# Лабораторная работа 15. Транзакции MySQL

Цель работы: выполнение транзакций в MySQL.

Использован источник: http://webistore.ru/sql/tranzakcii-v-mysql/

Транзакция — это операция, состоящая из одного или нескольких запросов к базе данных. Суть транзакций — обеспечить корректное выполнение всех запросов в рамках одной транзакции, а так-же обеспечить механизм изоляции транзакций друг от друга для решения проблемы совместного доступа к данным.

Любая транзакция либо выполняется полностью, либо не выполняется вообще.

В транзакционной модели есть два фундаментальных понятия: COMMIT и ROLLBACK. COMMIT означает фиксацию всех изменений в транзакции. ROLLBACK означает отмену (откат) изменений, произошедших в транзакции.

При старте транзакции все последующие изменения сохраняются во временном хранилище. В случае выполнения COMMIT, все изменения, выполненные в рамках одной транзакции, сохранятся в физическую БД. В случае выполнения ROLLBACK произойдет откат и все изменения, выполненные в рамках этой транзакции, не сохранятся.

В MySQL транзакции поддерживаются только таблицами innoDB. Таблицы MyISAM транзакции не поддерживают. В innoDB по умолчанию включен autocommit, это значит, что по умолчанию каждый запрос эквивалентен одной транзакции.

Транзакция начинается со специального запроса «START TRANSACTION», либо «BEGIN». Чтобы закончить транзакцию, нужно либо зафиксировать изменения (запрос COMMIT), либо откатить их (запрос ROLLBACK).

Пример с COMMIT:

set autocommit=0; //отключаем autocommit

**Start transaction;** (также, можно написать **BEGIN**; )

…какие-то действий с БД (insert, update,delete…)

**commit; //Фиксация действий, запись их в физическую БД**

Пример с ROLLBACK:

set autocommit=0; //отключаем autocommit

**Start transaction;**

…какие-то действия с БД (insert, update,delete…)

**rollback; // отменяем серию действий, не производим запись в физическую БД**

В MySQL не существует механизма вложенных транзакций. Одно соединение с БД — одна транзакция. Новая транзакция в пределах одного соединения может начаться только после завершения предыдущей.

Для некоторых операторов нельзя выполнить откат с помощью ROLLBACK. Это операторы языка определения данных (Data Definition Language — DDL). Сюда входят запросы CREATE, ALTER, DROP, TRUNCATE, COMMENT, RENAME.

Следующие операторы неявно завершают транзакцию (как если бы перед их выпол­нением был выдан COMMIT):

* ALTER TABLE
* DROP DATABASE
* LOAD MASTER DATA
* SET AUTOCOMMIT = 1
* BEGIN
* DROP INDEX
* LOCK TABLES
* START TRANSACTION
* CREATE INDEX
* DROP TABLE
* RENAME TABLE
* TRUNCATE TABLE

Обратите внимание, что в случае SQL ошибки, транзакция сама по себе не откатится. Обычно ошибки обрабатываются уже с помощью sql wrapper’ов в самом приложении, таких как PHP PDO например.

Рассмотрим практический пример: есть 2 таблицы, пользователи — users и информация о пользователях — user\_info. Представим, что нужно либо выполнить 3 запроса к базе данных, либо не выполнять их вообще, так как иначе это приведет к сбоям в работе приложения.

**Задание 1.**

В MySQL **создайте** две таблицы, структура которых следует из следующего скрипта, и запустите его.

Start transaction;

use bd2;

INSERT INTO user (id, nik) VALUES (1, 'nikola');

INSERT INTO user\_info (id, id\_user, item\_name, item\_value) VALUES (1, 1, 'imya', 'Nikolay');

INSERT INTO user\_info (id, id\_user, item\_name, item\_value) VALUES (2, 1, 'vozr', '24');

commit;

Существуют проблемы параллельных транзакций. Рассмотрим пример. Представим, что во время выполнения этой транзакции, другой пользователь создал вторую параллельную транзакцию и сделал запрос SELECT \* FROM user после того, как в транзакции был выполнен первый запрос «INSERT INTO user (id, nik) VALUES (1, ‘nikola’)». Что увидит пользователь второй транзакции? Сможет ли он увидеть вставленную запись даже тогда, когда результаты первой транзакции еще не зафиксировались (не произошел COMMIT)? Или он сможет увидеть изменения только после того, как результаты первой транзакции будут зафиксированы? Оказывается имеют место быть оба варианта. Все зависит от уровня изоляции транзакции.

У транзакций есть 4 уровня изоляции:

* **0 — Чтение неподтверждённых данных (грязное чтение)** (Read Uncommitted, Dirty Read) — самый низкий уровень изоляции. При этом уровне возможно чтение незафиксированных изменений параллельных транзакций. В этом случае второй пользователь увидит вставленную запись из первой незафиксированной транзакции. Нет гарантии, что незафиксированная транзакция будет в любой момент откачена, поэтому такое чтение является потенциальным источником ошибок.
* **1 — Чтение подтверждённых данных** (Read Committed) — здесь возможно чтение данных только зафиксированных транзакций. Но на этом уровне существуют две проблемы. В этом режиме строки, которые участвуют в выборке в рамках транзакции, для других параллельных транзакций не блокируются, из этого вытекает проблема № 1: «Неповторяемое чтение» (non-repeatable read) — это ситуация, когда в рамках транзакции происходит несколько выборок (SELECT) по одним и тем же критериям, и между этими выборками совершается параллельная транзакция, которая изменяет данные, участвующие в этих выборках. Так как параллельная транзакция изменила данные, результат при следующей выборке по тем же критериям в первой транзакции будет другой.  Проблема № 2 — «Фантомное чтение» — этот случай рассмотрен ниже.
* **2 — Повторяемое чтение** (Repeatable Read, Snapshot) —  на этом уровне изоляции так же возможно чтение данных только зафиксированных транзакций. Так же на этом уровне отсутствует проблема «Неповторяемого чтения», то есть строки, которые участвуют в выборке в рамках транзакции, блокируются и не могут быть изменены другими параллельными транзакциями. Но таблицы целиком не блокируются. Из-за этого остается проблема «фантомного чтения». «Фантомное чтение» — это когда за время выполнения одной транзакции результат одних и тех же выборок может меняться по причине того, что блокируется не вся таблица, а только те строки, которые участвуют в выборке. Это означает, что параллельные транзакции могут вставлять строки в таблицу, в которой совершается выборка, поэтому два запроса SELECT \* FROM table могут дать разный результат в разное время при вставке данных параллельными транзакциями.
* **3 — Сериализуемый** (Serializable) — сериализуемые транзакции. Самый надежный уровень изоляции транзакций, но и при этом самый медленный. На этом уровне вообще отсутствуют какие либо проблемы параллельных транзакций, но за это придется платить быстродействием системы, а быстродействие в большинстве случаев крайне важно.

По умолчанию в MySQL установлен уровень изоляции № 2 (Repeatable Read). С первого раза может показаться, что самый лучший вариант № 3 — он самый надежный, но на практике можно испытать большие неудобства из-за очень медленной работы приложения. Помните, что многое зависит не от того, насколько хорош уровень изоляции транзакций в БД, а от того, как спроектировано конкретное приложение. При грамотном программировании, можно даже использовать самый низкий уровень изоляции транзакций — все зависит от особенностей структуры и грамотности разработки приложения. Но не нужно стремиться к самому низкому уровню изоляции — нет, просто если использовать не самый защищенный режим, следует помнить о проблемах параллельных транзакций.

SET TRANSACTION — этот оператор устанавливает уровень изоляции следующей транзакции, глобально либо только для текущего сеанса.

* SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
{ READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }

Существующие соединения не затрагиваются. Для выполнения этого оператора нужно иметь привилегию SUPER. Применение ключевого слова SESSION уста­навливает уровень изоляции по умолчанию всех будущих транзакций только для теку­щего сеанса.

**Задание 2.** Запустите оператор SET TRANSACTION по варианту.

Варианты уровня изоляции:

1. READ UNCOMMITTED
2. READ COMMITTED
3. REPEATABLE READ
4. SERIALIZABLE

Можно также установить начальный глобальный уровень изоляции для сервера mysqld, запустив его с опцией —transaction-isolation

**Контрольные вопросы:**

1. Что означают COMMIT и ROLLBACK?

2. Какие бывают у транзакций уровни изоляции?

3. Какие операторы неявно завершают транзакцию?