МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота 12

з організації баз даних та знать

Виконав:

Студент групи КН-208

Воробель Адріан

Викладач:

Якимишин Х. М.

Львів – 2020р.

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 після видалення

```
create trigger distrib_delete before delete
on autoshop.distributor for each row
update autoshop.cardistributor set distrib_id = OLD.id-1
where distrib_id = OLD.id;
```

Видалимо одного з дистрибюторів

	id	car_id	distrib_id
	2	2	2
	3	3	3
	4	4	1
	5	5	2
	6	6	1
	7	7	1
	43	8	4
	44	9	4
	45	10	4
	60	11	1
	61	12	2

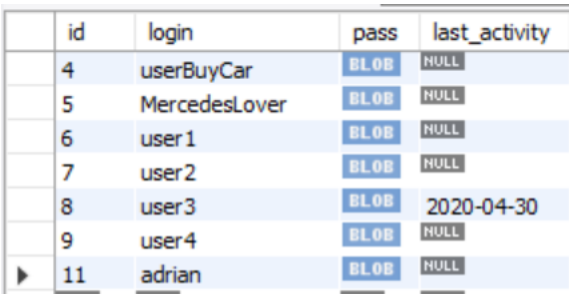
```
delete from distributor where id = 4;
```

	id	car_id	distrib_id
	6	6	1
	7	7	1
	43	8	3
	44	9	3
	45	10	3
	60	11	1
	61	12	2
	62	13	1

2. Шифрування паролю після додавання нового користувача

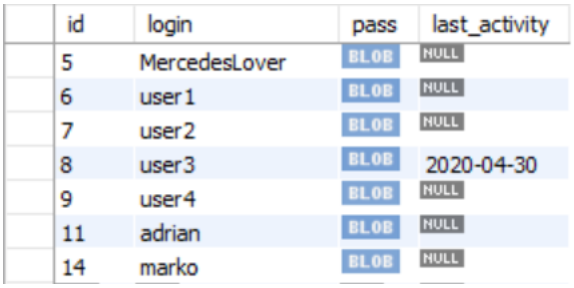
```
CREATE

TRIGGER encrypt_password BEFORE INSERT ON autoshop.users FOR EACH ROW

SET NEW.pass = AES_ENCRYPT(NEW.pass, 'key-key');
```

А тепер додаємо якогось користувача



id	login	pass	last_activity
4	userBuyCar	BLOB	NULL
5	MercedesLover	BLOB	NULL
6	user1	BLOB	NULL
7	user2	BLOB	NULL
8	user3	BLOB	2020-04-30
9	user4	BLOB	NULL
11	adrian	BLOB	NULL

```
insert into users(login, pass) value ('marko', 'adrian2001');
```

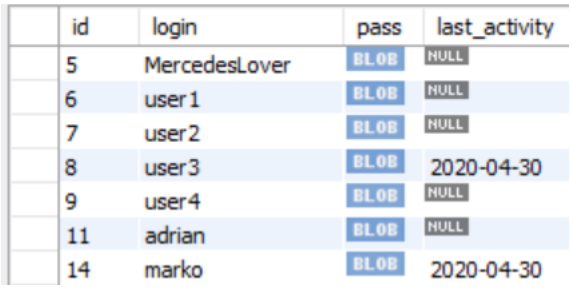


id	login	pass	last_activity
5	MercedesLover	BLOB	NULL
6	user1	BLOB	NULL
7	user2	BLOB	NULL
8	user3	BLOB	2020-04-30
9	user4	BLOB	NULL
11	adrian	BLOB	NULL
14	marko	BLOB	NULL

3. Додавання дати останньої активності користувача після того як користувач зробить замовлення

```
create trigger last_activity after insert

on autoshop.orders for each row

update autoshop.users set last_activity = current_date()

where users.id = NEW.user_id;
```

Додаємо замовлення



id	login	pass	last_activity
5	MercedesLover	BLOB	NULL
6	user1	BLOB	NULL
7	user2	BLOB	NULL
8	user3	BLOB	2020-04-30
9	user4	BLOB	NULL
11	adrian	BLOB	NULL
14	marko	BLOB	2020-04-30

```
insert into orders(user_id, carDistrib_id) value (12,64);
```

	id	login	pass	last_activity
	5	MercedesLover	BLOB	NULL
	6	user1	BLOB	NULL
	7	user2	BLOB	NULL
	8	user3	BLOB	2020-04-30
	9	user4	BLOB	NULL
	11	adrian	BLOB	2020-04-30
	14	marko	BLOB	2020-04-30

Висновок.

На цій лабораторній роботі було розглянуто тригери, їх призначення, створення та використання.