

**Міністерство освіти і науки України  
Національний університет «Львівська політехніка»**

**Кафедра систем штучного інтелекту**



**Лабораторна робота №11  
на тему:  
«Розробка та застосування транзакцій»  
з курсу:  
«ОБДЗ»**

Виконала:  
ст. гр. КН-210  
Ямнюк Аліна  
Прийняла:  
Мельникова Н.І.

**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 відмінити 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) Продемонструвати неуспішне виконання транзакції.





Оскільки, в базі даних немає `id_client = 5`, то транзакція не виконається.

	 id_order ÷	 id_client ÷	 dateOfTheOrder ÷	 timeOfTheOrder ÷
1	1	1	2020-06-05	14:00:00
2	2	2	2020-06-06	12:00:00
3	3	3	2020-06-07	16:00:00

```
START TRANSACTION;
INSERT INTO beauty.orderOfTheClient (id_order, id_client, dateOfTheOrder, timeOfTheOrder)
VALUE ('4', '1', '20.06.05', '14:00');

INSERT INTO beauty.orderOfTheClient (id_order, id_client, dateOfTheOrder, timeOfTheOrder)
VALUE ('5', '2', '20.06.06', '12:00');

INSERT INTO beauty.orderOfTheClient (id_order, id_client, dateOfTheOrder, timeOfTheOrder)
VALUE ('6', '5', '20.06.07', '16:00');
```

	 id_order ÷	 id_client ÷	 dateOfTheOrder ÷	 timeOfTheOrder ÷
1	1	1	2020-06-05	14:00:00
2	2	2	2020-06-06	12:00:00
3	3	3	2020-06-07	16:00:00
4	4	1	2020-06-05	14:00:00
5	5	2	2020-06-06	12:00:00

### Результат:

```
beauty> INSERT INTO beauty.orderOfTheClient (id_order, id_client, dateOfTheOrder, timeOfTheOrder)
VALUE ('6', '5', '20.06.07', '16:00')
[2020-05-14 12:11:16] [23000][1452] Cannot add or update a child row: a foreign key constraint fails (`beauty`
[2020-05-14 12:11:16] [23000][1452] Cannot add or update a child row: a foreign key constraint fails (`beauty`
```

2) Продемонструвати успішне виконання транзакції.

Додаємо вірні дані про замовлення клієнта та виконуємо транзакцію.

```
START TRANSACTION;  
INSERT INTO beauty.orderOfTheClient (id_order, id_client, dateOfTheOrder, timeOfTheOrder)  
VALUES ('6', '1', '20.11.10', '15:00'),  
      ('7', '2', '20.07.06', '10:00');  
  
COMMIT;
```

Після виконання команди COMMIT (збереження змін, зроблених даною транзакцією), виконаємо команду, щоб переглянути збережений результат.

```
SELECT * FROM beauty.orderOfTheClient;
```

Результат:

	id_order	id_client	dateOfTheOrder	timeOfTheOrder
1	1	1	2020-06-05	14:00:00
2	2	2	2020-06-06	12:00:00
3	3	3	2020-06-07	16:00:00
4	4	1	2020-06-05	14:00:00
5	5	2	2020-06-06	12:00:00
6	6	1	2020-11-10	15:00:00
7	7	2	2020-07-06	10:00:00

**Висновок:** на цій лабораторній роботі я навчилась використовувати механізм транзакцій у СУБД MySQL. Розробила SQL запити, які виконуються як єдине ціле в рамках однієї транзакції.