МІНІСТЕРСТВО ОСВІТИ І НАУКИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Лабораторна робота №12
З дисципліни «Організація баз даних та знань»

Виконав:
студент групи КН-210
Черещук Любомир

Перевірив:
Кандидат тех. наук, ст. викладач
Мельникова Н. І.

Львів – 2020

Мета: Розробити SQL запити, які моделюють роботу тригерів: каскадне знищення, зміна та доповнення записів у зв’язаних таблицях.

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 product при видаленні типу з таблиці type.
2. Шифрування паролю користувача під час внесення в таблицю.
3. Тригер для таблиці order, який буде фіксувати у таблиці user дату останнього замовлення користувача.

1. Перевіримо записи таблиці product:

SELECT * FROM sportproducts.product

id	name	price	Brand_id	Type_id	Category_id	description
1	tshort	600.00	1	1	2	for running
2	sportshoes	1100.00	3	3	1	for bodybuilding
3	shorts	800.00	4	1	2	for running
4	protein	1200.00	1	2	1	for bodybuilding
5	creatine	380.00	5	2	1	forbodybuilding
6	pants	500.00	1	1	2	forrunnig
7	grif	1580.00	2	4	1	20kggrig
8	ball	800.00	6	4	3	forcrossfit
9	bca	400.00	5	2	1	forbodybuilding

Тепер видалимо 2 тип, і подивимось, що станеться з записами в product:

```
DELETE FROM sportproducts.type

WHERE id = 2;
```

Результ:

```
Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`sportproducts`.`order_prod.
```

Не встановлено, що слід робити при видаленні поля цього ключа.

Для цього, створимо тригер, який встановлюватиме значення Type_id за замовчуванням 4.

Код створення тригера:

```
CREATE TRIGGER type_delete

BEFORE DELETE ON sportproducts.type

FOR EACH ROW

UPDATE sportproducts.product SET Type_id=4 WHERE Type_id=OLD.id;
```

Тепер видалимо 2 тип:

```
DELETE FROM sportproducts.type

WHERE id = 2;
```

```
3  22:49:27  DELETE FROM sportproducts.type WHERE id = 2
```

SELECT * FROM sportproducts.product

Результа:

id	name	price	Brand_id	Type_id	Category_id	description
1	tshort	600.00	1	1	2	for running
2	sportshoes	1100.00	3	3	1	for bodybuilding
3	shorts	800.00	4	1	2	for running
4	protein	1200.00	1	4	1	for bodybuilding
5	creatine	380.00	5	4	1	forbodybuilding
6	pants	500.00	1	1	2	forrunnig
7	grif	1580.00	2	4	1	20kggrig
8	ball	800.00	6	4	3	forcrossfit
9	bca	400.00	5	4	1	forbodybuilding

type_id в елементів з типом 2 змінився на значення за замовчуванням 4.

2. Шифрування паролю користувача під час внесення в таблицю.

Код створення тригера:

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

Тепер спробуємо додати нових користувачів:

```
INSERT INTO sportproducts.user VALUES
(7, 'John', 'Coffe', 'johncoffe@gmail.com', 'pass7'),
(8, 'Bill', 'Tumber', 'billtumber@gmail.com', 'pass8'),
(9, 'Jery', 'Lon', 'jerylon@gmail.com', 'pass9');

select * from user
where id > 6;
```

Результат:

7	John	Coffe	johncoffe@gmail.com	BLOB
8	Bill	Tumber	billtumber@gmail.com	BLOB
9	Jery	Lon	jerylon@gmail.com	BLOB

3. Додамо поле last_ activity до таблиці User, і будемо заносити туди ту дату та час, коли користувач робив востаннє замовлення в магазині.

Спершу модифікуємо таблицю user:

```
ALTER TABLE sportproducts.user
ADD COLUMN last_activity DATETIME DEFAULT NULL;
```

Далі створюємо тригер:

```
CREATE TRIGGER user_last_activity
AFTER INSERT ON sportproducts.order
FOR EACH ROW
UPDATE sportproducts.user SET user.last_activity=(NEW.datetime)
WHERE user.id=NEW.User_id;
```

Тепер протестуємо роботу тригера:

```
INSERT INTO sportproducts.order VALUES
(10, '2020-06-29 21:21:25', 1),
(11, '2020-04-27 21:21:25', 2);
```

Результат:

Таблиця user:

id	name	surname	email	password	last_activity
1	Ivan	Mortu	ivan.brain@gmail.com	NULL	2020-06-29 21:21:25
2	Igor	Smit	igor.smit@gmail.com	NULL	2020-04-27 21:21:25

Отже, як бачимо, при зроблені користувачем замовлення, дата та час створення замовлення записуватиметься в таблицю user в поле last_activity автоматично за допомогою тригера.

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