Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

Лабораторна робота №12
на тему:
«Розробка та застосування тригерів»
з курсу:
«ОБДЗ»

Виконала:
ст. гр. КН-210
Ямнюк Аліна
Прийняла:
Мельникова Н.І.

Львів – 2020 р.

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) Каскадне оновлення таблиці order при видаленні працівника з таблиці staff.

Перевіримо початкові дані

```
SELECT * FROM beauty.`order` ;
```

	id_order	id_service	id_staff
1	1	3	1
2	2	2	2
3	3	1	3

Діюче обмеження зовнішнього ключа при видаленні працівника (id_staff = 2) з таблиці staff встановлює id_staff = null. Натомість, за допомогою тригера, присвоїмо id_staff = 1

```
DROP TRIGGER IF EXISTS example;
CREATE TRIGGER example BEFORE DELETE
    ON beauty.staff FOR EACH ROW
    UPDATE beauty.`order` SET id_staff = 1 WHERE id_staff = OLD.id_staf

DELETE FROM beauty.staff WHERE  id_staff = 2;
```

Результат:

	id_order	id_service	id_staff
1	1	3	1
2	2	2	1
3	3	1	3

2) Тригер, який при додаванні нового користувача, створює поле created і встановлює час додавання цього користувача.

```
ALTER TABLE beauty.isclient
    ADD COLUMN created DATETIME DEFAULT '01.01.01 00:00:00';

DROP TRIGGER IF EXISTS example1;
CREATE TRIGGER example1 BEFORE INSERT
    ON beauty.isclient FOR EACH ROW
    SET NEW.created = now();
```

Новий користувач

```
insert into isclient (id_client, first_name, last_name, telephone, isBlocked, created)
values (4, 'Sofia', 'Golub', '0937777777', 0, NULL);
```

Виконаємо запит та перевіримо результат

```
SELECT * FROM  isclient;
```

Результат:

	id_client	first_name	last_name	telephone	isBlocked	created
1	1	Solomia	Leshchak	0937641190	0	2001-01-01 00:00:00
2	2	Tania	Adel	0958769302	0	2001-01-01 00:00:00
3	3	Sofia	Kovtun	0936348390	0	2001-01-01 00:00:00
4	4	Sofia	Golub	0937777777	0	2020-05-14 02:45:58

3) Тригер, який перевіряє чи номер майстра починається з «380». Дані is_ukranian boolean default = 1.Відповідно, якщо номер майстра не український , то поле is_ukranian = 0

```
ALTER TABLE beauty.staff
    ADD COLUMN is_ukranian BOOLEAN DEFAULT 1;

DROP TRIGGER IF EXISTS example2;
CREATE TRIGGER example2 BEFORE UPDATE
    ON beauty.staff FOR EACH ROW
    IF (NEW.telephone NOT LIKE'380%') THEN
        SET NEW.is_ukranian = 0;
    END IF;
```

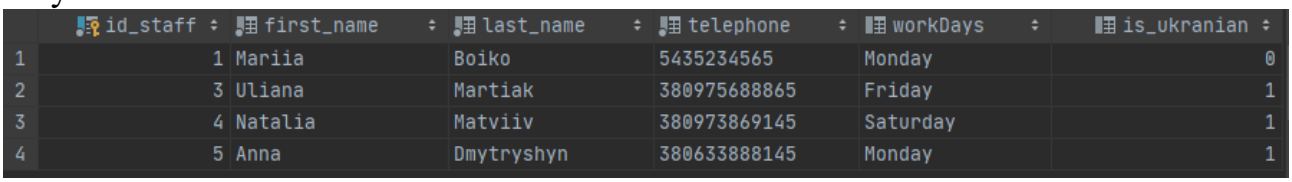
Додаємо номер , який починається з «543»

```
UPDATE beauty.staff SET telephone = '5435234565' WHERE id_staff = 1;
```

Виконаємо запит та перевіримо результат

```
SELECT * FROM staff;
```

Результат:



	id_staff	first_name	last_name	telephone	workDays	is_ukranian
1	1	Mariia	Boiko	5435234565	Monday	0
2	3	Uliana	Martiak	380975688865	Friday	1
3	4	Natalia	Matviiv	380973869145	Saturday	1
4	5	Anna	Dmytryshyn	380633888145	Monday	1

Висновок: на цій лабораторній роботі я навчилась розробляти SQL запити, які моделюють роботу тригерів.