Лабораторна робота №4

З дисципліни: Бази даних та інформаційні системи
Студента групи МІТ-31: Якоба Р.В.

Тема: Розширення можливостей PostgreSQL: користувацькі типи, функції та тригери

Мета роботи: Закріпити знання з розширюваності PostgreSQL. Навчитися створювати користувацькі типи даних. Реалізувати власну користувацьку функцію або агрегат. Створити тригери для логування змін у базі даних. Автоматично оновлювати пов’язані таблиці чи заповнювати значення. Оновити діаграму бази даних відповідно до виконаних завдань. Перевірити коректність роботи реалізованих об’єктів через виконання тестових SQL-запитів.

Завдання:

1. Створення користувацького типу даних
   - У нашій базі даних ветеринарної клініки можна створити користувацький тип для статусу прийому (наприклад, appointment_status), який може мати значення: scheduled, completed, canceled.

```
-- Створення користувацького типу ENUM:
CREATE TYPE appointment_status AS ENUM ('scheduled', 'completed', 'canceled');

-- Додавання нового стовпця до таблиці Appointment:
ALTER TABLE Appointment ADD COLUMN status appointment_status DEFAULT 'scheduled';

--Перевірка цілісності даних:
SELECT * FROM Appointment;
```

Результат:

	appointmentid [PK] integer	petid integer	vetid integer	date date	diagnosis text	status appointment_status
1	2	2	2	2024-02-12	Травма лапи	scheduled
2	1	1	1	2024-02-10	Алергія	completed
3	10	3	1	2024-03-01	Перевірка	scheduled

2. Створення користувацької функції або агрегату

```
-- Створення функції для розрахунку середнього віку тварин:
CREATE OR REPLACE FUNCTION calculate_average_pet_age() RETURNS NUMERIC AS $$
BEGIN
    RETURN (SELECT AVG(Age) FROM Pet);
END;
$$ LANGUAGE plpgsql;

-- Використання функції у тестовому запиті:
SELECT calculate_average_pet_age() AS average_pet_age;
```

Результат:

	average_pet_age numeric
1	3.4000000000000000

3. Створення тригерів для логування змін та автоматичного оновлення пов’язаних таблиць

```
-- Створення таблиці для логування змін у таблиці Appointment:
CREATE TABLE appointment_log (
    log_id SERIAL PRIMARY KEY,
    appointment_id INT,
    operation CHAR(1),
    changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Створення тригерної функції для логування:
CREATE OR REPLACE FUNCTION log_appointment_changes() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO appointment_log (appointment_id, operation)
    VALUES (NEW.AppointmentID, SUBSTRING(TG_OP, 1, 1)); -- Записує перший символ ('I', 'U', 'D')
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Прив’язання тригера до таблиці Appointment:
CREATE TRIGGER track_appointment_changes
AFTER INSERT OR UPDATE OR DELETE ON Appointment
FOR EACH ROW
EXECUTE FUNCTION log_appointment_changes();
```

```
-- Додавання нового запису
INSERT INTO Appointment (PetID, VetID, Date, Diagnosis, status)
VALUES (3, 1, '2024-03-01', 'Перевірка', 'scheduled');

-- Оновлення запису
UPDATE Appointment SET status = 'completed' WHERE AppointmentID = 1;

-- Видалення запису
DELETE FROM Appointment WHERE AppointmentID
= 9;

SELECT * FROM appointment_log;
```

Результат:

	log_id [PK] integer	appointment_id integer	operation character (1)	changed_at timestamp without time zone
1	1	9	I	2025-03-13 19:25:47.083522
2	2	1	U	2025-03-13 19:25:59.573349
3	4	10	I	2025-03-13 19:26:56.469177

```sql
-- Додавання стовпця totalappointments
ALTER TABLE Veterinarian
ADD COLUMN totalappointments INT DEFAULT 0;

--  тригер для оновлення totalappointments
CREATE OR REPLACE FUNCTION update_vet_appointments()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        UPDATE Veterinarian
        SET totalappointments = totalappointments + 1
        WHERE VetID = NEW.VetID;
    ELSIF TG_OP = 'DELETE' THEN
        UPDATE Veterinarian
        SET totalappointments = totalappointments - 1
        WHERE VetID = OLD.VetID;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER track_vet_appointments
AFTER INSERT OR DELETE ON Appointment
FOR EACH ROW
EXECUTE FUNCTION update_vet_appointments();
```

```sql
-- Перевірка роботи тригера
INSERT INTO Appointment (PetID, VetID, Date, Diagnosis, status)
VALUES (1, 1, '2024-03-01', 'Тест', 'scheduled');

-- Після додавання прийому
INSERT INTO Appointment (...) VALUES (...);
SELECT TotalAppointments FROM Veterinarian WHERE VetID = 1;
```

Результат:

	vetid [PK] integer	name character varying (255)	specialization character varying (255)	contact character varying (255)	totalappointments integer
1	2	Олена Петрова	Ортопед	petrova@example.com	0
2	3	Марія Сидоренко	Дерматолог	sydorenko@example.co…	0
3	4	Ігор Василенко	Ортопед	vasylenko@example.com	0
4	5	Наталія Коваль	Терапевт	koval@example.com	0
5	6	Андрій Мельник	Хірург	melnyk@example.com	0
6	7	Софія Лисенко	Дерматолог	lysenko@example.com	0
7	1	Дмитро Іванов	Дерматолог	ivanov@example.com	1

4. Оновлення діаграми бази даних
   - Додано стовпець status у таблиці Appointment.
   - Додано таблицю appointment_log.
   - Встановлено зв’язок між Appointment і appointment_log.
   -
5. Перевірка роботи

```sql
--Перевірка цілісності даних:
SELECT * FROM Appointment;
```

	appointmentid [PK] integer	petid integer	vetid integer	date date	diagnosis text	status appointment_status
1	2	2	2	2024-02-12	Травма лапи	scheduled
2	1	1	1	2024-02-10	Алергія	completed
3	10	3	1	2024-03-01	Перевірка	scheduled
4	11	1	1	2024-03-01	Тест	scheduled

```
-- Використання функції у тестовому запиті:
SELECT calculate_average_pet_age() AS average_pet_age;
```

	average_pet_age numeric
1	3.4000000000000000

```
SELECT * FROM appointment_log;
```

	log_id [PK] integer	appointment_id integer	operation character (1)	changed_at timestamp without time zone
1	1	9	I	2025-03-13 19:25:47.083522
2	2	1	U	2025-03-13 19:25:59.573349
3	4	10	I	2025-03-13 19:26:56.469177
4	5	11	I	2025-03-13 19:45:16.421146

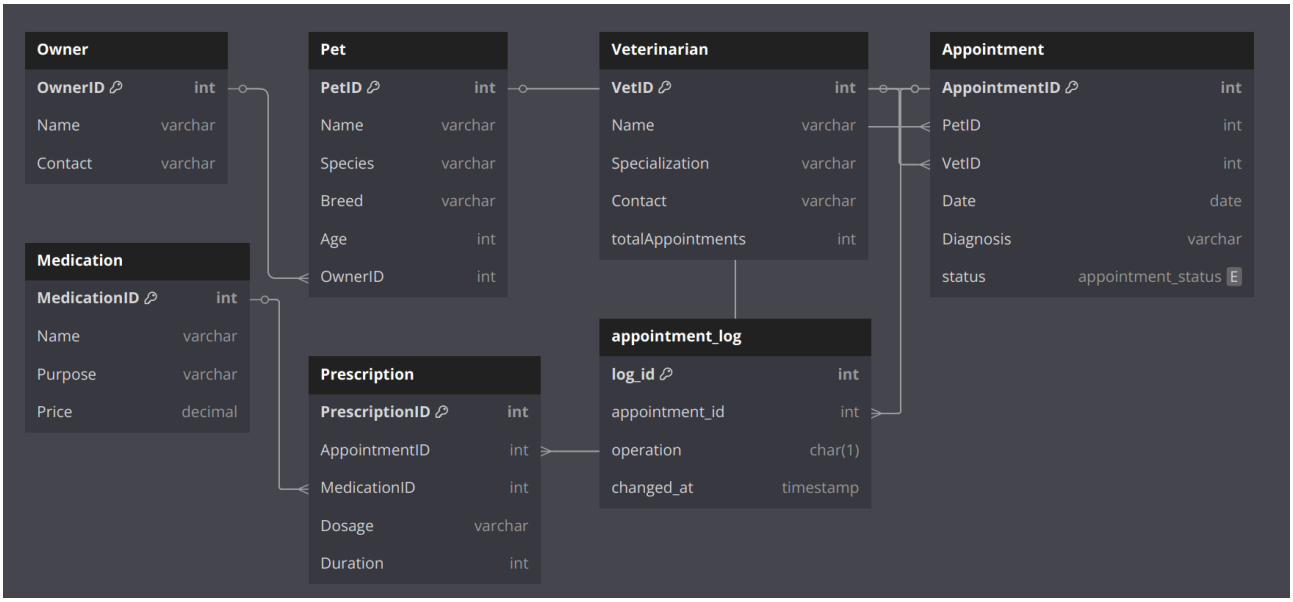
```
-- Перевірка роботи тригера
INSERT INTO Appointment (PetID, VetID, Date, Diagnosis, status)
VALUES (1, 1, '2024-03-01', 'Тест', 'scheduled');

-- Після додавання прийому
INSERT INTO Appointment (...) VALUES (...);
SELECT TotalAppointments FROM Veterinarian WHERE VetID = 1;

SELECT * FROM VETERINARIAN;
```

	vetid [PK] integer	name character varying (255)	specialization character varying (255)	contact character varying (255)	totalappointments integer
1	2	Олена Петрова	Ортопед	petrova@example.com	0
2	3	Марія Сидоренко	Дерматолог	sydorenko@example.co…	0
3	4	Ігор Василенко	Ортопед	vasylenko@example.com	0
4	5	Наталія Коваль	Терапевт	koval@example.com	0
5	6	Андрій Мельник	Хірург	melnyk@example.com	0
6	7	Софія Лисенко	Дерматолог	lysenko@example.com	0
7	1	Дмитро Іванов	Дерматолог	ivanov@example.com	1

- Оновлена ER-діаграма бази даних



Висновок:

Під час лабораторної роботи було розширено функціонал бази даних ветеринарної клініки за допомогою PostgreSQL. Створено користувацький тип appointment_status для контролю статусів прийомів, реалізовано функцію автоматичного розрахунку середнього віку тварин та тригери для оновлення лічильника прийомів ветеринарів і логування змін. Оновлено ER-діаграму, що відображає нові сутності та зв’язки. Ці зміни покращили автоматизацію, запобігли помилкам у даних і забезпечили зручність аналізу, демонструючи практичну цінність розширених можливостей СУБД.