Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп’ютерних наук та інформаційних технологій

Кафедра «Системи штучного інтелекту»

Лабораторна робота №12

З дисципліни:

«Організація баз даних та знань»

Виконав:
Студент групи КН-208
Лукаш О.В.
Викладач:
Мельникова Н.І.

Львів-2020

Тема: Розробка та застосування тригерів

Мета: 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 *repair_work* при видаленні механіка з таблиці *mechanic*.

2. Змінити кількість автомобілей які обслуговує механік при добавлення нового авто у даний автосервіс.

3. Змінити кількість доступних деталей у магазині при купівлі їх для виконання ремонтних робіт.

1. При звільненні механіка з роботи, автомобілі які він обслуговував повинні отримати нового майстра. За допомогою тригера, всі автомобілі, які обслуговував видалений механік потрібно присвоїти іншого майстра, в моєму випадку це механік із id_mechanic=2.

CREATE TRIGGER delete_mechanic **BEFORE DELETE ON** mechanic **FOR EACH ROW**

UPDATE repair_work **SET** id_mechanic = 2 **WHERE** old.id_mechanic=id_mechanic;

Перевіримо роботу тригера, видаливши механіка з номером 1:

SELECT * **FROM** car_servise.repair_work;

id_work	id_car	id_centrum	status	date	id_mechanic
1	1	1	in_progres	2020-04-25	2
2	2	1	completed	2020-02-12	2
3	3	2	in_progres	2020-04-02	3
4	4	2	in_progres	2020-05-05	4
5	5	3	completed	2020-02-10	5
6	6	1	in_progres	2020-05-12	1
NULL	NULL	NULL	NULL	NULL	NULL

SELECT * **FROM** car_servise.mechanic;

id_mechanic	name	surname	phone	work_year	work_place
1	Petro	Petrov	0962531452	2020-05-12	1
2	Andriy	Baran	0963265485	2018-02-28	1
3	Maxim	Bombal	0970356214	2015-02-20	2
4	Vova	Mantor	0671325984	2018-09-26	2
5	Vasyl	Menshot	0697458635	2019-03-28	3
NULL	NULL	NULL	NULL	NULL	NULL

DELETE FROM mechanic **WHERE** id_mechanic=1;

SELECT * **FROM** car_servise.mechanic;

id_mechanic	name	surname	phone	work_year	work_place
2	Andriy	Baran	0963265485	2018-02-28	1
3	Maxim	Bombal	0970356214	2015-02-20	2
4	Vova	Mantor	0671325984	2018-09-26	2
5	Vasyl	Menshot	0697458635	2019-03-28	3
NULL	NULL	NULL	NULL	NULL	NULL

SELECT * **FROM** car_servise.repair_work;

id_work	id_car	id_centrum	status	date	id_mechanic
1	1	1	in_progres	2020-04-25	2
2	2	1	completed	2020-02-12	2
3	3	2	in_progres	2020-04-02	3
4	4	2	in_progres	2020-05-05	4
5	5	3	completed	2020-02-10	5
6	6	1	in_progres	2020-05-12	2
NULL	NULL	NULL	NULL	NULL	NULL

2. Змінити кількость автомобілей які обслуговує механік при добавлення нового авто у даний автосервіс.

CREATE TRIGGER add_new_car **AFTER INSERT ON** repair_work **FOR EACH ROW**

UPDATE mechanic **SET** num_of_car=num_of_car+1 **WHERE** New.id_mechanic=id_mechanic;

Перевіримо роботу тригера, додамо новий автомобіль до 2-го сервісного центру і дамо йому механіка з id_mechanic=3:

SELECT * **FROM** car_servise.mechanic;

id_mechanic	name	surname	phone	work_year	work_place	num_of_car
2	Andriy	Baran	0963265485	2018-02-28	1	3
3	Maxim	Bombal	0970356214	2015-02-20	2	1
4	Vova	Mantor	0671325984	2018-09-26	2	1
5	Vasyl	Menshot	0697458635	2019-03-28	3	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

INSERT INTO repair_work

VALUE(7, 7, 2, "in_progres","2020-05-12", 3);

id_work	id_car	id_centrum	status	date	id_mechanic
1	1	1	in_progres	2020-04-25	2
2	2	1	completed	2020-02-12	2
3	3	2	in_progres	2020-04-02	3
4	4	2	in_progres	2020-05-05	4
5	5	3	completed	2020-02-10	5
6	6	1	in_progres	2020-05-12	2
7	7	2	in_progres	2020-05-12	3
NULL	NULL	NULL	NULL	NULL	NULL

SELECT * **FROM** car_servise.mechanic;

id_mechanic	name	surname	phone	work_year	work_place	num_of_car
2	Andriy	Baran	0963265485	2018-02-28	1	3
3	Maxim	Bombal	0970356214	2015-02-20	2	2
4	Vova	Mantor	0671325984	2018-09-26	2	1
5	Vasyl	Menshot	0697458635	2019-03-28	3	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Змінити кількость доступних деталей у магазині при купівлі їх для виконання ремонтних робіт.

CREATE TRIGGER count_details_in_store **AFTER UPDATE ON** check_list **FOR EACH ROW**

UPDATE repair_detail **SET** count_in_store=count_in_store-1 where new.id_detail=id_detail;

Перевіримо роботу тригера, в таблиці *check_list* змінемо поле *id_detail*:

SELECT * **FROM** car_servise.repair_detail;

id_detail	name	price	id_store	brend_original	count_in_store
1	detail_1	500	1	1	42
2	detail_2	350	1	0	23
3	detail_3	800	2	1	12
NULL	NULL	NULL	NULL	NULL	NULL

UPDATE check_list

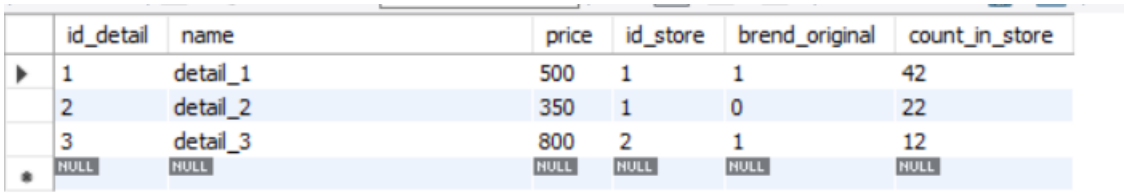
SET id_detail = 2

WHERE id_repair=1 and id_service=6;

SELECT * **FROM** car_servise.check_list;

id_repair	id_service	garante_month	id_detail
1	6	8	2
1	8	6	NULL
1	1	6	1
1	2	6	NULL
2	6	8	NULL
2	1	6	NULL
2	7	6	NULL
3	2	6	NULL

SELECT * **FROM** car_servise.repair_detail;



id_detail	name	price	id_store	brend_original	count_in_store
1	detail_1	500	1	1	42
2	detail_2	350	1	0	22
3	detail_3	800	2	1	12
NULL	NULL	NULL	NULL	NULL	NULL

Висновок: на даній лабораторній роботі було розглянуто тригери, їх призначення, створення та використання. Мною було розроблено тригери **BEFORE DELETE, AFTER INSERT, AFTER UPDATE**.