Лабораторна робота №12
з курсу “ОБДЗ”
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 користувачів при видаленні ролі з таблиці Role.
2. Шифрування паролю користувача під час внесення в таблицю.
3. Тригер для таблиці Session, який буде фіксувати у таблиці Author дату останнього входу користувача в систему.

1. Каскадне оновлення таблиці користувачів при видаленні ролі з таблиці Role. 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 Guests з roleID=4).

```
CREATE
 TRIGGER role_delete BEFORE DELETE
 ON mycms.role FOR EACH ROW
 UPDATE mycms.author SET roleID=4 WHERE roleID=OLD.roleID;
```

Перевіримо роботу тригера, видаливши роль з номером 9:

```
DELETE FROM mycms.role WHERE roleID=9;
SELECT * FROM mycms.author LIMIT 10, 5;
```

authorID	name	login	password	created	email	profile	roleID
37	*NULL*	user7	[BLOB - 6 Bytes]	2008-04-16 00:00:00	user7@kml.com	*NULL*	2
38	*NULL*	user8	[BLOB - 6 Bytes]	2009-04-22 00:00:00	user8@ml.com	*NULL*	2
39	*NULL*	superuser4	[BLOB - 11 Bytes]	2009-05-01 00:00:00	suser4@gl.com	*NULL*	4
40	*NULL*	suser5	[BLOB - 12 Bytes]	2009-05-01 00:00:00	suser5@gl.com	*NULL*	4

2. Створимо тригер, який буде шифрувати пароль користувача функцією AES_ENCRYPT перед тим як внести його у таблицю Author.

```
CREATE
 TRIGGER author_password BEFORE
 INSERT ON mycms.author FOR
 EACH ROW
 SET NEW.password = AES_ENCRYPT(NEW.password, 'key-key');
```

Перевіримо виконання тригера:

```
INSERT INTO mycms.author VALUES
 (NULL, NULL, 'superuser6', 'suser6_pass', '2009-
 05-09', 'suser6@g.com', NULL, 6),
 (NULL, NULL, 'superser7', 'suser7_pass', '2009-
 05-09', 'suser7@g.com', NULL, 6);

SELECT * FROM mycms.author LIMIT 14, 2;
```

authorID	name	login	password	created	email	profile	roleID
41	*NULL*	superuser6	[BLOB - 16 Bytes]	2009-05-09 00:00:00	suser6@g.com	*NULL*	6
42	*NULL*	superser7	[BLOB - 16 Bytes]	2009-05-09 00:00:00	suser7@g.com	*NULL*	6

3. У таблицю Session за допомогою тригера потрібно записувати службову інформацію при кожному вході користувача у систему. Тригер буде фіксувати дату входу і записувати її у таблицю користувачів.

Перед створенням тригера, створимо нове поле lastseen у таблиці Author.

```
ALTER TABLE mycms.author
 ADD COLUMN lastseen DATE DEFAULT NULL;

CREATE TRIGGER user_lastseen AFTER
 INSERT ON mycms.session FOR EACH
 ROW
 UPDATE mycms.author SET author.lastseen=DATE(NEW.start)
 WHERE author.authorID=NEW.authorID;
```

Перевіримо роботу тригера:

```
INSERT INTO session VALUES
('2weQ34rt', '3', '2009-05-09 00:01:01', NULL),
('314eqrtE', '4', '2009-05-10 22:01:51', NULL);
SELECT authorID, login, email, lastseen FROM author LIMIT
3;
```

authorID	login	email	lastseen
2	admin	admin@admin.com	*NULL*
3	user1	user1@gmail.com	2009-05-09
4	user2	user2@gmail.com	2009-05-10

Висновок: на цій лабораторній роботі було розглянуто тригери, їх призначення, створення та використання. Було розроблено тригери для таблиць Session, Role та Author.