

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №5 з дисципліни
«Бази даних»

**«Основи програмування з використанням мови SQL.
Збережені процедури. Курсори.
Створення, програмування та керування тригерами.»**

Варіант 1

Виконав(ла) ІП-35 Адаменко Арсен Богданович _____

Перевірив Марченко Олена Іванівна _____

Київ 2024

Лабораторна робота №4

«Основи програмування з використанням мови SQL.

Збережені процедури. Курсори.

Створення, програмування та керування тригерами.»

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 значень, котрі повертаються, 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.

Постановка задачі лабораторної роботи №5

При виконанні лабораторної роботи необхідно виконати наступні дії:

1) Збережені процедури:

- a. створення процедури, в якій використовується тимчасова таблиця, котра створена через змінну типу TABLE;
- b. створення процедури з використанням умовної конструкції IF;
- c. створення процедури з використанням циклу WHILE;
- d. створення процедури без параметрів;
- e. створення процедури з вхідним параметром та RETURN;
- f. створення процедури оновлення даних в деякій таблиці БД;
- g. створення процедури, в котрій робиться вибірка даних.

2) Функції:

- a. створити функцію, котра повертає деяке скалярне значення;
- b. створити функцію, котра повертає таблицю з динамічним набором стовпців;
- c. створити функцію, котра повертає таблицю наперед заданої структури.

3) Робота з курсорами (створити процедуру, в котрій демонструються наведені нижче дії):

- a. створення курсору;
- b. відкриття курсору;
- c. вибірка даних;
- d. робота з курсорами.

4) Робота з тригерами:

- а. створити тригер, котрий буде спрацьовувати при видаленні даних;
- б. створити тригер, котрий буде спрацьовувати при модифікації даних;
- с. створити тригер, котрий буде спрацьовувати при додаванні даних.

5) Оформити звіт з роботи. В звіт включити тексти кодів збережених процедур, функцій, тригерів, їх словесний опис та результати виконання.

Варіант 1

Програмне забезпечення військової частини. Військові частини округу розквартировані по різних місцях дислокації, причому в одному місці можуть розташовуватися кілька частин. Кожна військова частина складається з рот, роти з взводів, взводи з відділень, в свою чергу військові частини об'єднуються в дивізії, корпуси або бригади, а ті в армії. Військовий округ представлений офіцерським складом (генерали, полковники, підполковники, майори, капітани, лейтенанти) і рядовим і сержантським складом (старшини, сержанти, прапорщики, рядові). Кожна з перерахованих категорій військовослужбовців може мати характеристики, властиві тільки цій категорії: для генералів це може бути дата закінчення академії, дата присвоєння генеральського звання і т.д. Кожне з підрозділів має командира, причому військовослужбовці офіцерського складу можуть командувати будь-яким з перерахованих вище підрозділів, а військовослужбовці рядового і сержантського складу тільки взводом і відділенням. Всі військовослужбовці мають одну або кілька військових спеціальностей. Кожна військова частина має бойову і транспортну техніку: БМП, тягачі, автотранспорт тощо. і озброєння: карабіни, автоматична зброя, артилерія, ракетне озброєння тощо. Кожна з перерахованих категорій бойової техніки і озброєння також має специфічні, притаманні лише їй атрибути і по кожній категорії може бути кілька видів техніки і озброєння. Треба мати можливість отримувати інформацію про всі частини військового округу, дані про офіцерський, рядовий та сержантський склад, отримувати місця дислокації, дані про наявне озброєння тощо.

1.a. Отримання кількості всіх військових:

```
CREATE OR REPLACE FUNCTION servicemen_count()
RETURNS INT AS $$
DECLARE
    record_count INT;
BEGIN
    CREATE TEMPORARY TABLE serviceman_unit_temp AS
    SELECT s.fio AS serviceman_name, mu.name_ AS unit_name
    FROM serviceman s
        LEFT JOIN serviceman_affiliation sa ON sa.serviceman_id = s.id_ AND
sa.unit_type = 'military_unit'
        LEFT JOIN military_unit mu ON sa.unit_id = mu.id_;

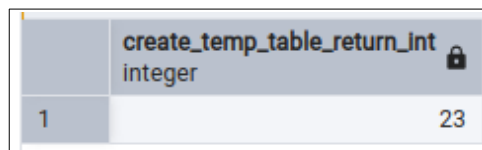
    SELECT COUNT(*) INTO record_count FROM serviceman_unit_temp;

    DROP TABLE serviceman_unit_temp;

    RETURN record_count;

    RAISE NOTICE 'Temporary table "serviceman_unit_temp" created successfully
with % records.', record_count;
END;
$$ LANGUAGE plpgsql;

select create_temp_table_return_int();
```



create_temp_table_return_int
1

1.b. Процедура оновлює тип ресурсу залежно від переданого параметра. Якщо тип невідомий, виводиться повідомлення:

```
CREATE OR REPLACE FUNCTION update_resource_type(resource_id INT,
new_type VARCHAR)
RETURNS VOID AS $$
BEGIN
    IF new_type = 'rocket' THEN
        UPDATE resource SET type_ = 'rocket' WHERE id_ = resource_id;
    ELSEIF new_type = 'artillery' THEN
        UPDATE resource SET type_ = 'artillery' WHERE id_ = resource_id;
    ELSE
```

```

        RAISE NOTICE 'Unsupported resource type: %', new_type;
    END IF;
END;
$$ LANGUAGE plpgsql;

```

```
select update_resource_type(3, 'rocket');
```

```
select * from resource;
```

27	28	weapon12	weapon	28
28	29	weapon13	weapon	29
29	30	weapon14	weapon	30
30	31	weapon15	weapon	31
31	32	weapon16	weapon	32
32	3	transport3	rocket	3

1.c. створення процедури з використанням циклу WHILE;

```

CREATE OR REPLACE FUNCTION calculate_speciality_count()
RETURNS INT AS $$
DECLARE
    serviceman_count INT;
    current_id INT := 1;
    speciality_sum INT := 0;
BEGIN
    SELECT COUNT(*) INTO serviceman_count FROM serviceman;

    WHILE current_id <= serviceman_count LOOP
        SELECT speciality_sum + COUNT(*)
        INTO speciality_sum
        FROM serviceman_speciality
        WHERE serviceman_id = current_id;

        current_id := current_id + 1;
    END LOOP;

    RETURN speciality_sum;
END;
$$ LANGUAGE plpgsql;

select calculate_speciality_count();

```

	calculate_speciality_count	
	integer	
1		31

1.d. Процедура повертає перелік усіх ресурсів із бази даних:

```
CREATE OR REPLACE FUNCTION list_all_resources()
RETURNS TABLE(id INT, name VARCHAR, type VARCHAR) AS $$
BEGIN
    -- Вибірка всіх ресурсів із таблиці resource
    RETURN QUERY SELECT id_, name_, type_ FROM resource;
END;
$$ LANGUAGE plpgsql;

select list_all_resources();
```

	list_all_resources	
	record	
1	1,transport1,transport	
2	2,transport2,transport	
3	3,transport3,transport	
4	4,transport4,transport	
5	5,transport5,transport	
6	6,transport6,transport	
7	7,transport7,transport	

1.e. Процедура приймає ID ресурсу та повертає його назву:

```
CREATE OR REPLACE FUNCTION get_resource_name(resource_id INT)
RETURNS VARCHAR AS $$
DECLARE
    resource_name VARCHAR;
BEGIN
    SELECT name_, type_ INTO resource_name FROM resource WHERE id_ =
resource_id;
    RETURN resource_name;
END;
$$ LANGUAGE plpgsql;

select get_resource_name(7);
```

	get_resource_name
	character varying
1	transport7

1.f. Процедура оновлює звання військовослужбовця залежно від переданого ID та нового звання:

```
CREATE OR REPLACE PROCEDURE update_serviceman_rank(serviceman_id
INT, new_rank VARCHAR)
AS $$
BEGIN
    UPDATE serviceman SET rank_ = new_rank WHERE id_ = serviceman_id;
END;
$$ LANGUAGE plpgsql;
```

```
call update_serviceman_rank(2, 'president');
```

```
select * from serviceman;
```

21	22	Anas.	2003-01-01	private	22	[null]
22	23	FictBot	2000-01-01	flagman	23	[null]
23	2	Denys	2007-01-01	president	2	[null]

1.g. Процедура повертає перелік ресурсів заданого типу:

```
CREATE OR REPLACE FUNCTION list_resources_by_type(resource_type
VARCHAR)
RETURNS TABLE(id INT, name VARCHAR, type VARCHAR) AS $$
BEGIN
    RETURN QUERY SELECT id_, name_, type_ FROM resource WHERE type_ =
resource_type;
END;
$$ LANGUAGE plpgsql;
```

```
select list_resources_by_type('transport');
```

	list_resources_by_type record
1	1,transport1,transport
2	2,transport2,transport
3	3,transport3,transport
4	4,transport4,transport
5	5,transport5,transport
6	6,transport6,transport
7	7,transport7,transport

2.a. Процедура для отримання кількості військових у конкретному підрозділі:

```
DROP FUNCTION get_serviceman_count_by_unit;
```

```
CREATE OR REPLACE FUNCTION get_serviceman_count_by_unit(x_unit_id
INT, x_unit_type unit_type)
RETURNS INT AS $$
DECLARE
    count_servicemen INT;
BEGIN
    SELECT COUNT(*)
    INTO count_servicemen
    FROM serviceman_affiliation sa
    WHERE sa.unit_id = x_unit_id AND sa.unit_type = x_unit_type;

    RETURN count_servicemen;
END;
$$ LANGUAGE plpgsql;
```

```
SELECT get_serviceman_count_by_unit(1, 'military_unit');
```

	get_serviceman_count_by_unit integer
1	8


2.b. Отримання різних стовбців з таблиці з військовозобов'язаними:

```
CREATE OR REPLACE FUNCTION get_dynamic_table(columns_list TEXT)
RETURNS TABLE(result JSONB) AS $$
BEGIN
```



```
-- Створюємо динамічний SQL-запит для вибірки
RETURN QUERY EXECUTE
'SELECT to_jsonb(t) FROM (SELECT ' || columns_list || ' FROM serviceman) t';
END;
$$ LANGUAGE plpgsql;

select * from get_dynamic_table('id_', fio');
```

	result jsonb 
1	{"fio": "Arsen", "id_": 1}
2	{"fio": "Loleps", "id_": 3}
3	{"fio": "Mia", "id_": 4}
4	{"fio": "Piata", "id_": 5}
5	{"fio": "Danya", "id_": 6}
6	{"fio": "Popov", "id_": 7}
7	{"fio": "Vadim", "id_": 8}

2.с. Повертає таблицю з іменами військовослужбовців та назвами підрозділів, до яких вони належать:

```
CREATE OR REPLACE FUNCTION get_serviceman_and_unit()
RETURNS TABLE(serviceman_name VARCHAR, unit_name VARCHAR) AS $$
BEGIN
    -- Повертаємо дані про військовослужбовців та їх підрозділи
    RETURN QUERY
    SELECT s.fio AS serviceman_name, mu.name_ AS unit_name
    FROM serviceman s
    LEFT JOIN serviceman_affiliation sa ON sa.serviceman_id = s.id_
    LEFT JOIN military_unit mu ON sa.unit_id = mu.id_;
END;
$$ LANGUAGE plpgsql;

select * from get_serviceman_and_unit();
```

	serviceman_name character varying 🔒	unit_name character varying 🔒
1	Arsen	Vera
2	Denys	Bratya
3	Loleps	Mayonsk
4	Mia	Vera
5	Piata	Bratya
6	Danya	Mayonsk
7	Popov	Vera

3. Пройтися та вивести всіх командирів:

```

CREATE OR REPLACE PROCEDURE demo_cursor()
LANGUAGE plpgsql AS
$$
DECLARE
    serviceman_cursor CURSOR FOR
        SELECT id_, fio
            FROM serviceman s
            JOIN serviceman_affiliation sa ON s.ID_ = sa.serviceman_id
            WHERE sa.is_commander = true;

    v_id INT;
    v_fio VARCHAR(64);
BEGIN
    OPEN serviceman_cursor;

    LOOP
        FETCH serviceman_cursor INTO v_id, v_fio;

        EXIT WHEN NOT FOUND;
        RAISE NOTICE 'ID: %, FIO: %', v_id, v_fio;
    END LOOP;

    CLOSE serviceman_cursor;
END;
$$;

CALL demo_cursor();

```

```
ЗАМЕЧАНИЕ: ID: 1, FIO: Arsen
ЗАМЕЧАНИЕ: ID: 3, FIO: Loleps
ЗАМЕЧАНИЕ: ID: 2, FIO: Denys
CALL

Query returned successfully in 51 msec.
```

а. Призначення тригера: безкоштовна акція по отриманню звання сержанта при самовільному пониженні до звання рядового (увага, акція триває лише до 4 курсу після успішного захисту диплому):

```
CREATE table update_log(
    serviceman_id int,
    old_fio text,
    new_fio text,
    old_rank text,
    new_rank text,
    updated_at timestamp,
    last_modified timestamp
);
```

```
CREATE OR REPLACE FUNCTION after_update_serviceman()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO update_log (serviceman_id, old_fio, new_fio, old_rank, new_rank,
updated_at)
        VALUES (OLD.id_, OLD.fio, NEW.fio, OLD.rank_, NEW.rank_,
CURRENT_TIMESTAMP);

    IF NEW.rank_ = 'Private' THEN
        NEW.rank_ := 'Sergeant';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER trigger_after_update
BEFORE UPDATE ON serviceman
FOR EACH ROW
EXECUTE FUNCTION after_update_serviceman();
```

	serviceman_id integer	old_fio text	new_fio text	old_rank text	new_rank text	updated_at timestamp without time zone	last_time
1	5	Piata	Piata	officer	Private	2024-12-27 03:22:06.907736	[null]

23	5	Piata		2004-01-01	Sergeant		5 [null]
----	---	-------	--	------------	----------	--	----------

с. Логувати усіх військовозобов'язаних, кого було додано до таблиці військових:

```
CREATE OR REPLACE FUNCTION log_serviceman_insertion()
RETURNS TRIGGER AS $$
BEGIN
    -- Insert the id_ and insertion timestamp into the log table
    INSERT INTO serviceman_insertion_log (serviceman_id, inserted_at)
    VALUES (NEW.id_, CURRENT_TIMESTAMP);
    RETURN NEW; -- Return the NEW record (the inserted serviceman)
END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER trigger_log_serviceman_insertion
AFTER INSERT ON serviceman
FOR EACH ROW
EXECUTE FUNCTION log_serviceman_insertion();
```

```
insert into serviceman(fio, birth_date, rank_)
values ('Vecherkovka Anastasia', '1977-01-01', 'officer');
```

	serviceman_id integer	Inserted_at timestamp without time zone
1	24	2024-12-27 03:27:49.903247

с. Логувати усіх військовозобов'язаних, кого було видалено з таблиці військових. В даному випадку процедура була використана для каскадного знищення усіх висячих запитів з залежних таблиць:

```
CREATE TABLE serviceman_deletion_log (
    serviceman_id INT,
    deleted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE OR REPLACE FUNCTION log_serviceman_deletion()
RETURNS TRIGGER AS $$
BEGIN
    -- Insert the id_ and deletion timestamp into the log table
```

```

INSERT INTO serviceman_deletion_log (serviceman_id, deleted_at)
VALUES (OLD.id_, CURRENT_TIMESTAMP);
RETURN OLD; -- Return the OLD record (the deleted serviceman)
END;
$$ LANGUAGE plpgsql;

```

```

CREATE OR REPLACE TRIGGER trigger_log_serviceman_deletion
AFTER DELETE ON serviceman
FOR EACH ROW
EXECUTE FUNCTION log_serviceman_deletion();

```

```

create or replace procedure delete_serviceman(x_id_ int)
AS $$
declare
    rank_id int := (select rank_data_id from serviceman s where s.id_ = x_id_);
BEGIN
    if (select count(*) from serviceman where id_ = x_id_) = 0 then
        return;
    end if;

    delete from serviceman_speciality ss where ss.serviceman_id = x_id_;
    delete from serviceman s where s.id_ = x_id_;
    delete from rank_data rd where rd.id_ = rank_id;
END;
$$ LANGUAGE plpgsql;

```

```
delete from serviceman where birth_date = '1977-01-01';
```

```
call delete_serviceman(23);
```

```
select * from serviceman_deletion_log;
```

```
select * from serviceman;
```

	serviceman_id integer	deleted_at timestamp without time zone
1	24	2024-12-27 03:30:38.774513

	serviceman_id integer	deleted_at timestamp without time zone
1	24	2024-12-27 03:30:38.774513
2	23	2024-12-27 03:46:30.043287
3	22	2024-12-27 03:46:49.179233
4	21	2024-12-27 03:46:51.729732

id	rank	name	birth_date	speciality	
16	18	Kolya	2003-01-01	private	
17	19	Sasha	2006-01-01	private	
18	20	Derii	2005-01-01	flagman	
19	2	Denys	2007-01-01	president	
20	5	Piata	2004-01-01	Sergeant	

Висновок

У ході виконання лабораторної роботи було вивчені такі нові поняття, як функції (FUNCTION), процедури (PROCEDURE), різноманітні за суттю тригери (TRIGGER), курсори (CURSOR) та конструкції для контролювання ходу виконання команд (IF, WHILE, FETCH) та запитів SQL.

Було розглянуто створення нових функцій, процедур з та без аргументів, з та без повернення значень з них, декларування нових змінних різних типів, цикли та умови. Було приділено увагу такій річчі, як динамічне виконання коду за допомогою QUERY EXECUTE та оператора конкатування «||» для отримання валідного DML запиту для отримання даних з вже існуючих таблиць та представлень. Було прооперовано з тимчасовими таблицями.

Було розглянуто, що таке курсор, як його відкривати, ітерувати, та, нарешті, закривати. Вони є гнучким інструментом для покращення якості та зрозумілості коду в правильних руках.

Також було розглянуто призначення тригерів та їхня суть в коді і їх важливість при обробці бази даних. Вони можуть слугувати інструментом для забезпечення бізнес-правил, або для виконання дій на якісь події, наприклад вставку та видалення запитів з таблиць.