



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

Звіт
з дисципліни «Бази даних»
Лабораторна робота №5
"Основи програмування з використанням мови SQL. Збережені процедури.
Курсори. Створення, програмування та керування тригерами."

Виконав:

Студент II курсу
гр. ІІІ-33
Соколов О. В.

Перевірила:

Марченко О. І.

Лабораторна робота № 5.

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:

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:

- a. створити тригер, котрий буде спрацьовувати при видаленні даних;
- b. створити тригер, котрий буде спрацьовувати при модифікації даних;
- c. 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.

Програмне забезпечення автопідприємства. Автопідприємство міста займається організацією пасажирських і вантажних перевезень всередині міста. У віданні підприємства знаходиться автотранспорт різного призначення: автобуси, таксі, маршрутні таксі, інший легковий транспорт, вантажний транспорт, транспорт допоміжного характеру, представлений різними марками. Кожна з перерахованих категорій транспорту має характеристики, властиві тільки цій категорії: наприклад, до характеристик вантажного транспорту відноситься вантажопідйомність, пасажирський транспорт характеризується місткістю і т.д. З плином часу, з одного боку, транспорт старіє і списується (можливо, продається), а з іншого, підприємство поповнюється новим автотранспортом. Підприємство має штат водіїв, закріплених за автомобілями (за одним автомобілем може бути закріплено більше одного водія). Водії об'єднуються в бригади, якими керують бригадири. Пасажирський автотранспорт (автобуси, маршрутні таксі) перевозить пасажирів за визначеними маршрутами, за кожним з них закріплені окремі одиниці автотранспорту. Ведеться облік числа перевезених пасажирів, на підставі чого проводиться перерозподіл транспорту з одного маршруту на інший.

Основні множини сутностей

Vehicle (Транспортний засіб):

- **id:** Унікальний ідентифікатор транспортного засобу.
- **registration_number:** Номер реєстрації транспортного засобу.
- **model:** Модель транспортного засобу.
- **brand:** Бренд транспортного засобу.
- **year_of_manufacture:** Рік виготовлення транспортного засобу.
- **status:** Статус транспортного засобу (наприклад, активний, списаний, проданий).
- **vehicle_type_id:** Зовнішній ключ, що посилається на тип транспортного засобу (VehicleType).
- **capacity:** (Опціонально, залежно від типу транспортного засобу) Місткість для пасажирських транспортних засобів.
- **load_capacity:** (Опціонально, залежно від типу транспортного засобу) Вантажопідйомність для вантажних транспортних засобів.
- **team_id:** Зовнішній ключ, що посилається на команду (Team).

VehicleType (Тип транспортного засобу):

- **id:** Унікальний ідентифікатор типу транспортного засобу.
- **name:** Назва типу транспортного засобу (наприклад, автобус, таксі, вантажівка тощо).
- **description:** Короткий опис типу транспортного засобу.

Driver (Водій):

- **id:** Унікальний ідентифікатор водія.
- **name:** Повне ім'я водія.
- **license_number:** Номер водійських прав водія.
- **employment_date:** Дата прийняття водія на роботу.
- **team_id:** Зовнішній ключ, що посилається на команду (Team).

Team (Команда):

- **id:** Унікальний ідентифікатор команди.
- **name:** Назва команди.
- **foreman_id:** Зовнішній ключ, що посилається на водія (Driver).

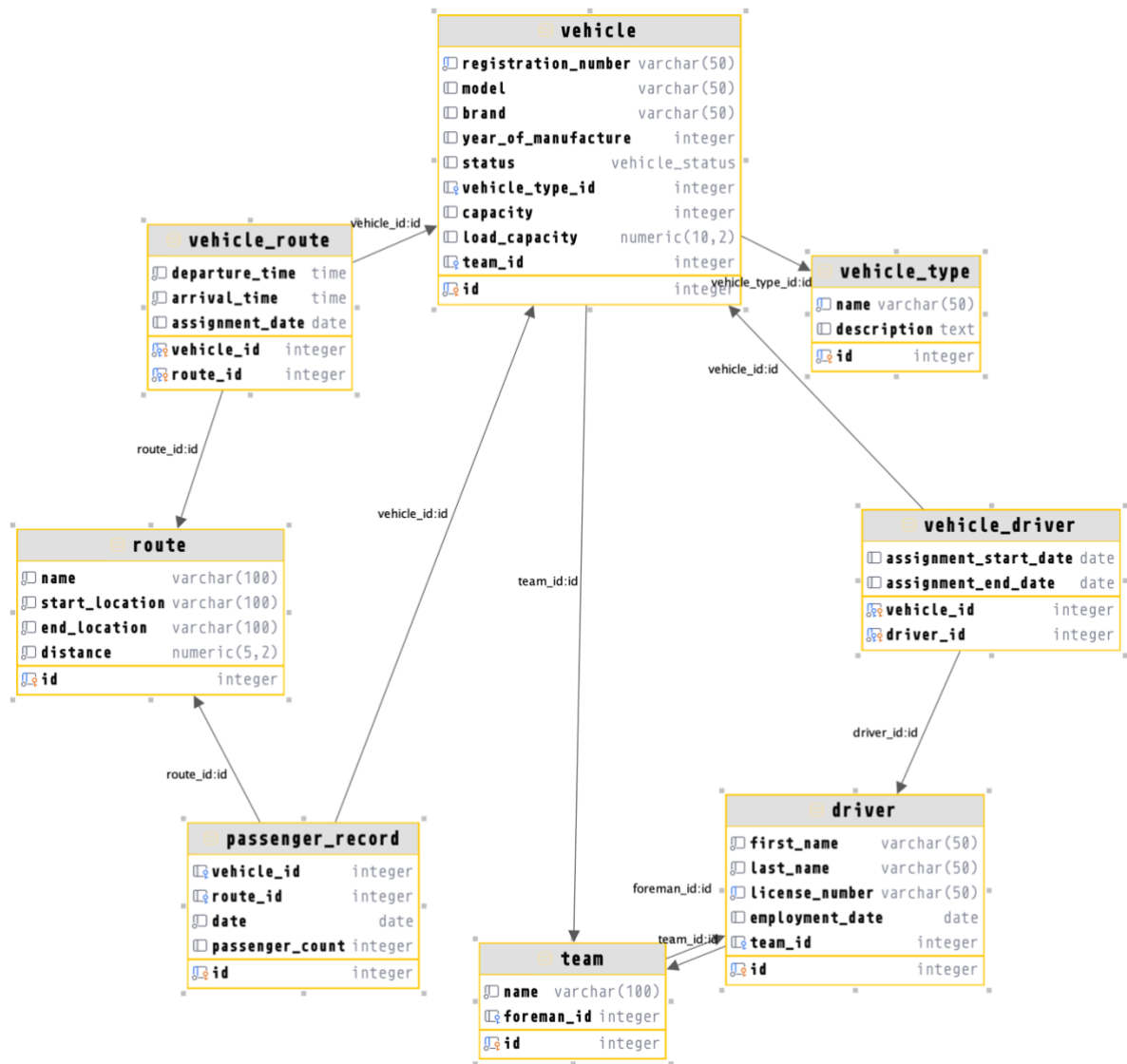
Route (Маршрут):

- **id:** Унікальний ідентифікатор маршруту.
- **name:** Назва або номер маршруту.
- **start_location:** Початкове місце маршруту.
- **end_location:** Кінцеве місце маршруту.
- **distance:** Відстань маршруту в кілометрах.
- **assignment_date:** Дата призначення на маршрут

PassengerRecord (Запис про пасажирів):

- **id:** Унікальний ідентифікатор запису про пасажирів.
- **vehicle_id:** Зовнішній ключ, що посилається на транспортний засіб (Vehicle).
- **route_id:** Зовнішній ключ, що посилається на маршрут (Route).
- **date:** Дата запису.
- **passenger_count:** Кількість перевезених пасажирів.

ER-Модель



SQL Скрипты

```

1  -- а. створення процедури, в якій використовується тимчасова таблиця,
2  -- котра створена через змінну типу TABLE;
3
4  CREATE OR REPLACE PROCEDURE get_vehicles_for_route(route_id_param INT)
5  LANGUAGE plpgsql
6  AS
7  $$
8  DECLARE
9      record RECORD;
10 BEGIN
11     CREATE TEMP TABLE temp_vehicles
12     (
13         vehicle_id INT,
14         model VARCHAR(50),
15         capacity INT
16     );
17
18     INSERT INTO temp_vehicles
19     SELECT vehicle_id vehicle.id, vehicle.model, vehicle.capacity
20     FROM vehicle
21     WHERE vehicle.id IN (SELECT vehicle_route.vehicle_id
22                         FROM vehicle_route
23                         WHERE vehicle_route.route_id = route_id_param);
24 END;
25 $$;
26

```

temp_vehicles 2

vehicle_id	model	capacity
1	416 Miata MX-5	5
2	887 A6	13
3	949 Discovery	16

```

42 CREATE OR REPLACE PROCEDURE update_vehicle_status(vehicle_id INT, new_status vehicle_status)
43 LANGUAGE plpgsql
44 AS $$
45 BEGIN
46     IF new_status = 'decommissioned' THEN
47         UPDATE vehicle
48         SET status = new_status, team_id = NULL
49         WHERE id = vehicle_id;
50     ELSE
51         UPDATE vehicle
52         SET status = new_status
53         WHERE id = vehicle_id;
54     END IF;
55 END;
56 $$;
57
58 CALL update_vehicle_status( vehicle_id 1, new_status 'decommissioned');
59 -- Перевірка:
60 SELECT * FROM vehicle WHERE id = 1;
61

```

temp_vehicles 2 | Перевірка:

id	registration_number	model	brand	year_of_manufacture	status	vehicle_type_id
1	1 UT5825XJ	RX-8	Mazda	2004	decommissioned	3

```

61 -- б. створення процедури з використанням умовної конструкції IF;
62 CREATE OR REPLACE PROCEDURE update_vehicle_status(vehicle_id INT, new_status vehicle_status)
63 LANGUAGE plpgsql
64 AS $$
65 BEGIN
66     IF new_status = 'decommissioned' THEN
67         UPDATE vehicle
68         SET status = new_status, team_id = NULL
69         WHERE id = vehicle_id;
70     ELSE
71         UPDATE vehicle
72         SET status = new_status
73         WHERE id = vehicle_id;
74     END IF;
75 END;
76 $$;
77
78 CALL update_vehicle_status( vehicle_id 1, new_status 'decommissioned');
79
80 SELECT * FROM vehicle WHERE id = 1;
81

```

temp_vehicles 2 | Перевірка:

id	registration_number	model	brand	year_of_manufacture	status
1	1 UT5825XJ	RX-8	Mazda	2004	decommissioned

67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84

— c. створення процедури з використанням циклу WHILE;

CREATE OR REPLACE PROCEDURE deactivate_old_vehicles(cutoff_year INT)

LANGUAGE plpgsql

AS \$\$

DECLARE

current_vehicle RECORD;

BEGIN

FOR current_vehicle IN

SELECT id FROM vehicle WHERE year_of_manufacture < cutoff_year

LOOP

UPDATE vehicle

SET status = 'decommissioned'

WHERE id = current_vehicle.id;

RAISE NOTICE 'Vehicle ID % has been decommissioned', current_vehicle.id;

END LOOP;

END;

\$\$;

2024-12-23 23:13:37

completed in 4 ms

[lab3.public> CALL deactivate_old_vehicles(1971)

Vehicle ID 15 has been decommissioned

Vehicle ID 97 has been decommissioned

Vehicle ID 153 has been decommissioned

Vehicle ID 160 has been decommissioned

Vehicle ID 645 has been decommissioned

Vehicle ID 68 has been decommissioned

Vehicle ID 70 has been decommissioned

Vehicle ID 148 has been decommissioned

Vehicle ID 169 has been decommissioned

Vehicle ID 248 has been decommissioned

Vehicle ID 279 has been decommissioned

Vehicle ID 316 has been decommissioned

Vehicle ID 560 has been decommissioned

Vehicle ID 585 has been decommissioned

temp_vehicles 2 | Перегляд: | Перегляд: 2

temp_vehicles 2 | Перегляд: | Перегляд: 2

279 rows

status registration_number model

1 248 decommissioned SU7188TA 1955 Corvette

2 585 decommissioned KA2914ZZ 1956 Corvette

3 560 decommissioned HN76170G 1961 Tempest

4 169 decommissioned II5713UQ 1965 Thunderbird

5 717 decommissioned VY4395KZ 1965 GTO

6 68 decommissioned JH6181XZ 1966 Toronado

7 70 decommissioned HQ0198PT 1966 Interceptor

8 905 decommissioned OK1566OW 1966 GTO

9 279 decommissioned JP9180VS 1967 Cougar

10 316 decommissioned SG9502JL 1967 Camaro

11 15 decommissioned HH4136LM 1967 Camaro

12 148 decommissioned UJ6985AF 1968 LeMans

13 645 decommissioned EE0544AC 1970 Mustang

14 97 decommissioned HV8280ZX 1970 Camaro

15 153 decommissioned JG5284BZ 1970 Grand Prix

16 160 decommissioned SL5105FP 1970 914

17 756 decommissioned TS6260EI 1973 Grand Prix

18 799 decommissioned KV3169ID 1974 Mustang

19 486 decommissioned RW2061OR 1974 Camaro

20 122 decommissioned TA5582KL 1979 LUV

21 161 decommissioned WE5518RK 1983 Sunbird

22 323 decommissioned SC0171SR 1984 1000

89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117

— d. створення процедури без параметрів;

CREATE OR REPLACE PROCEDURE clear_old_routes()

LANGUAGE plpgsql

AS \$\$

BEGIN

DELETE FROM route

WHERE id NOT IN (SELECT DISTINCT vehicle_route.route_id FROM vehicle_route);

END;

\$\$;

CALL clear_old_routes();

SELECT * FROM route WHERE id NOT IN (SELECT DISTINCT vehicle_route.route_id FROM vehicle_route);

INSERT INTO route (name, start_location, end_location, distance)

VALUES (name 'Old Route', start_location 'Point A', end_location 'Point B', distance 15.0);

Перегляд: 5

281 rows

id name start_location end_location distance

1 7 nullian portitor l. 1 Comanche Point 8 Cambridge Place 95.98

2 8 curabitur convallis. 625 Helena Road 456 Elmside Parkway 57.86

3 18 odio consequat var. 57 Manley Pass 4302 Thackeray Drive 36.73

4 17 id consequat in co. 616 Juna Road 05600 Anderson Trail 7.21

5 18 lacinia aenean sit. 6534 Old Gate W. 39804 David Parkway 71.45

6 20 magnis dis partur. 82273 Grim Circ. 0525 Susan Alley 68.15

7 21 eget rutrum at lor. 5 Lakeland Hill 67 Aberg Park 73.89

8 27 mi in portitor pe. 35351 Mifflin P. 6 Bellgrove Road 98.61

9 31 ut mauris eget mas. 7 Del Mar Plaza 7 Schmedeman Place 13.55

10 44 sit amet lobortis l. 8200 Heath Park 6 Prairie Rose Point 42.96

Перегляд: 5

0 rows

id name start_location end_location


```

112 -- е. створення процедури з вхідним параметром та RETURN;
113 CREATE OR REPLACE FUNCTION calculate_total_passengers(route_id_param INT)
114 RETURNS INT
115 LANGUAGE plpgsql
116 AS $$
117 DECLARE
118     total_passengers INT;
119 BEGIN
120     SELECT SUM(passenger_record.passenger_count)
121     INTO total_passengers
122     FROM passenger_record
123     WHERE passenger_record.route_id = route_id_param;
124
125     RETURN total_passengers;
126 END;
127 $$;
128
129 ✓ SELECT calculate_total_passengers( route_id_param 1);
130
131 INSERT INTO passenger_record (vehicle_id, route_id, date, passenger_count)
132 VALUES ( vehicle_id 1, route_id 1, date CURRENT_DATE, passenger_count 50);
133
134

```

3 lab3.public.vehicle 4 Перебірка: Перебірка: 2 Перебірка: 3

1 row

calculate_total_passengers

1 37

```

141 CREATE OR REPLACE PROCEDURE update_driver_team(driver_id_param INT, new_team_id_param INT)
142 LANGUAGE plpgsql
143 AS
144 $$
145 BEGIN
146     UPDATE driver
147     SET team_id = new_team_id_param
148     WHERE id = driver_id_param;
149 END;
150 $$;
151
152 CALL update_driver_team( driver_id_param 1, new_team_id_param 2);
153
154 ✓ SELECT * FROM driver WHERE id = 1;
155
156
157
158

```

lab3.public.driver						
id	team_id	first_name	last_name	license_number	employment_date	
1	1	861 Anabel	Pepin	EAFNU7247262	2003-04-13	

lab3.public.driver						
id	team_id	first_name	last_name	license_number	employment_date	
1	1	2 Anabel	Pepin	EAFNU7247262	2003-04-13	

```

162 CREATE OR REPLACE PROCEDURE get_route_vehicles(route_id_param INT)
163     LANGUAGE plpgsql
164 AS
165 $$
166 DECLARE
167     record RECORD;
168 BEGIN
169
170     RAISE NOTICE 'Starting get_route_vehicles for route_id: %', route_id_param;
171
172
173     FOR record IN
174         SELECT vehicle.id, vehicle.model, vehicle.capacity, vehicle.status
175         FROM vehicle
176             INNER JOIN vehicle_route 1<->1..n: ON vehicle.id = vehicle_route.vehicle_id
177         WHERE vehicle_route.route_id = route_id_param
178     LOOP
179
180         RAISE NOTICE 'Vehicle ID: %, Model: %, Capacity: %, Status: %',
181             record.id, record.model, record.capacity, record.status;
182     END LOOP;
183
184
185     RAISE NOTICE 'get_route_vehicles completed for route_id: %.', route_id_param;
186 END;
187 $$;
188
189 ✓ CALL get_route_vehicles( route_id_param 1);
190

```

```

[2024-12-23 23:28:12] completed in 5 ms
lab3.public> CALL get_route_vehicles(1)
Starting get_route_vehicles for route_id: 1
Vehicle ID: 40, Model: Passport, Capacity: 18, Status: decommissioned
get_route_vehicles completed for route_id: 1.
[2024-12-23 23:28:24] completed in 7 ms

```

```

-- а. Створення функції, яка повертає скалярне значення
CREATE OR REPLACE FUNCTION get_available_vehicles_count()
RETURNS INT
LANGUAGE plpgsql
AS $$
DECLARE
    available_count INT;
BEGIN
    SELECT COUNT(*) INTO available_count
    FROM vehicle
    WHERE status = 'available';

    RETURN available_count;
END;
$$;

SELECT get_available_vehicles_count();

```

functions.sql - functions.sql

get_available_vehicles_count():inte	
get_available_vehicles_count	▽
1	230

```
20 -- b. Створення функції, яка повертає таблицю з динамічним набором стовпців
21 CREATE OR REPLACE FUNCTION get_vehicle_details_json(route_id_param INT)
22 RETURNS JSONB
23 LANGUAGE plpgsql
24 AS $$
25 DECLARE
26     result JSONB;
27 BEGIN
28     SELECT jsonb_agg(
29         jsonb_build_object(
30             'vehicle_id', vehicle.id,
31             'model', vehicle.model,
32             'capacity', vehicle.capacity,
33             'status', vehicle.status
34         )
35     )
36     INTO result
37     FROM vehicle
38     INNER JOIN vehicle_route 1<->1..n: ON vehicle.id = vehicle_route.vehicle_id
39     WHERE vehicle_route.route_id = route_id_param;
40
41     RETURN result;
42 END;
43 $$;
44
45 SELECT get_vehicle_details_json(route_id_param 4);
46
47
48 -- c. Створення функції, яка повертає таблицю наперед заданої структури
49 CREATE OR REPLACE FUNCTION get_vehicle_details_for_route(route_id_param INT)
50 RETURNS TABLE (
51     vehicle_id INT,
52     model VARCHAR,
53     capacity INT,
54     status VARCHAR
55 )
56 LANGUAGE plpgsql
57 AS $$
58 BEGIN
59     RETURN QUERY
60     SELECT vehicle.id, vehicle.model, vehicle.capacity, vehicle.status
61     FROM vehicle
62     INNER JOIN vehicle_route 1<->1..n: ON vehicle.id = vehicle_route.vehicle_id
63     WHERE vehicle_route.route_id = route_id_param;
64 END;
65 $$;
66
67
68 SELECT * FROM get_vehicle_details_for_route(route_id_param 4);
69
```

get_vehicle_details_json(4):jsonb x

1	[{"model": "Mitsubishi", "status": "in_service", "capacity": 5, "vehicle_id": 416}, {"model": "Discovery", "status": "under_maintenance", "capacity": 16, "vehicle_id": 949}, {"model": "A6", "status": "in_service", "capacity": 13, "vehicle_id": 887}]
---	---

Result 8 x

vehicle_id	model	capacity	status
1	416 Mitsubishi	5	in_service
2	949 Discovery	16	under_maintenance
3	887 A6	13	in_service

```

1 CREATE OR REPLACE PROCEDURE get_vehicle_details_with_cursor(route_id_param INT)
2 LANGUAGE plpgsql
3 AS $$
4 DECLARE
5     vehicle_cursor CURSOR FOR
6         SELECT vehicle.id, vehicle.model, vehicle.capacity, vehicle.status
7         FROM vehicle
8         INNER JOIN vehicle_route 1<->1..n: ON vehicle.id = vehicle_route.vehicle_id
9         WHERE vehicle_route.route_id = route_id_param;
10
11     vehicle_id INT;
12     model VARCHAR(50);
13     capacity INT;
14     status vehicle_status;
15 BEGIN
16     OPEN vehicle_cursor;
17
18     LOOP
19         FETCH vehicle_cursor INTO vehicle_id, model, capacity, status;
20
21         EXIT WHEN NOT FOUND;
22
23         RAISE NOTICE 'Vehicle ID: %, Model: %, Capacity: %, Status: %',
24             vehicle_id, model, capacity, status;
25     END LOOP;
26
27     CLOSE vehicle_cursor;
28 END;
29 $$;

```

Services

lab3.public> CALL get_vehicle_details_with_cursor(4)
 Vehicle ID: 416, Model: Miata MX-5, Capacity: 5, Status: in_service
 Vehicle ID: 949, Model: Discovery, Capacity: 16, Status: under_maintenance
 Vehicle ID: 887, Model: A6, Capacity: 13, Status: in_service
 [2024-12-23 23:37:29] completed in 5 ms

BD > lab5 > code > cursors.sql

а. Триггер на удаление данных

```

CREATE OR REPLACE FUNCTION log_vehicle_deletion()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO vehicle_operations_log (operation, vehicle_id, model, operation_date)
    VALUES (operation 'DELETE', vehicle_id OLD.id, OLD.model, operation_date CURRENT_TIMESTAMP);

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER vehicle_delete_trigger
AFTER DELETE ON vehicle
FOR EACH ROW
EXECUTE FUNCTION log_vehicle_deletion();

```

```

INSERT INTO vehicle (id, registration_number, model, brand, year_of_manufacture, status, vehicle_type_id, capacity, load_capacity, team_id)
VALUES (id 33333, registration_number 'AB123CD', model 'Test Model', brand 'Test Brand', year_of_manufacture 2020, status 'available', vehicle_type_id 1, capacity 50, load_capacity 1000, team_id 1);

DELETE FROM vehicle WHERE id = 33333;

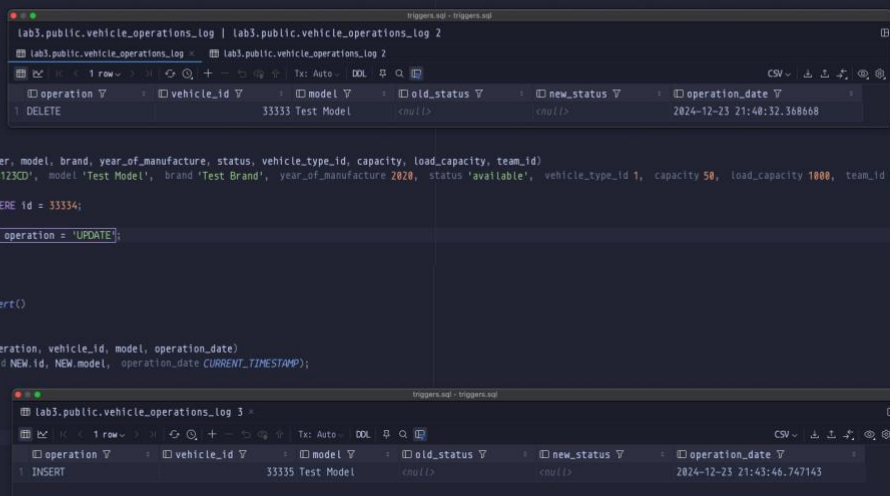
SELECT * FROM vehicle_operations_log WHERE operation = 'DELETE';

```

triggers.sql - triggers.sql

operation	vehicle_id	model	old_status	new_status	operation_date
DELETE	33333	Test Model	null	null	2024-12-23 21:40:32.368668

```
25
26 -- b. Тригер на модифікації даних
27
28 CREATE OR REPLACE FUNCTION log_vehicle_update()
29 RETURNS TRIGGER AS $$
30 BEGIN
31     INSERT INTO vehicle_operations_log (operation, vehicle_id, old_status, new_status, operation_date)
32     VALUES (operation 'UPDATE', vehicle_id OLD.id, old_status OLD.status, new_status NEW.status, operation_date CURRENT_TIMESTAMP);
33
34     RETURN NEW;
35 END;
36 $$ LANGUAGE plpgsql;
37
38 CREATE TRIGGER vehicle_update_trigger
39 AFTER UPDATE ON vehicle
40 FOR EACH ROW
41 EXECUTE FUNCTION log_vehicle_update();
42
43
44 INSERT INTO vehicle (id, registration_number, model, brand, year_of_manufacture, status, vehicle_type_id, capacity, load_capacity, team_id)
45 VALUES (id 33334, registration_number 'AB123CD', model 'Test Model', brand 'Test Brand', year_of_manufacture 2020, status 'available', vehicle_type_id 1, capacity 50, load_capacity 1000, team_id 1);
46
47 UPDATE vehicle SET status = 'available' WHERE id = 33334;
48
49 SELECT * FROM vehicle_operations_log WHERE operation = 'UPDATE';
50
51
52 -- c. Тригер на додавання даних
53
54 CREATE OR REPLACE FUNCTION log_vehicle_insert()
55 RETURNS TRIGGER AS $$
56 BEGIN
57     INSERT INTO vehicle_operations_log (operation, vehicle_id, model, operation_date)
58     VALUES (operation 'INSERT', vehicle_id NEW.id, NEW.model, operation_date CURRENT_TIMESTAMP);
59
60     RETURN NEW;
61 END;
62 $$ LANGUAGE plpgsql;
63
64 CREATE TRIGGER vehicle_insert_trigger
65 AFTER INSERT ON vehicle
66 FOR EACH ROW
67 EXECUTE FUNCTION log_vehicle_insert();
68
69 INSERT INTO vehicle (id, registration_number, model, brand, year_of_manufacture, status, vehicle_type_id, capacity, load_capacity, team_id)
70 VALUES (id 33335, registration_number 'AB1645CD', model 'Test Model', brand 'Test Brand', year_of_manufacture 2020, status 'available', vehicle_type_id 1, capacity 50, load_capacity 1000, team_id 1);
71
72 SELECT * FROM vehicle_operations_log WHERE operation = 'INSERT';
73
```



operation	vehicle_id	model	old_status	new_status	operation_date
DELETE	33333	Test Model	null()	null()	2024-12-23 21:48:32.368668

operation	vehicle_id	model	old_status	new_status	operation_date
INSERT	33335	Test Model	null()	null()	2024-12-23 21:43:46.747143

Git-репозиторій додається: https://github.com/sokolovgit/database_course

Висновок: У ході роботи було реалізовано створення та використання збережених процедур, функцій, курсорів і тригерів для вирішення різних завдань. Було досліджено особливості синтаксису й семантики процедур та функцій, способи їх параметризації, визначення результатів і виконання. Також опрацьовано основні характеристики курсорів, зокрема їх типи, команди SQL для створення, отримання та зміни даних.