Транзакції та блокування

І. Транзакції

Транзакція - послідовність операторів SQL, що виконуються як єдина операція, яка не перериває-ться іншими клієнтами. Поки відбувається робота з записами таблиці (їх оновлення або видалення), СУБД MySQL автоматично блокує доступ до них.

Підтримка транзакцій здійснюється тільки в таблицях BDB і InnoDB.

Транзакції об'єднують оператори в групу і гаран-тують, що всі операції всередині групи будуть вико-нані успішно. Якщо частина транзакції виконується зі збоєм, результати виконання всіх операторів транзакції до місця збою скасуються, приводячи базу даних до вигляду, в якому вона була до виконання транзакції.

За замовчуванням СУБД MySQL працює в режимі автоматичного завершення транзакцій, після виконання оператора оновлення даних, який модифікує таблицю, MySQL зберігає зміни на диску. Для об'єднання в транзакцію кількох операторів слід відключити цей режим. Це здійснюється за допомогою системної змінної ***AUTOCOMMIT***, значення якої встановлюється оператором ***SET AUTOCOMMIT = 0;***

Після відключення режиму автоматичного завершення транзакцій слід використовувати оператор ***COMMIT***, щоб зберігати зміни на диску, або ***ROLLBACK***, щоб скасувати зміни, виконані з моменту початку транзакції. Для включення режиму автоматичного завершення транзакцій, слід виконати оператор ***SET AUTOCOMMIT = 1.***

Для включення режиму автоматичного завершення транзакцій тільки для окремої послідовності операторів, використовується оператор ***START TRANSACTION***.

Після виконання оператора ***START TRANSACTION*** режим автоматичного завершення транзакцій залишається включеним до явного завершення транзакції за допомогою оператора ***COMMIT*** або відкату транзакції за допомогою ***ROLLBACK***.

Приклад механізму транзакцій.

UPDATE user_account SET allsum = allsum + 1000 WHERE id ='1';

UPDATE user_account SET allsum = allsum - 1000 WHERE id ='2';

Це переказ грошей з особового рахунку клієнта з номером 2 на особовий рахунок клієнта з номером 1. Якщо перший запит успішно виконався, а другий з якихось причин (помилка бази даних, помилка на сервері і т.д.) - ні. Отримуємо ситуацію, яка загрожує грошовими втратами.

Для уникнення цього потрібно, щоб обидва запити виконувалися як одне ціле. І якщо виникла помилка в одному запиті, не були виконані інші. Для цього служить ***механізм транзакцій.***

Оператор, що відкриває транзакцію: ***"START TRANSACTION;"***. Після правильного виконання всіх запитів транзакцію можна або завершити внісши всі зміни в базу даних - ***"COMMIT;",*** або відкотити, повернувши все в початковий стан – ***“ROLLBACK".***

Слід врахувати що:

1. Транзакційний механізм підтримують тільки InnoDB і BDB. Тому всі таблиці з якими хочете працювати через транзакції слід переконвертувати у відповідний тип.

2. За замовчуванням MySQL працює в режимі autocommit. Це означає, що результати виконання будь-якого SQL - оператора, що змінює дані, будуть відразу зберігатися.

Режим ***autocommit*** можна відключити командою ***SET AUTOCOMMIT = 0***. При відключеному режимі ***autocommit*** кожну транзакцію треба явно завершувати операторами ***COMMIT/ROLLBACK.***

Таким чином, для того, щоб реалізувати однократну транзакцію, яка вирішує поставлену на початку статті проблему, необхідно виконати такий код:

```
START   TRANSACTION;
UPDATE user_account
SET allsum = allsum + 1000 WHERE id =‘1’;
UPDATE user_account
SET allsum = allsum - 1000 WHERE id =‘2’;
COMMIT;
```

2. Блокування таблиць

Для таблиць типу MyISAM використання транзакцій недоступне. Проте їх можна емулювати при допомогою операторів LOCK TABLES та UNLOCK TABLES. Ці оператори блокують всю таблицю, ніхто не може працювати з таблицями до тих пір, поки вони залишаються заблокованими. Оператор LOCK TABLES виконує блокування таблиць, а UNLOCK TABLES знімає блокування.

Усі таблиці, заблоковані в поточному з'єднанні, розблокуються при повторному виклику оператора LOCK TABLES.

Оператори LOCK TABLES і UNLOCK TABLES мають синоніми LOCK TABLE і UNLOCK TABLE, відповідно

```
LOCK  TABLES catalogs WRITE;
INSERT INTO catalogs VALUES (NULL, ‘Принтер');
INSERT INTO catalogs VALUES(NULL, ‘Різне');
UNLOCK  TABLES;
```

Лістинг демонструє блокування таблиці ***catalogs*** на час додавання даних в базу даних. Після оператора LOCK TABLES записується ім'я блокованої таблиці, при знятті блокування за допомогою оператора UNLOCK TABLES вказання імені таблиці не потрібне. Основна причина застосування блокування таблиці за допомогою LOCK TABLES - це збільшення швидкості оновлення таблиць і додавання великих обсягів даних.

При використанні блокувань можна явно вказати тип блокування - на читання (READ) або на запис (WRITE).

```
LOCK TABLES catalogs WRITE;
INSERT INTO catalogs VALUES(NULL,'Периферія')
INSERT INTO catalogs VALUES(NULL,'Різне ');
UNLOCK TABLES catalogs;
LOCK TABLES catalogs READ;
INSERT INTO catalogs VALUES (NULL,'Периферія')
INSERT INTO catalogs VALUES(NULL,‘Різне ');
UNLOCK TABLES;
```

Різниця між блокуваннями (READ) і (WRITE) полягає в тому, що клієнт, який встановив блокування на читання, і інші клієнти можуть тільки читати з таблиці дані. При блокуванні на запис (WRITE) встановивший його клієнт може як вносити записи в таблицю, так і читати, в той час як доступ інших клієнтів до таблиці блокується. Причому всі інші клієнти очікують, коли блокування буде скасоване оператором UNLOCK TABLES.

Один оператор LOCK TABLES може блокувати декілька таблиць, рекомендується блокувати всі таблиці, які беруть участь у запитах всередині блокування.

```
LOCK TABLES catalogs WRITE, products WRITE;
SELECT * FROM catalogs, products
 WHERE
        products.id_catalog = catalogs.id_catalogs
INSERT INTO catalogs VALUES(NULL,‘Різне');
UNLOCK TABLES catalogs;
```