5НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт до лабораторної роботи №5 з дисципліни

«Бази даних»

Прийняв:
Викладач кафедри ІПІ
Марченко О. І.
21 листопада 2021 року

Виконав
Студент групи ІТ-01
Бардін В. Д.

Київ – 2021

Лабораторна робота №5

Тема: Побудова простих запитів

Мета:

- Вивчити правила побудови ідентифікаторів, правила визначення змінних та типів. Визначити правила роботи з циклами та умовними конструкціями, роботу зі змінними типу Table.
- Вивчити синтаксис та семантику функцій та збережених процедур, способів їх ідентифікації, методів визначення та специфікації параметрів та повертаємих значень, виклик функцій та збережених процедур.
- Застосування команд для створення, зміни та видалення як скалярних, так і табличних функцій, збережених процедур.
- Вивчити призначення та типи курсорів, синтаксис та семантику команд мови SQL для створення курсорів, вибірки даних з курсорів, зміни даних із застосуванням курсорів.
- Вивчити призначення та типи тригерів, умов їх активації, синтаксису та семантики для їх створення, модифікації, перейменування, програмування та видалення.
- Вивчити правила побудови ідентифікаторів, правила визначення змінних та типів. Визначити правила роботи з циклами та умовними конструкціями, роботу зі змінними типу Table.
- Вивчити синтаксис та семантику функцій та збережених процедур, способів їх ідентифікації, методів визначення та специфікації параметрів та повертаємих значень, виклик функцій та збережених процедур.
- Застосування команд для створення, зміни та видалення як скалярних, так і табличних функцій, збережених процедур.
- Вивчити призначення та типи курсорів, синтаксис та семантику команд мови SQL для створення курсорів, вибірки даних з курсорів, зміни даних із застосуванням курсорів.
- Вивчити призначення та типи тригерів, умов їх активації, синтаксису та семантики для їх створення, модифікації, перейменування, програмування та видалення.

Завдання: Програмне забезпечення «Діяльність фірми з розробки програмних продуктів». Підприємства, о розробляють ПЗ, зазвичай мають декілька відділів, а саме: дирекція, бухгалтерія, маркетинговий відділ, відділ розробки ПЗ, відділ тестування ПЗ, відділ супроводження тощо. ПЗ, котре поставляється Замовнику, має назву, список розробників (внутрішній список тестувальників, котрий Замовнику не надається), вартість, документацію, дистрибутив, правила використання. Замовниками можуть бути як фізичні так і юридичні особи. Кожний Замовник має можливість замовити декілька ПП, на кожний з яких він отримує ліцензію, в якій вказано назву продукту, дату продажу, вартість, терміни апгрейдів.

Схема спроектованої бази даних

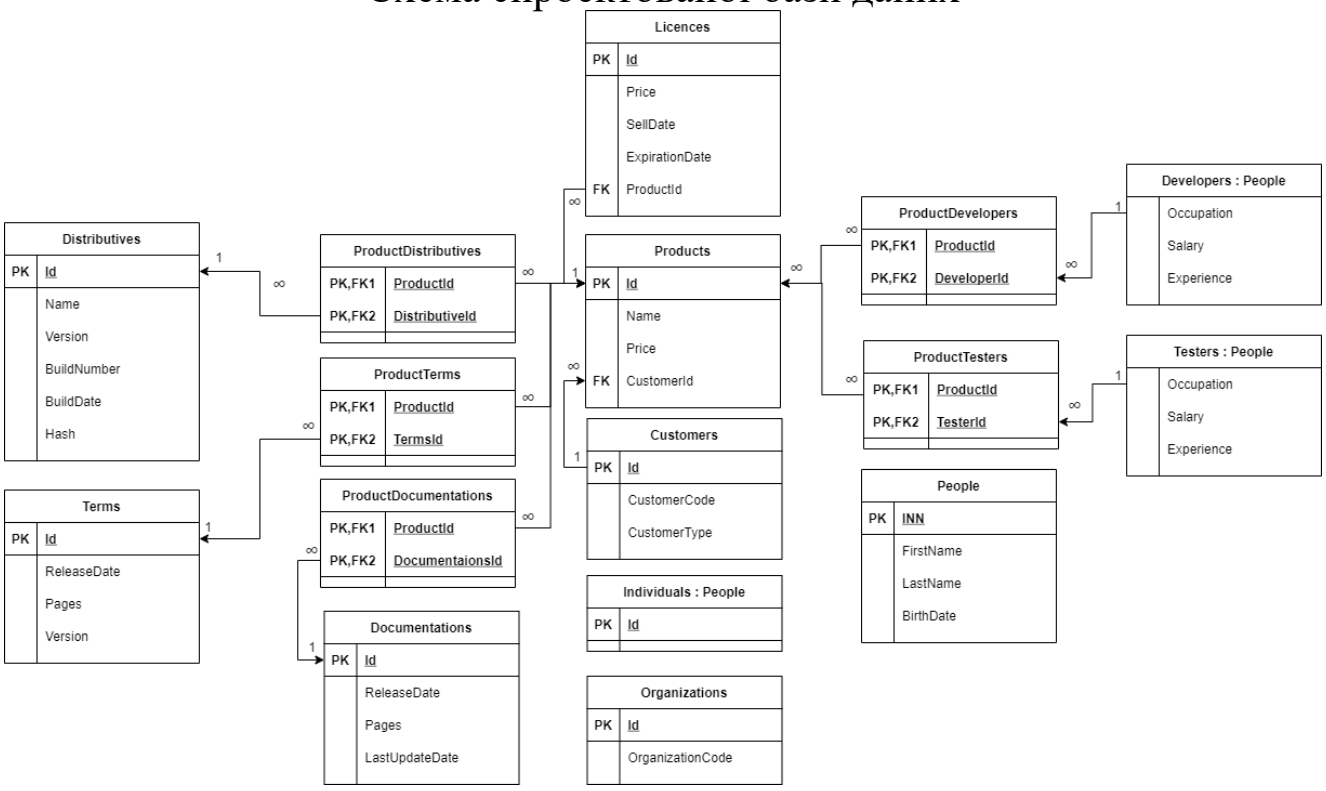

Рис. 1 — Схема БД

Для створення запитів буде використано базу попередніх лабораторних робіт. Схему БД наведено на рис. 1.

Частина 1: Збережені процедури

```
-- 1.a

CREATE OR REPLACE PROCEDURE sp_create_temp_table()
AS
$$
BEGIN
    CREATE TEMP TABLE IF NOT EXISTS devs_temp AS
    SELECT concat(D."FirstName", ' ', D."LastName")
    FROM "Developers" AS D;
END;
$$ LANGUAGE plpgsql;

CALL sp_create_temp_table();
SELECT *
FROM devs_temp;
DROP TABLE IF EXISTS devs_temp;

-- 1.b

CREATE OR REPLACE FUNCTION check_if_table_exists(schema_name varchar, _table_name
varchar)
    RETURNS BOOLEAN
AS
$$
SELECT EXISTS(
               SELECT
               FROM information_schema.tables
               WHERE table_schema = schema_name
                 AND table_name = _table_name
           );
```

```
$$ LANGUAGE sql;

CREATE OR REPLACE PROCEDURE sp_create_devs_temp_if_not_exists()
AS
$$
BEGIN
    IF check_if_table_exists('public', 'devs_temp') = FALSE THEN
        CALL sp_create_temp_table();
    END if;
END;
$$ LANGUAGE plpgsql;

CALL sp_create_devs_temp_if_not_exists();
SELECT *
FROM devs_temp;
DROP TABLE IF EXISTS devs_temp;

-- 1.c

CREATE OR REPLACE PROCEDURE sp_create_and_fill_table()
AS
$$
DECLARE
    i_increment INT := 1;
    i_current   INT := 1;
    i_end       INT := 350;
BEGIN
    DROP TABLE IF EXISTS test_1c;

    CREATE TEMP TABLE test_1c
    (
        num INTEGER
    );

    WHILE i_current <= i_end
        LOOP
            i_current := i_current + i_increment;
            INSERT INTO test_1c (num)
            SELECT i_current;
        END LOOP;
END;
$$
    LANGUAGE plpgsql;

CALL sp_create_and_fill_table();

SELECT *
FROM test_1c;
DROP TABLE IF EXISTS test_1c;

-- 1.d
-- this was done at 1.a, 1.b and 1.c

-- 1.e

CREATE OR REPLACE PROCEDURE get_test_1c_stat(
    out min_val int,
    out max_val int,
    out avg_val numeric)
AS
$$
BEGIN
    SELECT min(num),
           max(num),
           avg(num)
    INTO min_val, max_val, avg_val
```

```
    FROM test_1c;

END ;
$$ LANGUAGE plpgsql;

CALL sp_create_and_fill_table();
CALL get_test_1c_stat(0, 0, 0);
DROP TABLE IF EXISTS test_1c;

-- 1.f
-- impossible to complete at PostgreSQL because stored procedures can't RETURN
anything

-- 1.g
CREATE OR REPLACE PROCEDURE sp_update_table()
AS
$$
    UPDATE test_1c
    SET num = 1;
$$
    LANGUAGE sql;

CALL sp_create_and_fill_table();
SELECT *
FROM test_1c;
CALL sp_update_table();
SELECT *
FROM test_1c;
DROP TABLE IF EXISTS test_1c;

-- 1.f
CREATE OR REPLACE PROCEDURE sp_select_all_developers()
AS
$$
SELECT 1
INTO b;
$$
    LANGUAGE sql;

CALL sp_select_all_developers();
SELECT *
FROM b;
DROP TABLE IF EXISTS b;
```

Частина 2: Функції

```
-- 2.a
CREATE OR REPLACE FUNCTION check_if_table_exists(schema_name varchar, _table_name
varchar)
    RETURNS BOOLEAN
AS
$$
SELECT EXISTS(
               SELECT
               FROM information_schema.tables
               WHERE table_schema = schema_name
                 AND table_name = _table_name
           );
$$ LANGUAGE sql;

SELECT check_if_table_exists('public', 'Developers');

-- 2.b
CREATE TABLE t1
(
    id   serial PRIMARY KEY,
```

```
    col1 text
    -- infowindow does not exist
);
INSERT INTO t1(col1)
VALUES ('foo1'),
       ('bar1');

CREATE OR REPLACE FUNCTION f_tbl_plus_info_window(_table regclass)
    RETURNS void AS -- no direct return type
$func$
DECLARE
    -- appending _tmp for temp table
    _tmp text := quote_ident(_table::text || '_tmp');
BEGIN

    -- Create temp table only for duration of transaction
    EXECUTE format(
            'CREATE TEMP TABLE %s ON COMMIT DROP AS TABLE %s LIMIT 0', _tmp,
_table);

    IF EXISTS(
            SELECT 1
            FROM pg_attribute a
            WHERE a.attrelid = _table
              AND a.attname = 'infowindow'
              AND a.attisdropped = FALSE)
    THEN
        EXECUTE format('INSERT INTO %s SELECT * FROM %s', _tmp, _table);
    ELSE
        -- This is assuming a NOT NULL column named "id"!
        EXECUTE format($x$
      ALTER  TABLE %1$s ADD COLUMN infowindow text;
      INSERT INTO %1$s
      SELECT *, 'ID: ' || id::text
      FROM   %2$s $x$
            , _tmp, _table);
    END IF;

END
$func$ LANGUAGE plpgsql;

BEGIN;
SELECT f_tbl_plus_info_window('t1');
SELECT *
FROM t1_tmp; -- do something with the returned rows
COMMIT;

-- 2.c
CREATE OR REPLACE FUNCTION get_developers(_pattern VARCHAR)
    RETURNS TABLE
            (
                dev_full_name VARCHAR
            )
    LANGUAGE plpgsql
AS
$$
BEGIN
    RETURN QUERY
        SELECT concat(d."FirstName", ' ', D."LastName")::VARCHAR as dev_full_name
        FROM "Developers" AS D
        WHERE concat(d."FirstName", ' ', D."LastName") ILIKE _pattern;
END;
$$;

SELECT * FROM get_developers ('Vl%');
```

Частина 3: Робота з курсорами

```
CREATE TABLE test
(
    col text
);
INSERT INTO test
VALUES ('123');

CREATE FUNCTION reffunc(refcursor)
    RETURNS refcursor
AS
$$
BEGIN
    OPEN $1 FOR SELECT col FROM test;
    RETURN $1;
END;
$$
    LANGUAGE plpgsql;

BEGIN;
SELECT reffunc('funccursor');
FETCH ALL IN funccursor;
COMMIT;
```

Частина 4: Робота з тригерами

```
CREATE TABLE ProductsAudit
(
    "Operation" CHAR(1)   NOT NULL,
    "Stamp"     TIMESTAMP NOT NULL,
    "UserId"    TEXT      NOT NULL,
    "ProductId" UUID      NOT NULL
);

CREATE OR REPLACE FUNCTION process_products_audit() RETURNS TRIGGER AS
$product_audit$
BEGIN
    --
    -- Create a row in emp_audit to reflect the operation performed on emp,
    -- making use of the special variable TG_OP to work out the operation.
    --
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO ProductsAudit SELECT 'D', now(), user, OLD."Id";
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO ProductsAudit SELECT 'U', now(), user, NEW."Id";
    ELSIF (TG_OP = 'INSERT') THEN
        INSERT INTO ProductsAudit SELECT 'I', now(), user, NEW."Id";
    END IF;
    RETURN NULL; -- result is ignored since this is an AFTER trigger
END;
$product_audit$ LANGUAGE plpgsql;

CREATE TRIGGER products_audit
    AFTER INSERT OR UPDATE OR DELETE
    ON "Products"
    FOR EACH ROW
EXECUTE FUNCTION process_products_audit();

INSERT INTO "Products" ("Id", "Price", "CustomerId")
VALUES ('1a116c2e-ec71-416c-afc0-a28e24ad7b0c', 1491122, '782e88d2-e7ba-433b-b3f9-
7cea29f8fbfc');
```

```
UPDATE "Products"
SET "Price" = 10000000
WHERE "Id" = '1a116c2e-ec71-416c-afc0-a28e24ad7b0c';
DELETE FROM "Products" WHERE "Id" = '1a116c2e-ec71-416c-afc0-a28e24ad7b0c';
```

Висновок:

В результаті виконання даної лабораторної роботи було створено запити з використанням ф-цій агрегації, відображення, а також системних збережених процедур.