

Київський національний університет ім. Тараса Шевченка

Кафедра мережевих та інтернет технологій

## Лабораторна робота №4

**Дисципліна:** Базы даних та інформаційні системи

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

**Мета:** 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.

Виконала студентка групи MIT-31

Пась Олександра

### Хід виконання

#### 1. Створення користувацького типу даних

```
-- 1. Спочатку створюємо новий тип ENUM
CREATE TYPE payouts_status_type AS ENUM ('pending', 'completed', 'failed');

-- 2. Видаляємо обмеження DEFAULT зі стовпця status
ALTER TABLE payouts ALTER COLUMN status DROP DEFAULT;

-- 3. Змінюємо тип стовпця з явним перетворенням значень
ALTER TABLE payouts
ALTER COLUMN status TYPE payouts_status_type
USING (
    CASE status
        WHEN 'pending' THEN 'pending'::payouts_status_type
        WHEN 'completed' THEN 'completed'::payouts_status_type
        WHEN 'failed' THEN 'failed'::payouts_status_type
        ELSE 'pending'::payouts_status_type -- значення за замовчуванням для інших випадків
    END
);

-- 4. Встановлюємо нове значення за замовчуванням
ALTER TABLE payouts
ALTER COLUMN status SET DEFAULT 'pending'::payouts_status_type;

SELECT * FROM payouts;
```

Результат:





	id [PK] integer	claim_id integer	amount numeric (12,2)	payout_date date	method character varying (20)	status payouts_status_type
1	1	1	15000.00	2023-05-28	bank_transfer	completed
2	2	2	8000.00	2023-06-22	bank_transfer	completed

## 2. Створення користувачької функції:

```
--Створення користувачької функції
CREATE OR REPLACE FUNCTION client_payouts_analysis(client_id_param INT)
RETURNS TABLE (
    total_amount DECIMAL(12,2),
    completed_amount DECIMAL(12,2),
    pending_amount DECIMAL(12,2),
    failed_amount DECIMAL(12,2)
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        COALESCE(SUM(po.amount), 0) AS total_amount,
        COALESCE(SUM(CASE WHEN po.status = 'completed' THEN po.amount ELSE 0 END), 0) AS completed_amount,
        COALESCE(SUM(CASE WHEN po.status = 'pending' THEN po.amount ELSE 0 END), 0) AS pending_amount,
        COALESCE(SUM(CASE WHEN po.status = 'failed' THEN po.amount ELSE 0 END), 0) AS failed_amount
    FROM payouts po
    JOIN claims cl ON po.claim_id = cl.id
    JOIN policies p ON cl.policy_id = p.id
    WHERE p.client_id = client_id_param;
END;
$$ LANGUAGE plpgsql;

-- Приклад використання функції
SELECT * FROM client_payouts_analysis(1);
```

Результат:

	total_amount 	completed_amount 	pending_amount 	failed_amount 
	numeric	numeric	numeric	numeric
1	15000.00	15000.00	0	0

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

```
-- Таблиця для логування змін у виплатах
CREATE TABLE payouts_log (
    log_id SERIAL PRIMARY KEY,
    payout_id INT NOT NULL,
    operation VARCHAR(10) NOT NULL,
    changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    user_name TEXT DEFAULT CURRENT_USER,
    old_status payouts_status_type,
    new_status payouts_status_type,
    old_amount DECIMAL(12,2),
    new_amount DECIMAL(12,2)
);

-- Тригерна функція для логування змін
CREATE OR REPLACE FUNCTION log_payout_changes() RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO payouts_log (payout_id, operation, new_status, new_amount)
        VALUES (NEW.id, TG_OP, NEW.status, NEW.amount);
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO payouts_log (payout_id, operation, old_status, new_status, old_amount, new_amount)
        VALUES (NEW.id, TG_OP, OLD.status, NEW.status, OLD.amount, NEW.amount);
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO payouts_log (payout_id, operation, old_status, old_amount)
        VALUES (OLD.id, TG_OP, OLD.status, OLD.amount);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Додавання тригера до таблиці payouts
CREATE TRIGGER track_payout_changes
AFTER INSERT OR UPDATE OR DELETE ON payouts
FOR EACH ROW
EXECUTE FUNCTION log_payout_changes();

-- Тестування ENUM типу
INSERT INTO payouts (claim_id, amount, payout_date, method, status)
VALUES (1, 5000.00, '2023-12-01', 'bank_transfer', 'pending');

-- Оновлення статусу виплати
UPDATE payouts SET status = 'failed' WHERE id = 1;
SELECT * FROM payouts;
```

## Результат оновлення статусу:

3	1	1	15000.00	2023-05-28	bank_transfer	failed
---	---	---	----------	------------	---------------	--------

-- Видалення виплати

```
DELETE FROM payouts WHERE id = 1;
```

## Результат видалення:

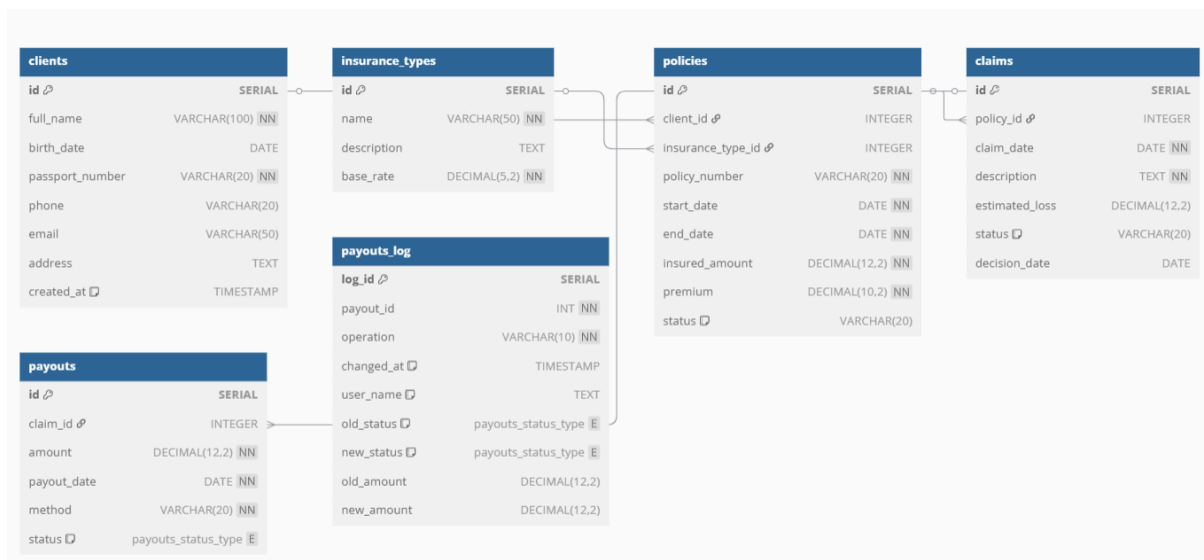
	id [PK] integer	claim_id integer	amount numeric (12,2)	payout_date date	method character varying (20)	status payouts_status_type
1	2	2	8000.00	2023-06-22	bank_transfer	completed
2	5	1	5000.00	2023-12-01	bank_transfer	pending

```
SELECT * FROM payouts_log;
```

## Результат вибірки:

	log_id [PK] integer	payout_id integer	operation character varying (10)	changed_at timestamp without time zone	user_name text	old_status payouts_status_type	new_status payouts_status_type	old_amount numeric (12,2)	new_amount numeric (12,2)
1	1	5	INSERT	2025-05-07 18:11:28.024496	postgres	[null]	pending	[null]	5000.00
2	2	1	UPDATE	2025-05-07 18:12:39.707467	postgres	completed	failed	15000.00	15000.00
3	3	1	DELETE	2025-05-07 18:14:51.445068	postgres	failed	[null]	15000.00	[null]

## 4. Оновлення діаграми:



З'явилася нова таблицка payouts\_log

У таблиці payouts з'явився новий стовпець status

Встановлено зв'язок між таблицями payouts та payouts\_log

**Висновок:** Під час виконання цієї лабораторної роботи було розширено функціонал бази даних створеної під час лабораторної №2. Створено користувацький тип для перевірки статусів виплат, реалізовано функцію, яка аналізує всі виплати конкретного клієнта, повертаючи загальну суму виплат та їх розподіл за статусами (завершені, очікуючі, невдалі). Оновлено ER-діаграму, яка показує розширений функціонал.

