МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"
ІНСТИТУТ ПІСЛЯДИПЛОМНОЇ ОСВІТИ
КАФЕДРА СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №11

Підготувала:

Студентка групи КН-209

Кульчицька Олена

Викладач:

Мельникова Н.І.

2020

Лабораторна робота №11

на тему:

“Розробка та застосування транзакцій”

Мета роботи

Навчитися використовувати механізм транзакцій у СУБД MySQL. Розробити SQL запити, які виконуються як єдине ціле в рамках однієї транзакції.

Короткі теоретичні відомості

Транзакція – це сукупність директив SQL, які виконуються як єдине ціле з можливістю відміни результатів їх виконання. Зміни в таблицях записуються у базу даних лише після успішного виконання всіх директив транзакції. Інакше, всі зроблені зміни ігноруються. Це дозволяє уникати помилок при маніпулюванні великими обсягами записів, зберігати цілісність даних при помилках під час додавання, видалення, модифікації значень у різних таблицях і полях тощо. СУБД MySQL також підтримує глобальні розподілені транзакції, які виконуються на декількох базах даних, або на різних серверах баз даних (XA-транзакції). Для організації транзакцій в MySQL використовують такі директиви, як SET autocommit, START TRANSACTION, COMMIT і ROLLBACK.

START TRANSACTION

Вказує на початок транзакції. Директива вимикає автоматичне збереження змін для всіх подальших запитів, поки не буде виконано команду COMMIT, або ROLLBACK.

COMMIT

Зберегти зміни, зроблені даною транзакцією.

ROLLBACK

Відмінити дану транзакцію і зроблені нею зміни у базі даних. Слід зауважити, що зміни у схемі бази даних не можна відмінити, тобто результат видалення, зміни або створення таблиці завжди зберігається.

SET autocommit=0

Вимикає автоматичне збереження змін для поточної сесії зв’язку з сервером БД. За замовчуванням, зміни зберігаються автоматично, тобто результат виконання запиту, який змінює таблицю, одразу записується на диск без можливості відміни операції.

AND CHAIN

Одразу після завершення даної транзакції розпочати виконання наступної.

RELEASE

Одразу після виконання даної транзакції завершити поточну сесію зв’язку з сервером.

Транзакції можна розбивати на окремі логічні частини, оголошуючи так звані точки збереження. Це дозволяє відміняти результати виконання не всієї транзакції, а лише тих запитів, які виконувались після оголошеної точки збереження (SAVEPOINT).

SAVEPOINT *мітка*

Оголошує точку збереження всередині транзакції та задає її назву.

ROLLBACK TO [SAVEPOINT] *мітка*

Відміняє результати виконання запитів, вказаних після даної точки збереження.

RELEASE SAVEPOINT *мітка*

Видаляє точку збереження.

Хід роботи

1. Відміна транзакції.

Транзакція складається з двох запитів на додавання нових інгредієнтів з постачальниками "Astra" та "Aro". При цьому, постачальника "Aro" з id=3 в базі даних не існує, а отже, транзакція не виконується.

```
USE Confectionary;
START TRANSACTION;
INSERT INTO confectionary.ingredient VALUES
    (NULL, 'darkChocolate', 10.30, 1 , 1),
    (NULL, 'nut', 20.80, 10, 3);
COMMIT;
```

Відповідь сервера:

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`confectionary`.`ingredient`, CONSTRAINT `provider` FOREIGN KEY (`provider_id`) REFERENCES `provider` (`id`) ON DELETE CASCADE ON UPDATE CASCADE)

2. Успішна транзакція.

Транзакція складається з запитів на додавання таких же інгредієнтів з постачальниками "Astra " та "Aro" після створення у таблиці *provider* постачальника "Aro".

```
INSERT INTO confectionary.provider
VALUES (3, 'Aro', '2345768','nuts', 'aro@gmail.com');

START TRANSACTION;
INSERT INTO confectionary.ingredient VALUES
    (NULL, 'darkChocolate', 10.30, 1 , 1),
    (NULL, 'nut', 20.80, 10, 3);
COMMIT;
```

Результат успішного додавання інгредієнтів у таблицю :

```
SELECT * FROM confectionary.ingredient;
```

id	name	price_for_unit	quantity	provider_id
1	flour	89.90	10	2
2	egg	1.50	50	1
3	milk	20.50	10	2
4	sugar	30.00	40	2
5	chocolate	30.00	50	2
6	cacao	20.00	10	2
7	carrot	10.00	10	1
8	cherry	20.00	50	1
9	blueberry	10.00	100	1
10	apple	10.00	50	1
11	darkCho...	10.30	1	1
12	nut	20.80	10	3
NULL	NULL	NULL	NULL	NULL

3. Створимо дві точки збереження ,відредагуємо таблицю dish та відмінимо всі зміни. Перед початком виконаємо наступну команду, для того, щоб відключити автоматичнее збереження змін для поточної сесії зв’язку з сервером БД :

```
SET autocommit = 0;
```

Переглянемо таблицю dish :

```
SELECT * FROM confectionary.dish;
```

id	name	weight	price	kkal
1	tiramisu	200	150.00	460
2	cheesecake	500	200.00	1100
3	carrot cake	500	170.00	900
4	apple pie	600	150.00	920
5	oreo cheesecake	500	210.00	1000
6	blueberry cake	600	180.00	920
7	nutcake	700	190.00	950
8	cherry pie	600	140.00	910
NULL	NULL	NULL	NULL	NULL

Створимо точку збереження, використовуючинаступний запит:

```
SAVEPOINT SP1;
```

Тепер виконаємо наступні запити:

```
UPDATE confectionary.dish SET price = 500 WHERE kkal = 1100;
UPDATE confectionary.dish SET weight = '100' WHERE id = 2;
```

На даний момент таблиця містить такі записи:

id	name	weight	price	kkal
1	tiramisu	200	150.00	460
2	cheesecake	100	500.00	1100
3	carrot cake	500	170.00	900
4	apple pie	600	150.00	920
5	oreo cheesecake	500	210.00	1000
6	blueberry cake	600	180.00	920
7	nutcake	700	190.00	950
8	cherry pie	600	140.00	910
NULL	NULL	NULL	NULL	NULL

Створимо ще одну точку збереження :

```
SAVEPOINT SP2;
```

І виконаємо наступні запити:

```
DELETE FROM confectionary.dish WHERE name = 'apple pie' ;
DELETE FROM confectionary.dish WHERE id = 7 ;
DELETE FROM confectionary.dish WHERE id = 8;
```

Тепер таблиця містить такі записи:

id	name	weight	price	kkal
1	tiramisu	200	150.00	460
2	cheesecake	100	500.00	1100
3	carrot cake	500	170.00	900
5	oreo cheesecake	500	210.00	1000
6	blueberry cake	600	180.00	920
NULL	NULL	NULL	NULL	NULL

Тоді повернемося до точки збереження SP1 за допомогою команди:

```
ROLLBACK TO SP1;
```

Після виконання данного запиту, таблиця буде зберігати такі записи:

id	name	weight	price	kkal
1	tiramisu	200	150.00	460
2	cheesecake	500	200.00	1100
3	carrot cake	500	170.00	900
4	apple pie	600	150.00	920
5	oreo cheesecake	500	210.00	1000
6	blueberry cake	600	180.00	920
7	nutcake	700	190.00	950
8	cherry pie	600	140.00	910
NULL	NULL	NULL	NULL	NULL

Так повернулися до стану таблиці на момент створення точки збереження SP1. Тепер, коли нам більше не потрібні ці точки збереження, можемо їх звільнити:

```
RELEASE SAVEPOINT SP1;
```

Висновок

Під час виконання даної лабораторної роботи я навчилася використовувати механізм транзакцій у СУБД MySQL. Розробила SQL запити, які виконуються як єдине ціле в рамках однієї транзакції, використовуючи різні директиви для організації транзакцій.