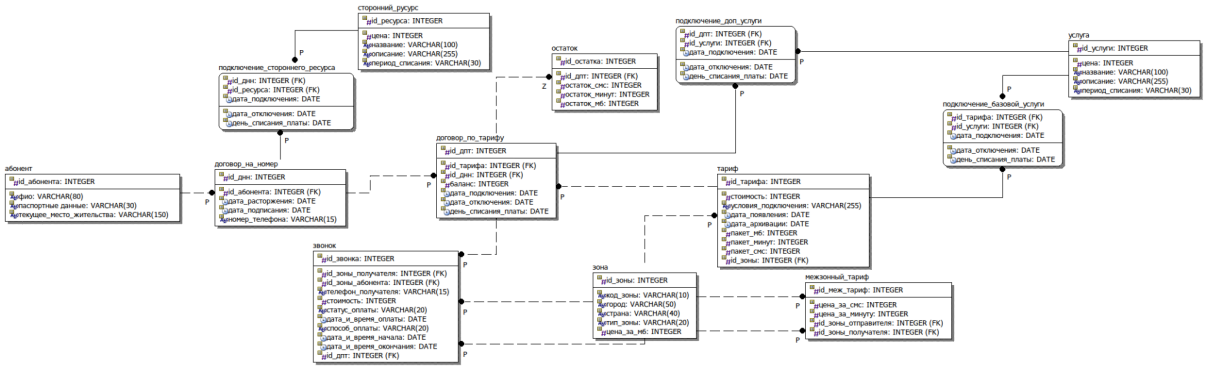

Рисунок 1. Схема инфологической модели IDEF1X

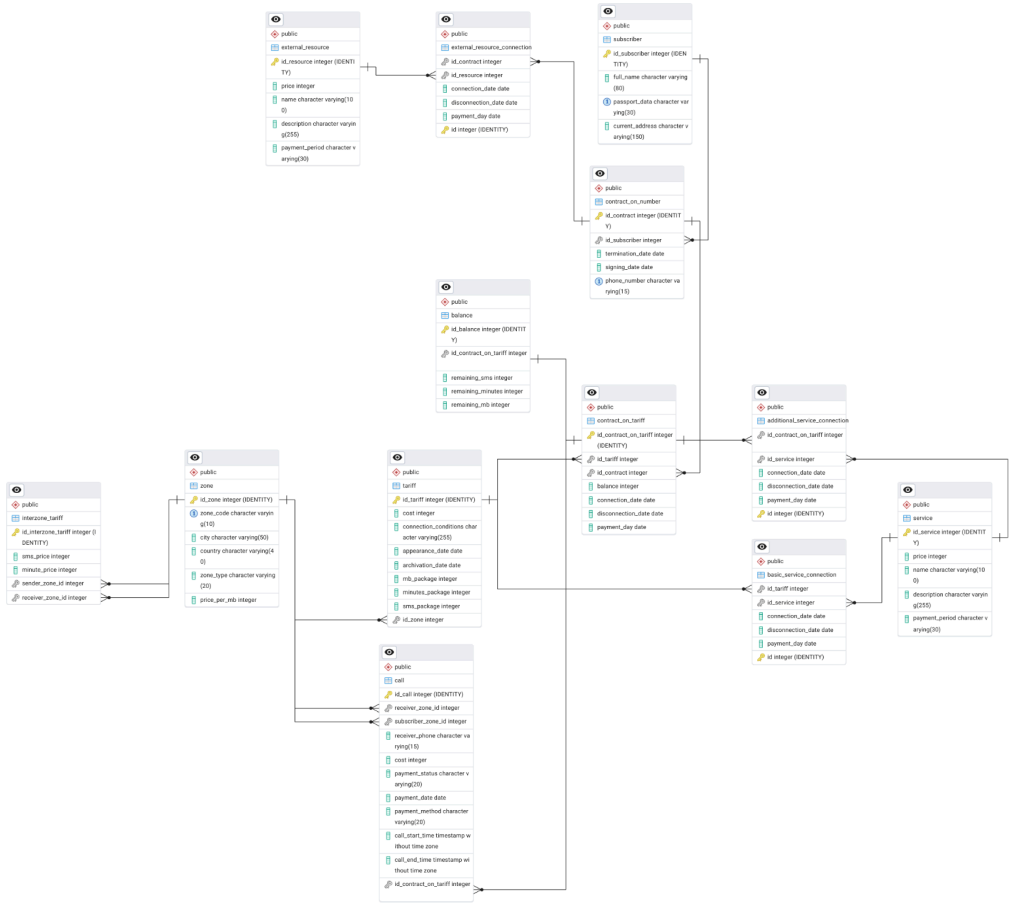

Рисунок 2. Схема логической модели базы данных

Задание 1. Создать процедуры

- добавить данные о новом звонке абонента.

```
phone_provider_v3=# CREATE OR REPLACE PROCEDURE add_new_call(
    p_receiver_zone_id INTEGER,
    p_subscriber_zone_id INTEGER,
    p_receiver_phone VARCHAR(15),
    p_cost INTEGER,
    p_payment_status VARCHAR(20),
    p_payment_date DATE,
    p_payment_method VARCHAR(20),
    p_call_start_time TIMESTAMP WITHOUT TIME ZONE,
    p_call_end_time TIMESTAMP WITHOUT TIME ZONE,
    p_id_contract_on_tariff INTEGER
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO public.call (
        receiver_zone_id,
        subscriber_zone_id,
        receiver_phone,
        cost,
        payment_status,
        payment_date,
        payment_method,
        call_start_time,
        call_end_time,
        id_contract_on_tariff
    )
    VALUES (
        p_receiver_zone_id,
        p_subscriber_zone_id,
        p_receiver_phone,
        p_cost,
        p_payment_status,
        p_payment_date,
        p_payment_method,
        p_call_start_time,
        p_call_end_time,
        p_id_contract_on_tariff
    );
END;
$$;
CREATE PROCEDURE
phone_provider_v3#
```

```
phone_provider_v3=# CALL add_new_call(
    p_receiver_zone_id := 1,
    p_subscriber_zone_id := 2,
    p_receiver_phone := '+79991234567',
    p_cost := 10,
    p_payment_status := 'не оплачено',
    p_payment_date := NULL,
    p_payment_method := 'Рубли',
    p_call_start_time := '2025-05-14 10:00:00'::TIMESTAMP,
    p_call_end_time := '2025-05-14 10:05:00'::TIMESTAMP,
    p_id_contract_on_tariff := 1
);
CALL
phone_provider_v3=# SELECT * FROM call
ORDER BY id_call DESC
LIMIT 1;
 id_call | receiver_zone_id | subscriber_zone_id | receiver_phone | cost | payment_status | payment_date | payment_method |   call_start_time   |    call_end_time    | id_contract_on_tariff
---------+------------------+--------------------+----------------+------+----------------+--------------+----------------+---------------------+---------------------+-----------------------
     104 |                1 |                  2 | +79991234567   |   10 | не оплачено    |              | Рубли          | 2025-05-14 10:00:00 | 2025-05-14 10:05:00 |                     1
(1 строка)

phone_provider_v3#
```

- вывести задолженность по оплате для заданного абонента.

```
phone_provider_v3=# CREATE OR REPLACE FUNCTION get_subscriber_debt(
    p_subscriber_id INTEGER
)
RETURNS INTEGER
LANGUAGE plpgsql
AS $$
DECLARE
    total_debt INTEGER := 0;
BEGIN
    SELECT COALESCE(SUM(c.cost), 0)
    INTO total_debt
    FROM call c
    JOIN contract_on_tariff cot ON c.id_contract_on_tariff = cot.id_contract_on_tariff
    JOIN contract_on_number con ON cot.id_contract = con.id_contract
    WHERE con.id_subscriber = p_subscriber_id
      AND c.payment_status = 'не оплачено';

    RETURN total_debt;
END;
$$;
CREATE FUNCTION
```

```
phone_provider_v3=# SELECT get_subscriber_debt(1);
 get_subscriber_debt
---------------------
                  10
(1 строка)

phone_provider_v3=# SELECT get_subscriber_debt(3);
 get_subscriber_debt
---------------------
                   0
(1 строка)
```

- рассчитать общую стоимость звонков по каждой зоне за истекшую неделю.

```
phone_provider_v3=# CREATE OR REPLACE FUNCTION get_weekly_call_cost_by_origin_zone()
RETURNS TABLE (
    zone_id INTEGER,
    zone_code VARCHAR,
    city VARCHAR,
    country VARCHAR,
    total_call_cost BIGINT
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        z.id_zone,
        z.zone_code,
        z.city,
        z.country,
        COALESCE(SUM(c.cost), 0)
    FROM call c
    JOIN zone z ON c.subscriber_zone_id = z.id_zone
    WHERE c.call_start_time >= NOW() - INTERVAL '7 days'
    GROUP BY z.id_zone, z.zone_code, z.city, z.country
    ORDER BY COALESCE(SUM(c.cost), 0) DESC;
END;
$$;
CREATE FUNCTION
```

```
phone_provider_v3=# SELECT * FROM get_weekly_call_cost_by_origin_zone();
 zone_id | zone_code |      city       | country | total_call_cost
---------+-----------+-----------------+---------+-----------------
       2 | 812       | Санкт-Петербург | Россия  |             670
       1 | 495       | Москва          | Россия  |             630
       3 | 383       | Новосибирск     | Россия  |              70
       7 | 351       | Челябинск       | Россия  |              65
       4 | 343       | Екатеринбург    | Россия  |              55
      14 | 1         | Нью-Йорк        | США     |               0
(6 строк)
```

Задание 2. Создать 7 триггеров

1. Триггер на логирование изменений адреса абонента.

```
phone_provider_v3=# CREATE TABLE subscriber_address_log (
    log_id SERIAL PRIMARY KEY,
    subscriber_id INTEGER NOT NULL,
    old_address VARCHAR(150),
    new_address VARCHAR(150),
    change_timestamp TIMESTAMP WITHOUT TIME ZONE DEFAULT NOW(),
    FOREIGN KEY (subscriber_id) REFERENCES subscriber(id_subscriber) ON DELETE CASCADE
);
CREATE TABLE
phone_provider_v3=# CREATE OR REPLACE FUNCTION log_subscriber_address_change()
RETURNS TRIGGER AS $$
BEGIN
    IF OLD.current_address IS DISTINCT FROM NEW.current_address THEN
        INSERT INTO subscriber_address_log (subscriber_id, old_address, new_address, change_timestamp)
        VALUES (OLD.id_subscriber, OLD.current_address, NEW.current_address, NOW());
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
phone_provider_v3=# CREATE TRIGGER trigger_log_subscriber_address
AFTER UPDATE OF current_address ON subscriber
FOR EACH ROW
EXECUTE FUNCTION log_subscriber_address_change();
CREATE TRIGGER
```

```
phone_provider_v3=# SELECT id_subscriber, full_name, current_address FROM subscriber WHERE id_subscriber
= 1;
 id_subscriber |      full_name       |          current_address
---------------+----------------------+-----------------------------------
             1 | Иванов Иван Иванович | г. Москва, ул. Ленина, д. 1, кв. 1
(1 строка)

phone_provider_v3=# UPDATE subscriber
SET current_address = 'г. Москва, ул. Пушкина, д. 10, кв. 5'
WHERE id_subscriber = 1;
UPDATE 1
phone_provider_v3=# SELECT * FROM subscriber_address_log WHERE subscriber_id = 1 ORDER BY change_timestam
p DESC LIMIT 1;
 log_id | subscriber_id |            old_address             |             new_address              |
  change_timestamp
--------+---------------+------------------------------------+--------------------------------------+----
-----------------------
      1 |             1 | г. Москва, ул. Ленина, д. 1, кв. 1 | г. Москва, ул. Пушкина, д. 10, кв. 5 | 202
5-05-14 14:51:34.391723
(1 строка)
```

2. Триггер на автоматическое создание записи в таблице баланса.

```
phone_provider_v3=# CREATE OR REPLACE FUNCTION create_balance_for_new_contract_on_tariff()
RETURNS TRIGGER AS $$
DECLARE
    initial_mb INTEGER;
    initial_minutes INTEGER;
    initial_sms INTEGER;
BEGIN
    SELECT mb_package, minutes_package, sms_package
    INTO initial_mb, initial_minutes, initial_sms
    FROM tariff
    WHERE id_tariff = NEW.id_tariff;

    INSERT INTO balance (id_contract_on_tariff, remaining_sms, remaining_minutes, remaining_mb)
    VALUES (NEW.id_contract_on_tariff, initial_sms, initial_minutes, initial_mb);

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
phone_provider_v3=# CREATE TRIGGER trigger_create_balance_on_contract_insert
AFTER INSERT ON contract_on_tariff
FOR EACH ROW
EXECUTE FUNCTION create_balance_for_new_contract_on_tariff();
CREATE TRIGGER
```

```
phone_provider_v3=# SELECT MAX(id_contract_on_tariff) FROM contract_on_tariff;
 max
-----
  22
phone_provider_v3=# INSERT INTO contract_on_tariff (id_tariff, id_contract, balance, connection_date, payment_day)
VALUES (20, 1, 1000, CURRENT_DATE, CURRENT_DATE);
INSERT 0 1
phone_provider_v3=# SELECT id_contract_on_tariff FROM contract_on_tariff WHERE id_contract = 1 AND id_tariff = 20 OR
DER BY connection_date DESC LIMIT 1;
 id_contract_on_tariff
-----------------------
                    23
(1 строка)

phone_provider_v3=# SELECT * FROM balance WHERE id_contract_on_tariff = (SELECT id_contract_on_tariff FROM contract_
on_tariff WHERE id_contract = 1 AND id_tariff = 20 ORDER BY connection_date DESC LIMIT 1);
 id_balance | id_contract_on_tariff | remaining_sms | remaining_minutes | remaining_mb
------------+-----------------------+---------------+-------------------+--------------
         17 |                    23 |            60 |               450 |        15360
(1 строка)
```

3. Триггер на обновление баланса после завершения звонка.

```
phone_provider_v3=# CREATE OR REPLACE FUNCTION update_balance_after_call()
RETURNS TRIGGER AS $$
DECLARE
    call_duration_minutes INTEGER;
BEGIN
    IF NEW.payment_method = 'Пакет' THEN
        call_duration_minutes := CEIL(EXTRACT(EPOCH FROM (NEW.call_end_time - NEW.call_start_time)) / 60.0);

        UPDATE balance
        SET remaining_minutes = remaining_minutes - call_duration_minutes
        WHERE id_contract_on_tariff = NEW.id_contract_on_tariff;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
phone_provider_v3=# CREATE TRIGGER trigger_update_balance_on_call_insert
AFTER INSERT ON call
FOR EACH ROW
EXECUTE FUNCTION update_balance_after_call();
CREATE TRIGGER
```

```
phone_provider_v3=# SELECT remaining_minutes FROM balance WHERE id_contract_on_tariff = 1;
 remaining_minutes
-------------------
               450
(1 строка)
```

```
phone_provider_v3=# CALL add_new_call(
    p_receiver_zone_id := 1,
    p_subscriber_zone_id := 1,
    p_receiver_phone := '+79998887766',
    p_cost := 0,
    p_payment_status := 'оплачено',
    p_payment_date := NOW()::DATE,
    p_payment_method := 'Пакет',
    p_call_start_time := (NOW() - INTERVAL '3 minutes')::TIMESTAMP WITHOUT TIME ZONE,
    p_call_end_time := NOW()::TIMESTAMP WITHOUT TIME ZONE,
    p_id_contract_on_tariff := 1
);
CALL
phone_provider_v3=# SELECT remaining_minutes FROM balance WHERE id_contract_on_tariff = 1;
 remaining_minutes
-------------------
               447
(1 строка)
```

4. Триггер на предотвращение звонков с неактивного контракта на тариф.

```
phone_provider_v3=# CREATE OR REPLACE FUNCTION prevent_call_on_inactive_contract()
RETURNS TRIGGER AS $$
DECLARE
    contract_disconnection_date DATE;
BEGIN
    SELECT disconnection_date
    INTO contract_disconnection_date
    FROM contract_on_tariff
    WHERE id_contract_on_tariff = NEW.id_contract_on_tariff;

    IF contract_disconnection_date IS NOT NULL AND contract_disconnection_date <= CURRENT_DATE THEN
        RAISE EXCEPTION 'Cannot add call: contract on tariff with ID % is inactive.', NEW.id_contract_on_tariff;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
phone_provider_v3=# CREATE TRIGGER trigger_prevent_call_on_inactive_contract
BEFORE INSERT ON call
FOR EACH ROW
EXECUTE FUNCTION prevent_call_on_inactive_contract();
CREATE TRIGGER
phone_provider_v3=#
```

```
phone_provider_v3=# CALL add_new_call(
        p_receiver_zone_id := 1,
        p_subscriber_zone_id := 1,
        p_receiver_phone := '+79991112233',
        p_cost := 10,
        p_payment_status := 'не оплачено',
        p_payment_date := NULL,
        p_payment_method := 'Рубли',
        p_call_start_time := (NOW() - INTERVAL '3 minutes')::TIMESTAMP WITHOUT TIME ZONE,
        p_call_end_time := NOW()::TIMESTAMP WITHOUT TIME ZONE,
        p_id_contract_on_tariff := 6
    );
ERROR:  Cannot add call: contract on tariff with ID 6 is inactive.
```

5. Триггер на поддержание счетчика активных контрактов на номер для каждого абонента.

```
phone_provider_v3=# ALTER TABLE subscriber
ADD COLUMN active_contract_count INTEGER DEFAULT 0;
ALTER TABLE
phone_provider_v3=# UPDATE subscriber s
SET active_contract_count = (
    SELECT COUNT(*)
    FROM contract_on_number con
    WHERE con.id_subscriber = s.id_subscriber
      AND (con.termination_date IS NULL OR con.termination_date > CURRENT_DATE)
);
UPDATE 21
```

```
phone_provider_v3=# CREATE OR REPLACE FUNCTION update_subscriber_active_contract_count()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'INSERT') THEN
        UPDATE subscriber
        SET active_contract_count = active_contract_count + 1
        WHERE id_subscriber = NEW.id_subscriber;
        RETURN NEW;

    ELSIF (TG_OP = 'DELETE') THEN
        IF OLD.termination_date IS NULL OR OLD.termination_date > CURRENT_DATE THEN
            UPDATE subscriber
            SET active_contract_count = active_contract_count - 1
            WHERE id_subscriber = OLD.id_subscriber;
        END IF;
        RETURN OLD;

    ELSIF (TG_OP = 'UPDATE' AND OLD.termination_date IS DISTINCT FROM NEW.termination_date) THEN
        IF (OLD.termination_date IS NULL OR OLD.termination_date > CURRENT_DATE) AND (NEW.termination_date IS NOT NU
LL AND NEW.termination_date <= CURRENT_DATE) THEN
            UPDATE subscriber
            SET active_contract_count = active_contract_count - 1
            WHERE id_subscriber = OLD.id_subscriber;
        ELSIF (OLD.termination_date IS NOT NULL AND OLD.termination_date <= CURRENT_DATE) AND (NEW.termination_date
IS NULL OR NEW.termination_date > CURRENT_DATE) THEN
             UPDATE subscriber
            SET active_contract_count = active_contract_count + 1
            WHERE id_subscriber = OLD.id_subscriber;
        END IF;
        RETURN NEW;
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
phone_provider_v3=# CREATE TRIGGER trigger_update_active_contract_count_insert
AFTER INSERT ON contract_on_number
FOR EACH ROW
EXECUTE FUNCTION update_subscriber_active_contract_count();
CREATE TRIGGER
phone_provider_v3=# CREATE TRIGGER trigger_update_active_contract_count_delete
AFTER DELETE ON contract_on_number
FOR EACH ROW
EXECUTE FUNCTION update_subscriber_active_contract_count();
CREATE TRIGGER
phone_provider_v3=# CREATE TRIGGER trigger_update_active_contract_count_update
AFTER UPDATE OF termination_date ON contract_on_number
FOR EACH ROW
EXECUTE FUNCTION update_subscriber_active_contract_count();
CREATE TRIGGER
```

```
phone_provider_v3=# SELECT id_subscriber, full_name, active_contract_count FROM subscriber WHERE id_subscriber = 1;
 id_subscriber |      full_name       | active_contract_count
---------------+----------------------+----------------------
             1 | Иванов Иван Иванович |                    2
(1 строка)

phone_provider_v3=# SELECT id_contract, phone_number, termination_date FROM contract_on_number WHERE id_subscriber =
 1;
 id_contract | phone_number | termination_date
-------------+--------------+-----------------
           1 | +79101234567 |
          16 | +79107654321 |
(2 строки)

phone_provider_v3=# INSERT INTO contract_on_number (id_subscriber, signing_date, phone_number)
VALUES (1, CURRENT_DATE, '+79109998877');
INSERT 0 1
phone_provider_v3=# SELECT id_subscriber, full_name, active_contract_count FROM subscriber WHERE id_subscriber = 1;
 id_subscriber |      full_name       | active_contract_count
---------------+----------------------+----------------------
             1 | Иванов Иван Иванович |                    3
(1 строка)
```

6. Триггер на предотвращение удаления Зоны, если с ней связаны активные Тарифы.

```
phone_provider_v3=# CREATE OR REPLACE FUNCTION zone_deletion_if_active_tariffs_exist()
RETURNS TRIGGER AS $$
DECLARE
    active_tariff_count INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO active_tariff_count
    FROM tariff
    WHERE id_zone = OLD.id_zone
      AND (archivation_date IS NULL OR archivation_date > CURRENT_DATE);

    IF active_tariff_count > 0 THEN
        RAISE EXCEPTION 'Cannot delete zone with ID %: active tariffs are linked to this zone.', OLD.id_zone;
    END IF;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
phone_provider_v3=# CREATE TRIGGER trigger_zone_deletion_with_active_tariffs
BEFORE DELETE ON zone
FOR EACH ROW
EXECUTE FUNCTION zone_deletion_if_active_tariffs_exist();
CREATE TRIGGER
```

```
phone_provider_v3=# SELECT t.id_tariff, t.archivation_date
FROM tariff t
WHERE t.id_zone = 1 AND (t.archivation_date IS NULL OR t.archivation_date > CURRENT_DATE);
 id_tariff | archivation_date
-----------+------------------
         1 |
         2 |
(2 строки)

phone_provider_v3=# DELETE FROM zone WHERE id_zone = 1;
ERROR:  Cannot delete zone with ID 1: active tariffs are linked to this zone.
КОНТЕКСТ:  PL/pgSQL function zone_deletion_if_active_tariffs_exist() line 12 at RAISE
```

7. Когда проектировал БД, я добавил валидацию времени в ограничения таблицы. Это было ошибкой, так как ломало восстановление БД из бекапа. Переписываю на триггеры.

```
phone_provider_v3=# CREATE OR REPLACE FUNCTION check_additional_service_connection_date()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.connection_date < CURRENT_DATE THEN
        RAISE EXCEPTION 'Cannot add additional service connection: connection_date (%) must be today or in the futur
e.', NEW.connection_date;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
phone_provider_v3=# CREATE TRIGGER trigger_check_additional_service_connection_date
BEFORE INSERT ON additional_service_connection
FOR EACH ROW
EXECUTE FUNCTION check_additional_service_connection_date();
CREATE TRIGGER
```

```
phone_provider_v3=# INSERT INTO additional_service_connection (id_contract_on_tariff, id_service, connection_date, d
isconnection_date, payment_day)
    VALUES (1, 12, CURRENT_DATE - INTERVAL '1 day', NULL, CURRENT_DATE - INTERVAL '1 day');
ERROR:  Cannot add additional service connection: connection_date (2025-05-13) must be today or in the future.
КОНТЕКСТ:  PL/pgSQL function check_additional_service_connection_date() line 4 at RAISE
```

Для basic_service_connection, contract_on_tariff, contract_on_number все аналогично.

Вывод

По результатам данной лабораторной работы были изучены и применены на практике хранимые процедуры, функции и триггеры. Самостоятельная разработка и тестирование этих элементов для реализованной базы данных позволили понять принципы их работы и получить опыт создания качественных баз данных.