Транзакции

Под транзакцией понимается неделимая с точки зрения воздействия на БД последовательность операторов манипулирования данными

Основное назначение транзакций в базе данных — переводить ее из одного согласованного состояния в другое

Транзакции также являются *единицами восстановления* данных после сбоев

Транзакции позволяют реализовывать *откладываемые ограничения целостности.*

Такие ограничения проверяются в конце транзакции, в случае их выполнения транзакция завершится оператором COMMIT.

Если ограничения будут нарушены, транзакция может быть откачена оператором ROLLBACK.

Откат транзакции

- В случае отката транзакции аннулируются все изменения, произведенные в ее рамках
- Проблемы
  - Как это реализуется?
  - Кто должен выдавать команду ROLLBACK?
  - Возможны ли вложенные транзакции?

Транзакции в многопользовательских системах

Транзакция рассматривается как последовательность элементарных атомарных операций

Элементарные операции различных транзакций могут выполняться в произвольной очередности

Определения

Набор из нескольких транзакций, элементарные операции которых чередуются друг с другом, называется ***смесью транзакций***

Последовательность, в которой выполняются элементарные операции заданного набора транзакций, называется ***графиком запуска*** набора транзакций

Если транзакции выполняются одна за другой, то график запуска называется ***последовательным***

Транзакции называются ***конкурирующими***, если они пересекаются по времени и обращаются к одним и тем же данным

Транзакция A

1. Читать из X
2. Уменьшить на 2
3. Записать в X

Транзакция B

1. Читать из X
2. Увеличить на 3
3. Записать в X

График 1

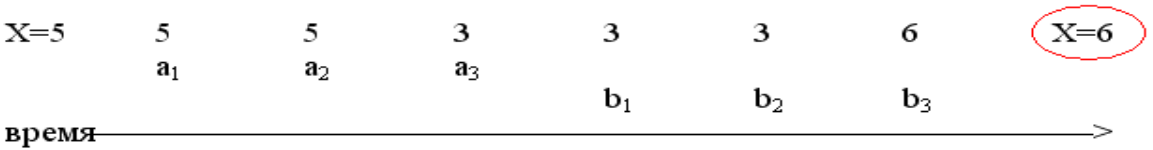


График 2

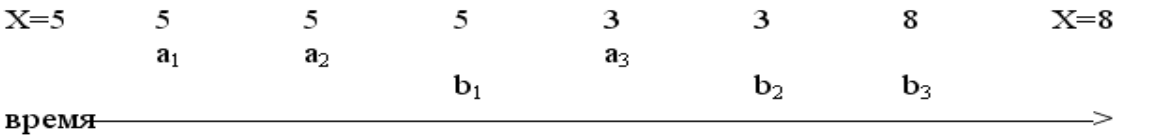
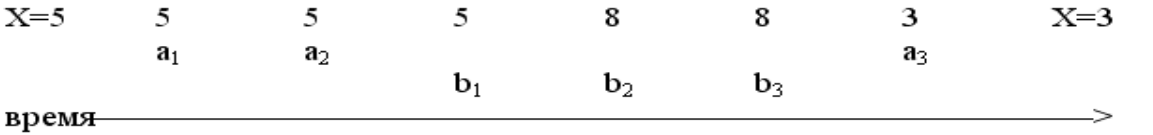


График 3



Определение

План (график) выполнения набора транзакций называется ***сериальным***, если результат совместного выполнения транзакций эквивалентен результату некоторого последовательного графика выполнения этих же транзакций.

Сериальный план гарантирует предсказуемость результата

Требования к транзакциям (ACID)

Неделимость (Atomicity). Транзакция либо выполняется, либо не выполняется полностью

Согласованность (Consistency). Транзакция переводит базу данных из одного согласованного состояния в другое

Изолированность (Isolation). Результаты транзакции становятся доступны для других транзакций только после ее фиксации

Продолжительность (Durability). После фиксации транзакции изменения становятся постоянными, пока не будут изменены другими транзакциями

Конфликты между транзакциями

Утраченные изменения (*конфликт типа W-W)*

Транзакция A	X	Транзакция B
R(x)	X0	
		R(x)
W(x)	Xb	
COMMIT	Xb	
	Xa	W(x)
		COMMIT

Утраченные (не восстановимые) изменения (*конфликт типа W-W)*

Транзакция A	X	Транзакция B
R(x_0)	x_0	
		R(x_0)
W(x_a)	x_a	
COMMIT	x_a	
	x_b	W(x_b)
	x_0	ROLLBACK

Незафиксированные зависимости (*конфликт типа W-R*) *Чтение «грязных» данных*

Транзакция A	X	Транзакция B
	x_0	$R(x_0)$
	x_b	$W(x_b)$
$R(x_b)$		
обработка (x_b)	x_b	
	x_0	ROLLBACK
COMMIT		

Несовместный анализ (*конфликт типа R-W)*

Случай 1
Неповторяемое считывание

Транзакция A	X	Транзакция B
$R(x_0)$	x_0	
		$R(x_0)$
	x_0	$W(x_0)$
		COMMIT
$R(x_0)$		
COMMIT		

Случай 2
Фиктивные элементы (фантомы)

Пишущая транзакция

```
insert into T(Id, val1)
          values(1000,1);
commit;
```

Читающая транзакция

```
select count(*) from T;



select count(*) from T;
commit;
```

Случай 3
Аномалия сериализации

Пишущая транзакция 1	Пишущая транзакция 2	X	Y	Читающая транзакция
R(x)		X_0		
R(y)			Y_0	
	R(y)			
	W(y)		Y_2	
	COMMIT			R(x),R(y) { X_0, Y_2}
W(x)		X_1		
COMMIT				

Изоляция транзакций

- СУБД должна иметь механизмы, для обеспечения требований ACID
- Эти механизмы могут обеспечивать разную степень (уровень) изоляции транзакций
- СУБД должна всегда обеспечивать недопустимость конфликта W-W. **Недопустимы** утраченные изменения

Изоляция транзакций

Стандартом SQL предусматриваются четыре уровня изоляции транзакций:

- *READ UNCOMMITTED* – чтение незафиксированных данных;
- *READ COMMITTED* – чтение зафиксированных данных;
- *REPEATABLE READ* – повторяющееся чтение;
- *SERIALIZABLE* – последовательное выполнение.

Уровни изоляции транзакций

Уровень изоляции	Утраченные изменения	«Грязное» чтение	Неповторяемое чтение	Фантомы	Аномалия сериализации
Read uncommitted	нет	да	да	да	да
Read committed	нет	нет	да	да	да
Repeatable read	нет	нет	нет	да	да
Serializable	нет	нет	нет	нет	нет

Механизмы

•обеспечить, чтобы конкурирующие транзакции выполнялись в разное время

- *Блокировки*
- *Временные метки*

• обеспечить, чтобы конкурирующие транзакции работали с разными версиями данных

Блокировки

- Монопольные (X) eXclusive locks
- Разделяемые (S) Shared locks

Транзакция A наложила	Транзакция B пытается наложить	
	S-блокировку	X-блокировку
S-блокировку	Да	НЕТ (Конфликт R-W)
X-блокировку	НЕТ (Конфликт W-R)	НЕТ (Конфликт W-W)

Что является объектом блокировки?

Логические

- Поле данных
- Кортеж
- Группа кортежей
- Таблица
- Группа таблиц
- База данных

Физические

- Блок
- Сектор
- Файл
- Группа файлов

Правила

- Перед чтением накладывается S блокировка
- Перед изменением накладывается X блокировка
- Если блокировка отвергается, транзакция переходит в состояние ожидания
- Двухфазный протокол синхронизационных захватов (2PL) – сначала наложить слабую блокировку S, если удалось – пытаться наложить X блокировку

Решение проблем с помощью блокировок

- Потеря результатов – да, *но возможны тупиковые ситуации*
- Чтение «грязных данных» - да
- Неповторяемое считывание – да
- «Фантомы» - нет, *но возможно решение с помощью более сложных механизмов блокировки*
- Аномалия сериализации– да, *но возможны тупиковые ситуации*

Возникновение тупика



Решение проблем тупиков

- Транзакция может задать максимально возможное время ожидания и выполнить откат, если оно превышено
- СУБД может обнаружить тупик и принести одну из транзакций в жертву
  - СУБД постоянно строит граф ожидания процессов. Если она видит замкнутый цикл (А -> Б -> А), она фиксирует тупик
  - СУБД принудительно прерывает одну из транзакций (делает ей ROLLBACK) и возвращает приложению ошибку. Вторая транзакция успешно завершается, а прерванную транзакцию приложение должно повторить (retry) заново.

Метод временных меток

- «Оптимистические» блокировки
- Используется, если вероятность возникновения конфликтов невелика
- Каждая транзакция получает временную метку (момент начала)
- Если транзакция блокирует объект, она помечает его своей меткой

Метод временных меток

- Прежде, чем заблокировать объект, транзакция анализирует его временную метку и конфликтность ситуации
- При отсутствии конфликта объект помечается наименьшей временной меткой и транзакции продолжают работу
- Если обнаружен конфликт, то более старая транзакция продолжит работу, пометив своим временем объект, а более молодая будет откачена и автоматически стартует вновь

Метод временных меток

- Преимущества
  - Отсутствие взаимных блокировок (deadlocks), так как транзакции не ожидают освобождения ресурсов друг другом
  - Хорошая производительность в системах с малым числом конфликтов между транзакциями

- Недостатки
  - Высокий риск каскадных откатов или массовых перезапусков транзакций при высокой конкуренции за одни и те же данные
  - Накладные расходы на хранение и обновление меток для каждого объекта данных

Версии данных
MultiVersion Concurrency Control

- Отказ от блокировок для транзакций, читающих данные
- Транзакциям, читающим данные, представляется версия данных, существовавшая в тот момент, когда читающая транзакция начала свою работу
- Для транзакций, меняющих данные, по-прежнему используется механизм блокировок

MultiVersion Concurrency Control

Читающая транзакция ищет среди версий данных те, которые существовали на момент ее старта, т.е. помеченные системным номером меньшим, чем у нее самой

Это гарантирует, что транзакция не увидит изменений, произошедших в данных, начиная с того момента как она стартовала

Если транзакция меняет данные (фиксирует изменение), то она помечает эти данные своим системным номером, не уничтожая старые версии данных, помеченные другими системными номерами

MultiVersion Concurrency Control

При использовании MVCC постоянно накапливаются устаревшие версии строк

От СУБД требуется следить за этими версиями и, как только они становятся неактуальными, предпринимать меры по их удалению

Проблема тупиков по-прежнему существует для конфликта W-W

Принципы изоляции в MVCC

- Никакая транзакция не может видеть неподтвержденные данные от других транзакций
- Самый низкий уровень изоляции позволяет транзакции видеть все подтвержденные изменения базы данных
- Самый высокий уровень изоляции позволяет гарантировать целостность на уровне всей базы данных для данных, с которыми работает транзакция

Транзакции в PostgreSQL

- Ни какая операция не может быть выполнена вне транзакции
- Перед началом новой транзакции предыдущая должна быть явно завершена операцией COMMIT или ROLLBACK
- Существует режим AUTOCOMMIT
- В начале новой транзакции устанавливается её уровень изоляции
- Если не указано явно, устанавливается уровень изоляции по умолчанию

Транзакции в PostgreSQL

- Команды начала транзакции

BEGIN [WORK | TRANSACTION] [*режим транзакции*]

START TRANSACTION [*режим транзакции*]

- Режим транзакции

ISOLATION LEVEL

{ SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
READ WRITE | READ ONLY

[[NOT] DEFERRABLE]

Транзакции в PostgreSQL

Уровень изоляции	Утраченные изменения	«Грязное» чтение	Неповторяемое чтение	Фантомы	Аномалия сериализации
Read Uncommitted	Работает как Read Committed				
Read Committed	нет	нет	да	да	да
Repeatable Read	нет	нет	нет	нет	да
Serializable	нет	нет	нет	нет	нет

Read Committed

- Транзакция читает данные из снимка базы данных в момент начала выполнения SQL запроса
- В транзакции видны все изменения, произведенные в ней
- В транзакции могут быть видны зафиксированные в других транзакциях изменения данных
- Если транзакция попытается изменить данные, которые были изменены и в другой параллельной транзакции она будет приостановлена до завершения этой параллельной транзакции
- После завершения параллельной транзакции текущая продолжит работу со старой или новой версией данных

Repeatable Read

- В транзакции видны только те данные, которые были зафиксированы до начала транзакции, но не видны незафиксированные данные и изменения, произведённые другими параллельными транзакциями
- Если транзакция попытается изменить данные, которые были изменены и в другой параллельной транзакции она будет приостановлена до завершения этой параллельной транзакции
- Если параллельная транзакция завершится откатом, текущая продолжит работу
- Если параллельная транзакция завершится фиксацией, текущая будет откачена с сообщением о невозможности сериализации

Serializable

- На этом уровне модели руется последовательное выполнение всех зафиксированных транзакций, как если бы транзакции выполнялись одна за другой, последовательно, а не параллельно
- Фактически этот режим изоляции работает так же, как и Repeatable Read, только он дополнительно отслеживает условия, при которых результат параллельно выполняемых сериализуемых транзакций может не согласовываться с результатом этих же транзакций, выполняемых по очереди

В дополнение

- READ WRITE | READ ONLY
- SERIALIZABLE READ ONLY DEFERRABLE
- Явные блокировки

SELECT FOR UPDATE

LOCK TABLE