МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Лабораторна робота №11

З дисципліни «Організація баз даних та знань**»**

Виконав:
студент групи КН-210
Бурак Марко

Львів – 2020

Тема: Розробка та застосування транзакцій.

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

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

Оголошує точку збереження всередині транзакції та задає її назву.

ROLLBACK TO [SAVEPOINT]

Відміняє результати виконання запитів, вказаних після даної точки збереження.

RELEASE SAVEPOINT

Видаляє точку збереження.

Хід роботи

1. В ході роботи, потрібно продемонструвати успішне і неуспішне виконання транзакції.
   Розроблю транзакцію, яка буде вносити дані в таблицю goods, тобто додавати товари. Транзакція буде відміняти всі зміни у таблицях при виникненні помилки чи іншої суперечливості.

   Буду виконувати транзакції до таблиці goods.
   У цій таблиці міститься 10 товарів, які поділяються на 2 підгрупи телефони та ноутбуки.

   Буду додавати дані, які містять id, назву товару, а також ключ, який підв’язаний до типу товару.

   У першому запиті додам 5 товарів, проте зроблю помилку у останньому запиті, зміню значення ключа на 3, якого не існує.

```
start transaction;
    insert into goods value(11,'google pixel',1);
    insert into goods value(12,'honor',1);
    insert into goods value(13,'huawei',1);
    insert into goods value(14,'dell',2);
    insert into goods value(15,'acer',3);
```

Всі 4 перші транзакції виконаються успішно, проте, у 5 виникне помилка.

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`store`.`goods`, CONSTRAINT `goods_type` FOREIGN KEY (`product_type_id`) REFERENCES `product_type` (`id`))

id	good_name	product_type_id
1	samsung	1
2	xiaomi	1
3	hp	2
4	lenovo	2
5	apple	1
6	meizu	1
7	msi	2
8	motorola	1
9	acer	2
10	nokia	1
11	google pixel	1
12	honor	1
13	huawei	1
14	dell	2
NULL	NULL	NULL

Щоб повернутись назад, та відмінити всі минулі інсерти потрібно застосувати команду ROLLBACK; Це я і зроблю.

```
rollback;
```

Ось результат виконання запиту до таблиці goods

id	good_name	product_type_id
1	samsung	1
2	xiaomi	1
3	hp	2
4	lenovo	2
5	apple	1
6	meizu	1
7	msi	2
8	motorola	1
9	acer	2
10	nokia	1
NULL	NULL	NULL

Зміни були скасовані.

Для того, щоб не виникало помилки при транзакції, я додам 3 елемент до таблиці видів, це будуть товари виду “tablet”.

	id	typ
	1	mobile
	2	laptop
	3	tablet
*	NULL	NULL

Тепер при виконанні транзакції все має працювати правильно.

```
1
2 •  start transaction;
3 •      insert into goods value(11,'google pixel',1);
4 •      insert into goods value(12,'honor',1);
5 •      insert into goods value(13,'huawei',1);
6 •      insert into goods value(14,'dell',2);
7 •      insert into goods value(15,'acer',3);
8 •  commit;
```

Помилок не було отримано, таблиця була змінена.

Результат запиту до таблиці goods.

	id	good_name	product_type_id
▶	1	samsung	1
	2	xiaomi	1
	3	hp	2
	4	lenovo	2
	5	apple	1
	6	meizu	1
	7	msi	2
	8	motorola	1
	9	acer	2
	10	nokia	1
	11	google pixel	1
	12	honor	1
	13	huawei	1
	14	dell	2
	15	acer	3
*	NULL	NULL	NULL

Висновок: На цій лабораторній роботі я ознайомився із механізмом транзакцій у СУБД MySQL.