МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту



Лабораторна робота №12

З курсу “Організація баз даних та знань”

Виконав:  
студент групи КН-210

Бурак Марко

Викладач:

Мельникова Наталя Іванівна

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.

**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.

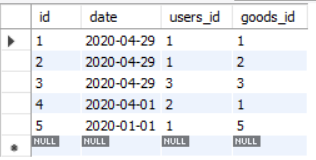
1. Каскадне оновлення таблиці accounting при видаленні юзера з таблиці users.

2. Шифрування паролю користувача під час внесення в таблицю.

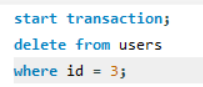
3. Тригер для таблиці Comments, який буде фіксувати у таблиці Users дату останнього написаного коментаря.

1. Перевіримо записи таблиці auto:





Тепер видалимо 3 юзера, і подивимось, що станеться з записами в accounting.



Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`store`.`comments`, CONSTRAINT `comment\_user` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`))

Виникла помилка, тому що було видалення зовнішнього ключа з таблиці accounting.

Для каскаду, створимо тригер, який встановлюватиме значення user\_id за замовчуванням 1 у двох таблицях.

Код створення тригера:

Delimiter //

CREATE TRIGGER user\_delete

BEFORE DELETE ON store.users

FOR EACH ROW Begin

UPDATE store.accounting SET users\_id=1 WHERE users\_id=OLD.id;

UPDATE store.comments SET user\_id=1 WHERE user\_id=OLD.id;

END;

//

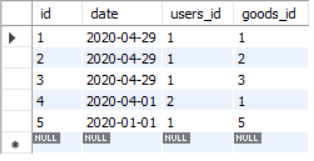
Delimiter ;

Тепер спробуємо видалити деякі типи:

delete from users

where id =3

Результат:



users\_id змінився на значення за замовчуванням 1.

2. Шифрування паролю користувача під час внесення в таблицю.

Код створення тригера:

delimiter //

create trigger crypting

before insert on users

for each row

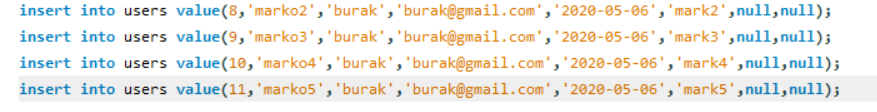
begin

set new.crypt = aes\_encrypt(new.password,'key-key');

set new.decrypt = aes\_decrypt(new.crypt,'key-key');

end;

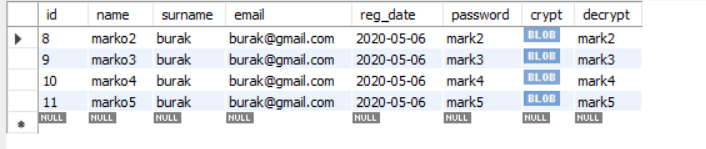
Тепер спробуємо додати нових користувачів:



select \* from users

where id >7

Результат:



3. Додамо поле last до таблиці users, і будемо заносити туди ту дату та час, коли користувач опублікував останній коментар.

Спершу модифікуємо таблицю user:

ALTER TABLE users

ADD COLUMN last DATE;

Далі створюємо тригер:

create trigger activity

after insert on store.comments

for each row

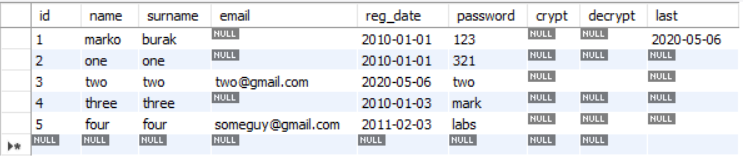
update users set users.last = new.date

where users.id = new.user\_id

Тепер протестуємо роботу тригера:

insert into comments value(7,'mark',2,1,'2020-05-06')

Таблиця user:



Отже, як бачимо, при додаванні користувачем коментаря, дата останнього запису записуватиметься в таблицю user в поле last автоматично за допомогою тригера.

**Висновок**: на цій лабораторній роботі було розглянуто тригери, їх

призначення, створення та використання.