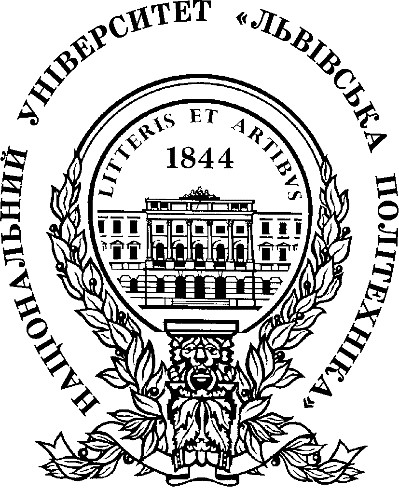
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра систем штучного інтелекту



ЗВІТ до лабораторної роботи № 12 **з курсу “ОБДЗ”** на тему:

# “ Розробка та застосування тригерів”

**Виконав:**

студент групи КН-208

Фіняк М.В.

**Викладач:** Якимишин Х.М.

# Лабораторна № 12

**Мета роботи:** Розробити SQL запити, які моделюють роботу тригерів:

каскадне знищення, зміна та доповнення записів у зв’язаних таблицях.

# Короткі теоретичні відомості

Тригер – це спеціальний вид користувацької процедури, який виконується автоматично при певних діях над таблицею, наприклад, при додаванні чи оновленні даних. Кожен тригер асоційований з конкретною таблицею і подією. Найчастіше тригери використовуються для перевірки коректності вводу нових даних та підтримки складних обмежень цілісності. Крім цього їх використовують для автоматичного обчислення значень полів таблиць, організації перевірок для захисту даних, збирання статистики доступу до таблиць баз даних чи реєстрації інших подій.

Для створення тригерів використовують директиву CREATE TRIGGER.

***Синтаксис:***

CREATE [DEFINER = { користувач | CURRENT\_USER }]

TRIGGER ім’я\_тригера час\_виконання подія\_виконання

ON назва\_таблиці FOR EACH ROW тіло\_тригера

***Аргументи:***

**DEFINER** - Задає автора процедури чи функції. За замовчуванням – це CURRENT\_USER.

**ім’я\_тригера** - Ім’я тригера повинно бути унікальним в межах однієї бази

даних.

**час\_виконання** - Час виконання тригера відносно події виконання. BEFORE –

виконати тіло тригера до виконання події, AFTER – виконати тіло тригера після події.

**подія\_виконання** - Можлива подія – це внесення (INSERT), оновлення (UPDATE), або видалення (DELETE) рядка з таблиці. Один тригер може бути пов’язаний лише з однією подією. Команда AFTER INSERT, AFTER UPDATE, AFTER DELETE визначає виконання тіла тригера відповідно після внесення, оновлення, або видалення даних з таблиці. Команда BEFORE INSERT, BEFORE UPDATE, BEFORE DELETE визначає виконання тіла тригера відповідно до внесення, оновлення, або видалення даних з таблиці.

**ON** - назва\_таблиці Таблиця, або віртуальна таблиця (VIEW), для якої створюється даний тригер. При видалені таблиці з бази даних, автоматично видаляються всі пов’язані з нею тригери.

**FOR EACH ROW тіло\_тригера** - Задає набір SQL директив, які виконує тригер. Тригер викликається і виконується для кожного зміненого рядка. Директиви можуть об’єднуватись командами BEGIN … END та містити спеціальні команди OLD та NEW для доступу до попереднього та нового значення поля у зміненому рядку відповідно. В тілі тригера дозволено викликати збережені процедури, але заборонено використовувати транзакції, оскільки тіло тригера автоматично виконується як одна транзакція.

**NEW.назва\_поля** - Повертає нове значення поля для зміненого рядка. Працює лише при подіях INSERT та UPDATE. У тригерах, які виконуються перед

(BEFORE) подією можна змінити нове значення поля командою SET NEW.назва\_поля = значення.

**OLD.назва\_поля** - Повертає старе значення поля для зміненого рядка. Можна використовувати лише при подіях UPDATE та DELETE. Змінити старе значення поля не можливо.

Щоб видалити створений тригер з бази даних, потрібно виконати команду **DROP TRIGGER** **назва\_тригера**.

Хід роботи

1. Розробимо тригер,який буде оновлювати значення цін в таблиці страв при внесенні нових інгредієнтів в таблицю інгредієнтів.

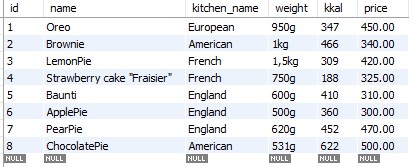
**CREATE TRIGGER trigger1 BEFORE INSERT ON ingredient**

**FOR EACH ROW**

**UPDATE dish SET price = price + 20 WHERE id = NEW.id;**

Початкові значення цін :

**SELECT \* FROM restaurant.dish;**

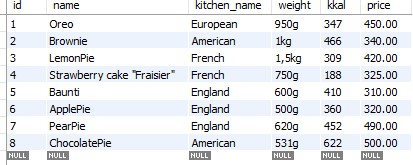


Перевіримо роботу тригера, додавши інгредієнти з id 6 i 7: **INSERT INTO ingredient VALUES**

**(6, 'DarkChocolate','4kg','300', 4),**

**(7, 'Sugar','10kg', '14.8', 5);**

**SELECT \* FROM restaurant.dish;**



1. Розробимо тригер,який при оновленні даних в таблиці інгредієнтів буде перевіряти чи нове значення ціни інгредієнта невід’ємне, якщо ж від’ємне, тоді значення ціни буде рівне нулю.

**delimiter //**

**CREATE TRIGGER trigger2 BEFORE UPDATE ON ingredient**

**FOR EACH ROW**

**BEGIN**

**IF NEW.price\_for\_unit < 0 THEN SET NEW.price\_for\_unit = 0;**

**END IF; END;// delimiter ;**

Початкові значення цін інгредієнтів :

**SELECT \* FROM restaurant.ingredient;**



Перевіримо роботу тригера, розробляючи запит для оновлення ціни інгредієнта з

id 2 :

**UPDATE ingredient SET price\_for\_unit = -29.50 WHERE id = 2;**



3. Розробимо тригер,який буде оновлювати значення ціни в таблиці страв при видаленні інгредієнта з таблиці інгредієнтів. А саме : значенню ціни страви буде присвоюватись значення ціни інгредієнта, який видаляємо.

**CREATE TRIGGER trigger3 BEFORE DELETE ON ingredient**

**FOR EACH ROW UPDATE dish SET price = OLD.price\_for\_unit WHERE**

**id = OLD.id;**

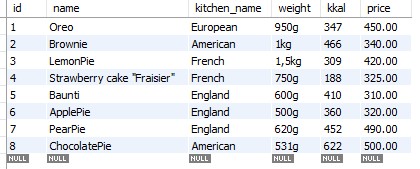
**Значення цін інгредієнтів :**

**SELECT \* FROM restaurant.ingredient;**



Початкові значення цін страв :

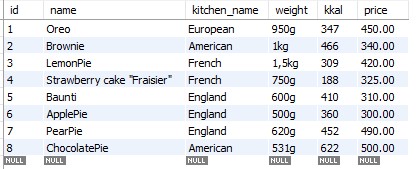
**SELECT \* FROM restaurant.dish;**



Перевіримо роботу тригера, видаливши інгредієнт з id 6 :

**DELETE FROM ingredient WHERE id=6;**

**SELECT \* FROM restaurant.dish;**



**Висновок:** під час виконання даної лабораторної роботи було розглянуто тригери, їх призначення, створення та використання. Здобув навики використання тригерів на практиці, розробивши тригери для таблиці ingredient.