Транзакции

Транзакция

- Транзакция — это архив для запросов к базе. Он защищает ваши данные благодаря принципу «всё, или ничего». Или выполнены все запросы, которые разработчик упаковал в одну транзакцию, или ни один.

- Пример с банком
- Пример с бухгалтерией

Транзакция

- Последовательность операций, производимых над базой данных и переводящих базу данных из одного согласованного состояния в другое согласованное состояние.
- Некоторое неделимое действие над базой данных, осмысленное с точки зрения пользователя.
- Логическая единица работы системы.

Отправка транзакции

- Чтобы обратиться к базе данных, сначала надо открыть соединение с ней. Это называется коннект.
- Чтобы сгруппировать запросы в одну атомарную пачку, используем транзакцию. Транзакцию надо:
  - Открыть
  - Выполнить все запросы внутри
  - Закрыть
- После закрытия транзакции, соединение освободилось и можно отправлять следующую транзакцию.

Пример транзакции

```
START TRANSACTION;

SELECT total FROM accounts WHERE user_id = 2;
UPDATE accounts SET total = total - 3000 WHERE user_id = 2;

UPDATE accounts SET total = total + 3000 WHERE user_id = 0;

COMMIT;
```

Концепции ACID

- Atomicity — Атомарность
- Consistency — Согласованность
- Isolation — Изолированность
- Durability — Надежность

Atomicity (Атомарность)

- Атомарность гарантирует, что никакая транзакция не будет зафиксирована в системе частично. Когда транзакции атомарны, не существует такого понятия, как частично выполненная транзакция. Атомарность гарантирует, что будут либо выполнены все подоперации транзакции, либо не выполнено ни одной.

Atomicity (Атомарность)

- Каждая транзакция представляет собой единицу работы.
- Она не может быть разбита на меньшие части.
- Выполняются либо все действия, определенные в данной транзакции, либо не выполняется ни одно из них.

Consistency (Согласованность)

- Согласованность означает, что любая завершённая транзакция (транзакция, которая достигла завершения транзакции – end of transaction) фиксирует только допустимые результаты. При выполнении принципа согласованности база данных должна всегда переходить из одного непротиворечивого состояния в другое непротиворечивое состояние. Другими словами, каждая успешная транзакция по определению фиксирует только допустимые результаты.

Consistency (Согласованность)

- Свойство согласованности гарантирует, что по мере выполнения транзакций данные переходят из одного согласованного состояния в другое — транзакция не разрушает взаимной согласованности данных.
- Для поддержания согласованности данных в процессе транзакции применяются все правила, проверки, ограничения и триггеры.

Isolation (Изолированность)

- Изолированность - транзакция должна быть изолирована от других, т.е. её результат не должен зависеть от выполнения других параллельных транзакций. Изолированность - требование дорогое, поэтому в реальных БД существуют режимы, не полностью изолирующие транзакцию (уровни изолированности).

Durability (Надежность)

- Сохраняемость гарантирует, что изменения, внесенные в ходе транзакции, будучи зафиксированными, становятся постоянными. Это означает, что изменения должны быть записаны так, чтобы данные не могли быть потеряны в случае сбоя системы.

Согласованность в базах данных

- База данных находится в согласованном состоянии, если для этого состояния выполнены все ограничения целостности.
- Ограничение целостности - это некоторое утверждение, которое может быть истинным или ложным в зависимости от состояния базы данных.
- Конкурентное использование ресурсов корректными программами может привести к некорректным результатам.

Управление транзакциями

- Для управления транзакциями используются следующие команды:
  - COMMIT - сохраняет изменения
  - ROLLBACK - откатывает (отменяет) изменения
  - SAVEPOINT - создаёт точку к которой группа транзакций может откатиться
  - SET TRANSACTION - размещает имя транзакции
- Команды управление транзакциями используются только для: INSERT, UPDATE, DELETE.

Rollback

- Применяется для того, чтобы:
  - Отменить все изменения, внесённые начиная с момента начала транзакции или с какой-то точки сохранения (`SAVEPOINT`)
  - Очистить все точки сохранения данной транзакции
  - Завершить транзакцию
  - Освободить все блокировки данной транзакции

Пример rollback

```
START TRANSACTION;

SELECT total FROM accounts WHERE user_id = 2;
UPDATE accounts SET total = total - 3000 WHERE user_id = 2;

UPDATE accounts SET total = total + 3000 WHERE user_id = 0;

ROLLBACK;
```

SAVEPOINT и ROLLBACK TO SAVEPOINT

- Точка сохранения представляет собой место в последовательности событий транзакции, которое может выступать промежуточной точкой восстановления. Откат транзакции может быть выполнен не к началу транзакции, а к точке сохранения. Для работы с точками сохранения предназначены два оператора:
- `SAVEPOINT` - создает точку сохранения
- `ROLLBACK TO SAVEPOINT` - позволяет откатиться к одной из точек сохранения

Пример SAVEPOINT

```
START TRANSACTION;

SELECT total FROM accounts WHERE user_id = 2;
SAVEPOINT accounts_2;

UPDATE accounts SET total = total - 2000 WHERE user_id = 2;

ROLLBACK TO SAVEPOINT accounts_2;
```

Несколько точек сохранения

- Допускается создание нескольких точек сохранения. Если текущая транзакция имеет точку сохранения с таким же именем, старая точка удаляется и устанавливается новая. Все точки сохранения транзакций удаляются, если выполняется оператор `COMMIT` или `ROLLBACK` без указания имени точки сохранения.

Уровни изоляции

- Стандарт SQL определяет четыре уровня изоляции с конкретными правилами, устанавливающими, какие изменения видны внутри и вне транзакции, а какие нет:
  - `READ UNCOMMITTED`
  - `READ COMMITTED`
  - `REPEATABLE READ`
  - `SERIALIZABLE`

READ UNCOMMITTED

- Используется редко, поскольку его производительность не намного выше, чем у других. На этом уровне вы видите промежуточные результаты чужих транзакций, т.е. осуществляете грязное чтение.

READ COMMITTED

- Подразумевает, что транзакция увидит только те изменения, которые были уже зафиксированы другими транзакциями к моменту ее начала. Произведенные ею изменения останутся невидимыми для других транзакций, пока она не будет зафиксирована. На этом уровне возможен феномен невоспроизводимого чтения. Это означает, что вы можете выполнить одну и ту же команду дважды и получить различный результат.

REPEATABLE READ

- Этот уровень изоляции установлен по умолчанию. Он гарантирует, что любые строки, которые считываются в контексте транзакции, будут выглядеть такими же при последовательных операциях чтения в пределах одной и той же транзакции, однако теоретически на этом уровне возможен феномен фантомного чтения (phantom reads). Он возникает в случае, если вы выбираете некоторый диапазон строк, затем другая транзакция вставляет новую строку в этот диапазон, после чего вы выбираете тот же диапазон снова. В результате вы увидите новую фантомную строку.

SERIALIZABLE

- Самый высокий уровень изоляции, решает проблему фантомного чтения, заставляя транзакции выполняться в таком порядке, чтобы исключить возможность конфликта. Уровень SERIALIZABLE блокирует каждую строку, которую транзакция читает. На этом уровне может возникать множество задержек и конфликтов при блокировках. На практике данный уровень изоляции применяется достаточно редко.

Изменение уровня изоляции

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Проблемы параллельного доступа

- Потерянное обновление (lost update)
- Грязное чтение (dirty read)
- Неповторяющееся чтение (non-repeatable read)
- Фантомное чтение (phantom reads)

Потерянное обновление

- Эффект проявляется при одновременном изменении одного блока данных разными транзакциями. Причём одно из изменений может теряться.
- Две транзакции выполняют одновременно UPDATE для одной и той же строки, и изменения, сделанные одной транзакцией, затираются другой.

Потерянное обновление

Транзакция 1	Транзакция 2
SELECT f2 FROM tbl1 WHERE f1=1;	SELECT f2 FROM tbl1 WHERE f1=1;
UPDATE tbl1 SET f2=20 WHERE f1=1;	
	UPDATE tbl1 SET f2=25 WHERE f1=1;

Грязное чтение

- Это такое чтение, при котором могут быть считаны добавленные или изменённые данные из другой транзакции, которая впоследствии не подтвердится (откатится).

Грязное чтение

Транзакция 1	Транзакция 2
SELECT f2 FROM tbl1 WHERE f1=1;	
UPDATE tbl1 SET f2=f2+1 WHERE f1=1;	
	SELECT f2 FROM tbl1 WHERE f1=1;
ROLLBACK WORK;	

Неповторяющееся чтение

- Проявляется, когда при повторном чтении в рамках одной транзакции, ранее прочитанные данные, оказываются изменёнными.

Неповторяющееся чтение

Транзакция 1	Транзакция 2
SELECT f2 FROM tbl1 WHERE f1=1;	SELECT f2 FROM tbl1 WHERE f1=1;
UPDATE tbl1 SET f2=f2+1 WHERE f1=1;	
COMMIT;	
	SELECT f2 FROM tbl1 WHERE f1=1;

Фантомное чтение

- Можно наблюдать, когда одна транзакция в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же критериям. При этом другая транзакция в интервалах между этими выборками добавляет или удаляет строки, или изменяет столбцы некоторых строк, используемых в критериях выборки первой транзакции, и успешно заканчивается. В результате получится, что одни и те же выборки в первой транзакции дают разные множества строк.

Фантомное чтение

Транзакция 1	Транзакция 2
	SELECT SUM(f2) FROM tbl1;
INSERT INTO tbl1 (f1,f2) VALUES (15,20);	
	SELECT SUM(f2) FROM tbl1;

Уровень изоляции	Потерянное обновление	Грязное чтение	Неповторяющееся чтение	Фантомное чтение
Не зафиксированные операции чтения	Нет	Да	Да	Да
Зафиксированные операции чтения	Нет	Нет	Да	Да
Повторяющиеся операции чтения	Нет	Нет	Нет	Да
Сериализуемый	Нет	Нет	Нет	Нет