УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите теоретический материал к лабораторной работе, ответьте письменно на контрольные вопросы, выполните задание практической работы в соответствии с Вашим вариантом по списку в журнале.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: xvsviv@rambler.ru в трехдневный срок с момента получения задания.

При возникновении вопросов по приведенному материалу обращаться по следующим номерам телефонов:072-138-93-11.

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лабораторная работа. Управление транзакциями в СУБД MySQL

Цель: изучить инструкции выполняющие управление транзакциями в СУБД MySQL.

Теоретические сведения

Мультиверсионность (на примере MySQL)

Идея мультиверсионности (MVCC, Multi-Versioned Concurrency Control) как метода достижения изоляции основана на том факте, что процесс, выполняющий чтение данных (оператор SELECT) заинтересован не столько в их свежести, сколько в их целостности. Следовательно, можно избавиться от необходимости блокировок на чтение, вместо этого предоставляя читающим транзакциям КОПИЮ (снэпшот) последнего, возможно, не самого свежего, но целостного набора запрашиваемых данных. На уровне REPEATABLE READ каждой транзакции присваивается таймпоинт, после чего она читает свой личный снэпшот целостных данных из журнала транзакций. Добавления, изменения или удаления данных, произведенные параллельными транзакциями с более поздними таймпоинтами, не видны всем операторам текущей транзакции.

На уровне READ COMMITTED каждый оператор SELECT работает со наиболее свежим на момент его старта целостным снэпшотом набора данных.

При этом если какая-то другая транзакция после этого изменяет этот набор данных и фиксируется, следующий оператор SELECT получит его новую версию в качестве целостного снэпшота.

Следует понимать, что мультиверсионность проявляется только при чтении данных (т.е. отсутствуют S-блокировки, не блокируются зависимости W-R и R-W). При этом запись осуществляется непосредственно в БД, а не в снэпшот, и если возникает зависимость W-W, используются блокировки.

На уровне READ UNCOMMITTED мультиверсионность не используется: каждый оператор SELECT читает последнюю (возможно, «грязную») версию данных непосредственно из БД.

Уровень SERIALIZABLE в MySQL эквивалентен уровню REPEATABLE READ с неявным принудительным включением разделяемых блокировок в операторах SELECT.

Принудительное включение блокировок

В MySQL есть возможность принудительного включения блокировок на чтение. Это достигается конструкцией SELECT … LOCK IN SHARE MODE.

При этом вне зависимости от уровня изоляции, на существующие в таблице строки, удовлетворяющие условию WHERE оператора SELECT текущей транзакции, накладываются разделяемые блокировки, что:

– не позволяет текущей транзакции читать данные, измененные другой транзакцией, которая еще не была зафиксирована («грязное чтение», W-R);

– не позволяет другим транзакциям изменять данные, прочитанные текущей транзакцией (неповторяемое чтение, R-W).

Разница между уровнями изоляции проявляется лишь в способах наложения блокировок. На уровнях READ COMMITTED и READ UNCOMMITTED блокируются только существующие строки, удовлетворяющие условию фильтрации. На уровне REPEATABLE READ используются диапазонные блокировки, что исключает возможность появления фантомов.

Все вышеописанное относится только к движку InnoDB, как единственному ACID-совместимому среди движков MySQL. Начиная с версии MySQL 5.5 (2010 г.) InnoDB является движком по умолчанию для таблиц в MySQL.

Методические указания

Задания выполняются с использованием сервера MySQL версии 5.5 или выше. В качестве клиента можно использовать либо поставляемую с сервером утилиту командной строки mysql, либо официальный инструментарий MySQL Workbench, либо стороннее ПО, например, dbForge Studio for MySQL или HeidiSQL. При использовании ПО с графическим интерфейсом пользователя задания выполняются в окне редактора SQL-запросов.

В лабораторной работе необходимо отключить интерактивный режим выполнения запросов, в котором работают все клиенты MySQL. В этом режиме каждый оператор SQL, обрабатываемый клиентом, облекается в отдельную транзакцию, которая автоматически фиксируется после его выполнения (т.н. режим автофискации, autocommit). В лабораторной работе исследуются методы разрешения зависимостей между параллельно (т.е. одновременно) выполняющимися транзакциями, обращающимися к одним и тем же данным, поэтому транзакции должны состоять из нескольких операторов и выполняться продолжительное время.

В интерфейсе командной строки или редакторе SQL начать транзакцию без автофиксации можно выполнив команду BEGIN. В MySQL Workbench можно достичь того же, отключив режим автофиксации транзакций (Auto-Commit Transactions), после чего все выполняемые операторы будут объединены в одну транзакцию, а также станут доступны команды фиксации и отката текущей транзакции (см. рисунок 1).

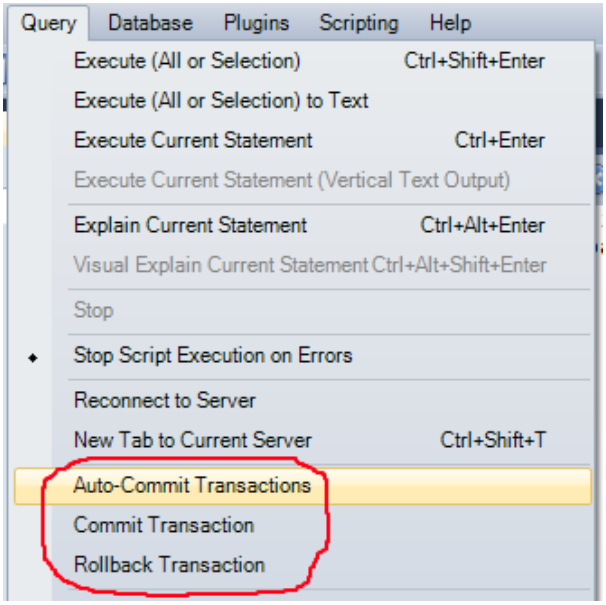

Рисунок 1 – Отключение режима автофиксации транзакций

Те же элементы управления продублированы на панели инструментов (см. рисунок 2).

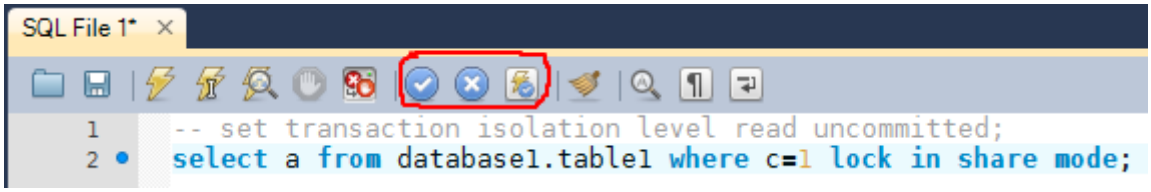

Рисунок 2 – Элементы управления на панели инструментов

В dbForge Studio for MySQL по умолчанию также включен режим автофиксации. Можно отключив его, нажав на кнопку начала транзакции или выбрав соответствующий элемент контекстного меню соединения (см. рисунок 3).

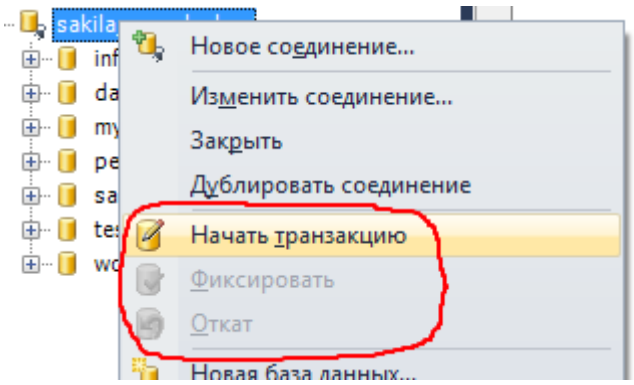

Рисунок 3 – Режим автофиксации по умолчанию в dbForge Studio for MySQL

После этого становятся доступны команды фиксации и отката текущей транзакции (см. рисунок 4).

Рисунок 4 – Команды фиксации и отката текущей транзакции

Для заданий задан уровень изоляции (RU – READ UNCOMMITTED, RC – READ COMMITTED, RR – REPEATABLE READ). Необходимо установить в подключении соответствующий уровень изоляции:

SET TRANSACTION ISOLATION LEVEL уровень; – для следующей транзакции

SET SESSION TRANSACTION ISOLATION LEVEL уровень; – для всех последующих транзакций данного подключения.

Затем необходимо запустить второе подключение к той же БД. Уровень изоляции можно оставить по умолчанию (в InnoDB это REPEATABLE READ, если не изменялись настройки).

Параллельно запускаются две транзакции из двух открытых подключений. Первая транзакция T1 (из подключения с уровнем изоляции согласно варианту) производит операции с данными, после чего вторая транзакция T2 также работает с этими данными и так далее. Список действий

должен полностью представлять реакцию системы на зависимости между транзакциями.

Для каждого варианта необходимо проанализировать:

1. Зависимость потерянного обновления

T1	**T2**
read(x)	
	write(x)
write(x)	

2. Зависимость грязного чтения

T1	**T2**
	write(x)
read(x)	
	write(x)

3. Зависимость неповторяемого чтения

T1	**T2**
read(x)	
	write(x)
read(x)	

4. Зависимость фантомов

T1	**T2**
filter(t)	
	add(t)
filter(t)	

Задания MVCC связаны с исследованием механизма мультиверсионности для движка InnoDB. При этом транзакцией T1 производятся целостные неблокирующие чтения (в InnoDB – обычный оператор SELECT).

Задания LOCK связаны с исследованием реализации принудительных разделяемых блокировок в движке InnoDB. При этом транзакцией T1 производятся блокирующие чтения (в InnoDB – оператор SELECT … LOCK IN SHARE MODE).

Пример 1

MVCC, уровень READ UNCOMMITTED, зависимость потерянного обновления

T1	**T2**
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED; BEGIN; SELECT * FROM t WHERE i = 3;	

+——+ | i | +——+ | 3 | +——+ 1 row in set (0.00 sec)	
	BEGIN; UPDATE t SET i=3 WHERE i=2; Query OK, 1 row affected (0.05 sec) Rows matched: 1 Changed: 1 Warnings: 0
UPDATE t SET i=5 WHERE i=2; ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction	
	COMMIT;

Вывод: потерянное обновление не допускается, транзакция T1, пытающаяся изменить данные, на которые наложена X-блокировка транзакцией T2, ждет, а затем завершается по таймауту.

Пример 2

LOCK, уровень READ COMMITTED, зависимость фантомов

T1	**T2**
SET TRANSACTION ISOLATION LEVEL READ COMMITTED; BEGIN; SELECT * FROM t WHERE i = 3 LOCK IN SHARE MODE; +——+ | i | +——+ | 3 | +——+ 1 row in set (0.00 sec)	
	BEGIN; INSERT INTO t VALUES (3); Query OK, 1 row affected (0.00 sec)
SELECT * FROM t WHERE i = 3 LOCK IN SHARE MODE;	

ожидание...	
	COMMIT;
+——+ | i | +——+ | 3 | | 3 | +——+ 2 rows in set (0.00 sec)	

Вывод: фантомы допускаются, повторное выполнение запроса транзакцией T1 получает больший набор строк.

Индивидуальные варианты заданий

Номер варианта вычисляется как ((N - 1) mod 6) + 1, где N – порядковый номер студента в учебном журнале.

Например, индивидуальный номер N = 35. Тогда номер варианта ((N - 1) mod 6) + 1 = ((35 – 1) mod 6) + 1 = (34 mod 6) + 1 = 4 + 1 = 5

Вариант	RU MVCC	RC MVCC	RR MVCC	RC LOCK	RR LOCK
1	+			+	
2		+			+
3			+	+	
4	+				+
5		+		+	
6			+		+

План выполнения

Используя ранее созданную базу данных в MySQL. В двух параллельно запущенных сеансах выполнить набор операторов SQL, позволяющий изучить взаимодействие одновременно выполняющихся транзакций при доступе к одним и тем же данным на заданных в индивидуальном варианте уровнях изоляции.

Содержание отчета

Титульный лист. Цель работы. Индивидуальный вариант задания. Листинги скриптов SQL пар транзакций, анализирующих наличие/отсутствие

проблем параллельного выполнения транзакций на разных уровнях изоляции с использованием разделяемых блокировок или мультиверсионностью. Выводы для каждого из листингов.

Вспомогательные материалы

Транзакционная модель InnoDB

Вопросы к защите

- Что такое транзакция?
- Что такое ACID-свойства?
- Какие существуют методы достижения изоляции?
- Опишите различия между уровнями изоляции X и Y.
- В чем заключается принцип мультиверсионности?
- Объясните разницу между эксклюзивными и разделяемыми блокировками.
- Поясните листинг A.