Лабораторна робота №11
з курсу “ОБДЗ”
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.

В ході роботи, потрібно продемонструвати успішне і неуспішне виконання транзакції.

Розробимо транзакцію, яка буде вносити дані в таблицю Author і Role. Транзакція буде відміняти всі зміни у таблицях при виникненні помилки чи іншої суперечливості.

1. Відміна транзакції.

Транзакція складається з чотирьох запитів на додавання нових користувачів у групу "ugroup" та "sugroup3". При цьому, групи "sugroup3" з roleID=9 в базі даних не існує, а отже, транзакція не виконується.

```
START TRANSACTION;
 INSERT INTO mycms.author VALUE (NULL, NULL, 'user7', 'u7pass',
  '2008-04-16', 'user7@kml.com', NULL, 2);
 INSERT INTO mycms.author VALUE (NULL, NULL, 'user8', 'u8pass',
  '2009-04-22', 'user8@kml.com', NULL, 2);
 INSERT INTO mycms.author VALUE (NULL,NULL, 'superuser4',
  'suser4_pass', '2009-05-01', 'suser4@gl.com', NULL, 9);
 INSERT INTO mycms.author VALUE (NULL,NULL, 'suser5',
  'suuser5_pass', '2009-05-01', 'suser5@gl.com', NULL, 9);
COMMIT;
```

Відповідь сервера:

```
#1452 - Cannot add or update a child row: a foreign key
constraint fails (`mycms/author`, CONSTRAINT `author_role` FOREIGN
KEY (`roleID`) REFERENCES `role` (`roleID`) ON DELETE NO ACTION ON
UPDATE NO ACTION)
```

2. Успішна транзакція.

Транзакція складається з запитів на додавання тих самих користувачів у групу "ugroup" та "sugroup3" після створення у таблиці Role групи "sugroup3".

```
INSERT INTO role
 VALUE (NULL,'SUGroup3', 'read, write, changeown,
 groupadmin');

START TRANSACTION;
 INSERT INTO mycms.author VALUE (NULL, NULL, 'user7',
  'u7pass', '2008-04-16', 'user7@kml.com', NULL, 2);
 INSERT INTO mycms.author VALUE (NULL, NULL, 'user8',
 'u8pass',
  '2009-04-22', 'user8@ml.com', NULL, 2);
 INSERT INTO mycms.author VALUE (NULL,NULL,
  'superuser4', 'suser4_pass', '2009-05-01',
  'suser4@gl.com', NULL, 9);
 INSERT INTO mycms.author VALUE (NULL,NULL, 'suser5',
  'suuser5_pass', '2009-05-01', 'suser5@gl.com', NULL, 9);
COMMIT;
```

Результат успішного додавання чотирьох користувачів у таблицю показано нижче:

```
SELECT * FROM author LIMIT 9, 5;
```

authorID	name	login	password	created	email	profile	roleID
11	*NULL*	user9	[BLOB - 16 Bytes]	2009-02-19 00:00:00	user9@mail.com	*NULL*	3
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*	9
40	*NULL*	suser5	[BLOB - 12 Bytes]	2009-05-01 00:00:00	suser5@gl.com	*NULL*	9

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