МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"
ІНСТИТУТ ПІСЛЯДИПЛОМНОЇ ОСВІТИ
КАФЕДРА СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №12

Підготувала:

Студентка групи КН-209

Кульчицька Олена

Викладач:

Мельникова Н.І.

2020

Лабораторна робота №12

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 інгредієнтів при видаленні постачальника з таблиці *provider*.
2. Робимо перевірку на від'ємність ціни в таблиці *ingredient* .
3. При видаленні інгредієнта ціна страви за його id міняється на ціну price_for_unit.

1. Каскадне оновлення таблиці користувачів при видаленні ролі з таблиці *provider*. Діюче обмеження зовнішнього ключа при видалені ролі встановлює для користувача невизначену роль (значення NULL). Натомість, за допомогою тригера, користувачеві потрібно присвоювати певну роль за замовчуванням (роль Ingredient з provider_id =1).
   Перевіримо місткість таблиць:

```
SELECT * FROM confectionary.ingredient;
SELECT * FROM confectionary.provider;
```

id	name	price_for_unit	quantity	provider_id
1	flour	89.90	10	2
2	egg	1.50	50	1
3	milk	20.50	10	2
4	sugar	30.00	40	2
5	chocolate	30.00	50	2
6	cacao	20.00	10	2
7	carrot	10.00	10	1
8	cherry	20.00	50	1
9	blueberry	10.00	100	1
10	apple	10.00	50	1
11	darkCho...	10.30	1	1
12	nut	20.80	10	3
NULL	NULL	NULL	NULL	NULL

id	name	telephone	classification	email
1	Astra	2346578	eggs	astra@gmail.com
2	ProZorro	2345668	flour	prozzoro@gmail.com
3	Aro	2345768	nuts	aro@gmail.com
NULL	NULL	NULL	NULL	NULL

Створюємо тригер.

```
CREATE TRIGGER trigger1 BEFORE DELETE
ON confectionary.provider FOR EACH ROW
UPDATE ingredient SET provider_id = 1 WHERE id = OLD.id;
```

Перевіримо роботу тригера, видаливши провайдера з номером 3:

id	name	price_for_unit	quantity	provider_id
1	flour	89.90	10	2
2	egg	1.50	50	1
3	milk	20.50	10	1
4	sugar	30.00	40	2
5	chocolate	30.00	50	2
6	cacao	20.00	10	2
7	carrot	10.00	10	1
8	cherry	20.00	50	1
9	blueberry	10.00	100	1
10	apple	10.00	50	1
11	darkCho...	10.30	1	1
NULL	NULL	NULL	NULL	NULL

2. Оскільки ціна не може бути від'ємною. Робимо перевірку на це тригером. Якщо вона є від'ємною, то ставимо 0.

   Тригер:

```
delimiter //
CREATE TRIGGER trigger2 BEFORE UPDATE
 ON confectionary.ingredient FOR EACH ROW
    BEGIN
      IF NEW.price_for_unit < 0 THEN SET NEW.price_for_unit = 0;
      END IF;
    END;//
 delimiter ;
```

Дивимось на вміст таблички *ingredient.*

```
SELECT * FROM confectionary.ingredient;
```

id	name	price_for_unit	quantity	provider_id
1	flour	89.90	10	2
2	egg	1.50	50	1
3	milk	20.50	10	1
4	sugar	30.00	40	2
5	chocolate	30.00	50	2
6	cacao	20.00	10	2
7	carrot	10.00	10	1
8	cherry	20.00	50	1
9	blueberry	10.00	100	1
10	apple	10.00	50	1
11	darkCho...	10.30	1	1
NULL	NULL	NULL	NULL	NULL

Перевіряємо роботу тригера, оновлюючи поле ціни на від'ємне число.

```
UPDATE ingredient SET price_for_unit = -29.50 WHERE id = 2;
```

id	name	price_for_unit	quantity	provider_id
1	flour	89.90	10	2
2	egg	0.00	50	1
3	milk	20.50	10	1
4	sugar	30.00	40	2
5	chocolate	30.00	50	2
6	cacao	20.00	10	2
7	carrot	10.00	10	1
8	cherry	20.00	50	1
9	blueberry	10.00	100	1
10	apple	10.00	50	1
11	darkCho...	10.30	1	1

3. При видаленні інгредієнта ціна страви за його id міняється на ціну price_for_unit.

```
SELECT * FROM confectionary.dish;
SELECT * FROM confectionary.ingredient;
```

Вміст таблиць:

id	name	weight	price	kkal
1	tiramisu	200	150.00	460
2	cheesecake	500	200.00	1100
3	carrot cake	500	170.00	900
4	apple pie	600	150.00	920
5	oreo cheesecake	500	210.00	1000
6	blueberry cake	600	180.00	920
7	nutcake	700	190.00	950
8	cherry pie	600	140.00	910
NULL	NULL	NULL	NULL	NULL

id	name	price_for_unit	quantity	provider_id
1	flour	89.90	10	2
2	egg	0.00	50	1
3	milk	20.50	10	1
4	sugar	30.00	40	2
5	chocolate	30.00	50	2
6	cacao	20.00	10	2
7	carrot	10.00	10	1
8	cherry	20.00	50	1
9	blueberry	10.00	100	1
10	apple	10.00	50	1
11	darkCho...	10.30	1	1
NULL	NULL	NULL	NULL	NULL

Тригер:

```
CREATE TRIGGER trigger3 BEFORE DELETE
ON confectionary.ingredient
FOR EACH ROW
UPDATE confectionary.dish SET price = OLD.price_for_unit WHERE id = OLD.id;
```

Перевіряємо роботу тригера, видаливши інгредієнт, id якого 6.

```
DELETE FROM confectionary.ingredient WHERE id=6;
```

id	name	weight	price	kkal
1	tiramisu	200	150.00	460
2	cheesecake	500	200.00	1100
3	carrot cake	500	170.00	900
4	apple pie	600	150.00	920
5	oreo cheesecake	500	210.00	1000
6	blueberry cake	600	20.00	920
7	nutcake	700	190.00	950
8	cherry pie	600	140.00	910
NULL	NULL	NULL	NULL	NULL

Висновок

На цій лабораторній роботі було розглянуто тригери, їх призначення, створення та використання. Було розроблено тригери для таблиць Ingredient, Order та Customer, Provider.