МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту

Звіт до лабораторної роботи №12

з дисципліни
“Організація Баз Даних та знань”

Виконала:
ст. гр. КН-210
Заремба Вікторія

Викладач:
Мельникова Н.І.

Львів – 2020

Лабораторна робота №12
з курсу “ОБДЗ”
на тему:
“Розробка та застосування тригерів”

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

Потрібно розробити тригери, які виконуватимуть наступні дії.

1. Каскадне оновлення таблиці
2. Шифрування паролю користувача під час внесення в таблицю.
3. Тригер для перевірки вхідних даних

1. Каскадне оновлення таблиці

Створимо каскадне оновлення. При видаленні інвестора, значення id_investor в проміжній таблиці міняється на id_investor = 5 (“інше”)

Тригер:

```
CREATE
    TRIGGER investor_del BEFORE DELETE
    ON pm_system.investor FOR EACH ROW
    UPDATE pm_system.budget_investor SET id_investor=5 WHERE id_investor=OLD.id_investor;
```

Перевіримо роботу тригера, видаливши інвестора 2:

```
DELETE FROM investor WHERE id_investor=2;
SELECT * FROM budget_investor ORDER BY id_investor;
```

Запит до видалення:

	id_budget	id_investor	sum
1	6	1	11220
2	10	2	150
3	1	3	5000
4	2	3	2500
5	3	3	1500

Запит після видалення:

	id_budget	id_investor	sum
14	16	4	900
15	17	4	400
16	7	5	0
17	8	5	0
18	10	5	150

2. Шифрування паролю користувача під час внесення в таблицю.

Для кодування паролю використаємо AES_ENCRYPT, а також переведемо у HEX для коректного збереження в таблиці.

```
CREATE
    TRIGGER user_pass BEFORE
    INSERT ON pm_system.user FOR EACH ROW
    SET NEW.password = HEX(AES_ENCRYPT(NEW.password, 'key-key'));
```

Перевіримо роботу тригера

```
INSERT INTO user VALUES
    (null,'Zhyk', 'Anastasia', 'nastiositi@gmail.com', '+380946735103', 'j43rfiejfioj398ufiej'),
    (null,'Kruchkovska', 'Krystya', 'Krych@gmail.com', '+398046294683', 'fijeodks');

SELECT * FROM user LIMIT 12, 2;
```

	id...	surname	name	password
1	13	Zhyk	Anastasia	8C4A45EF920198C14154F72FFD9…
2	14	Kruchkovska	Krystya	DA0CA4918567301979E45DD3A19…

3. **Тригер для перевірки вхідних даних**

Перевіримо суму інвестицій в таблиці budget_investor. У випадку некоректної суми виведемо SIGNAL

```
CREATE TRIGGER budget_investor_sum
   BEFORE INSERT ON budget_investor
   FOR EACH ROW
   BEGIN
      IF NEW.sum = 0 THEN
         SIGNAL SQLSTATE '45000'
         SET MESSAGE_TEXT = 'Некоректна сума інвестицій';
      end if;
   end;
```

Перевіримо роботу тригера:

```
INSERT INTO budget_investor VALUE (4,5, 0);
```

Результат:

```
pm_system> INSERT INTO budget_investor VALUE (4,5, 0)
[2020-05-20 03:26:27] [45000][1644] Некоректна сума інвестицій
[2020-05-20 03:26:27] [HY000][1644] Некоректна сума інвестицій
```

Висновок*:* на цій лабораторній роботі було розглянуто тригери, їх призначення, створення та використання