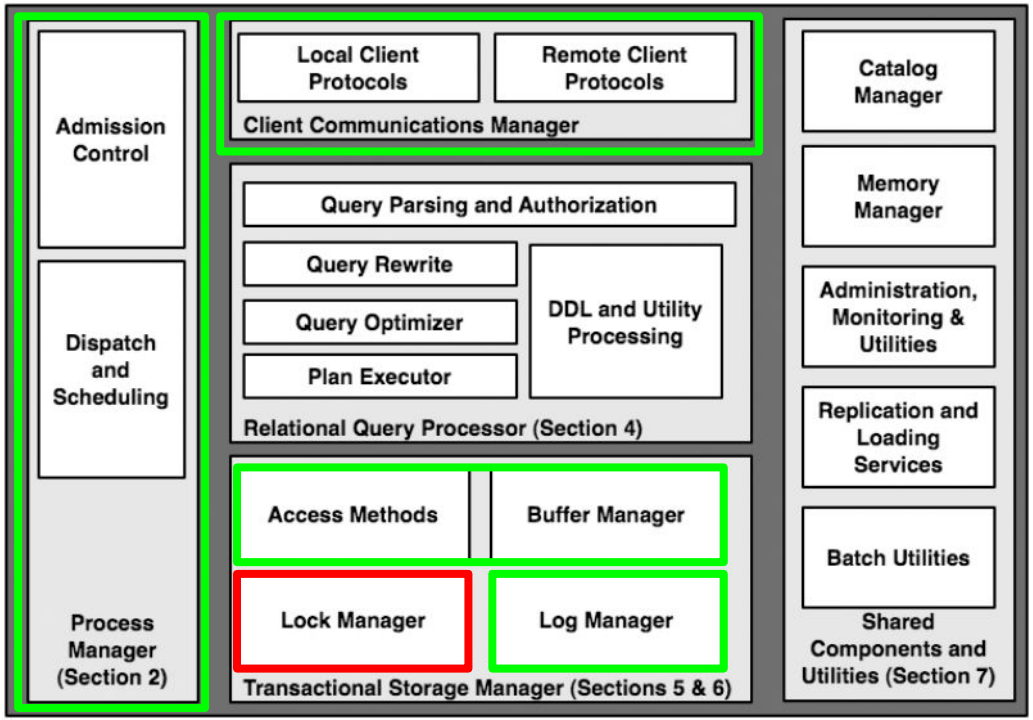
Технологии и разработка СУБД

Лекция 8. Изоляция транзакций

Анастасия Лубенникова
Александр Алексеев

После прошлых занятий



Лекция 8

- Часть 1. Изоляция транзакций
- Часть 2. Уровни изоляции
- Часть 3. Реализации контроля конкурентного доступа

Типичная транзакция

Перевод $50 со счета A на счет B

1. read (A)
2. A := A - 50
3. write (A)
4. read (B)
5. B := B + 50
6. write (B)

Критерий согласованности: сумма на счетах A и B не изменяется в результате выполнения транзакции.

Изоляция транзакций

Пример нарушения изолированности:

T1	T2
read (A) A: = A - 50 write (A)	
	read (A) read (B) print (A+B)
read (B) B: = B + 50 write (B)	

Тривиальный способ изолировать транзакции - выполнять их последовательно (**serially**)

Serializability

- **Сериализуемость** транзакций:
  - при параллельном выполнении несколько транзакций должны гарантированно выдавать такой же результат, как если бы они запускались по очереди в **некотором** порядке.
- **Расписание** (schedule) - последовательность, в которой выполняются операции нескольких конкурентных транзакций.
  - расписание содержит все операции данного набора транзакций
  - расписание сохраняет порядок инструкций внутри транзакции

Conflict serializability

- Две операции называются **конфликтующими**, если они обращаются к одним и тем же данным, и хотя бы одна из них является операцией записи.
  - $r_1(A)$, $r_2(A)$ - не конфликтующие
  - $r_1(A)$, $w_2(A)$ - конфликтующие
  - $w_1(A)$, $r_2(A)$ - конфликтующие
  - $w_1(A)$, $w_2(A)$ - конфликтующие
- Расписания S и S` называются **эквивалентными**, если расписание S может быть преобразовано  в расписание S` путем переупорядочивания не конфликтующих операций.
- Расписание называется **сериализуемым**, тогда и только тогда, когда оно эквивалентно последовательному расписанию.

Schedule 1

T1	T2
read (A) A := A - 50 write (A) read (B) B := B + 50 write (B) commit	
	read (A) A := A + 10 write (A) read (B) B := B - 10 write (B) commit

Schedule 2

T1	T2
read (A) A := A - 50 write (A)	
	read (A) A := A + 10 write (A)
read (B) B := B + 50 write (B) commit	
	read (B) B := B - 10 write (B) commit

Schedule 2

T1	T2
read (A) A := A - 50 write (A)	
	read (A) A := A + 10 write (A)
read (B) B := B + 50 write (B) commit	
	read (B) B := B - 10 write (B) commit

- Сохраняется инвариантность суммы (A+B)
- Эквивалентно последовательному расписанию
- Сериализуемое

Schedule 3

T1	T2
read (A)	
A := A - 50	
	read (A)
	A := A + 10
	write (A)
write (A)	
read (B)	
B := B + 50	
write (B)	
commit	
	read (B)
	B := B - 10
	write (B)
	commit

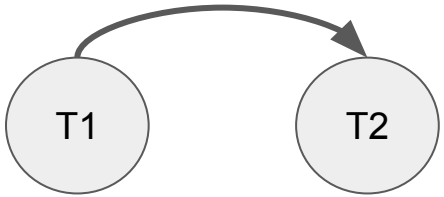
Schedule 3

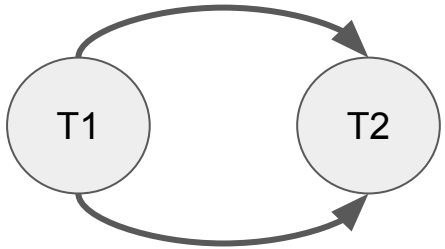
T1	T2
read (A) A := A - 50	
	read (A) A := A + 10 write (A)
write (A) read (B) B := B + 50 write (B) commit	
	read (B) B := B - 10 write (B) commit

- Не сохраняется инвариантность суммы (A+B)
- Не эквивалентно последовательному расписанию
- Не сериализуемое

Проверка сериализуемости

- Граф предшествования
  - узел для каждой подтвержденной (commited) транзакции
  - стрелка из T_i в T_j если транзакция T_i предшествует или конфликтует с T_j
- Расписание является сериализуемым тогда и только тогда, когда граф предшествования не содержит циклов





Для самостоятельного чтения

- Recoverable schedules
- Cascading rollbacks

Уровни изоляции

- Для некоторых приложений приемлемы более слабые уровни согласованности данных между транзакциями
  - Например, аналитическая транзакция, которой достаточно примерного значения общего баланса
- При параллельном выполнении транзакций возможны следующие проблемы
  - грязное чтение (dirty read)
  - неповторяемое чтение (non-repeatable read)
  - фантомное чтение (phantom read)
  - аномалия сериализации (serialization anomaly)

Dirty read

«Грязное» чтение

- транзакция читает данные, записанные параллельной незавершённой транзакцией

T1	T2
SELECT f2 FROM tbl1 WHERE f1=1;	
UPDATE tbl1 SET f2=f2+1 WHERE f1=1;	
	SELECT f2 FROM tbl1 WHERE f1=1;
ROLLBACK;	

Non-repeatable read

Неповторяемое чтение

- при повторном чтении одной и той же строки в рамках одной транзакции, ранее прочитанные значения оказываются изменёнными или удалёнными

T1	T2
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;

Phantom read

Фантомное чтение

- одни и те же запросы в одной транзакции дают разные результаты, если другая транзакция в интервалах между этими запросами добавляет или удаляет строки или изменяет значения, используемые в условиях запроса первой транзакции

T1	T2
	SELECT SUM(f2) FROM tbl1;
INSERT INTO tbl1 (f2) VALUES (20);	
COMMIT;	
	SELECT SUM(f2) FROM tbl1;

Serialization anomaly

Аномалия сериализации

- результат успешной фиксации группы транзакций оказывается несогласованным при всевозможных вариантах исполнения этих транзакций по очереди

Уровни изоляции

- степень защиты от вышеперечисленных несогласованностей данных
- в стандарте SQL-92 определены 4 уровня

	«Грязное» чтение	**Неповторяемое чтение**	**Фантомное чтение**	**Аномалия сериализации**
Read uncommited	Возможно	Возможно	Возможно	Возможно
Read commited	Невозможно	Возможно	Возможно	Возможно
Repeatable read	Невозможно	Невозможно	Возможно	Возможно
Serializable	Невозможно	Невозможно	Невозможно	Невозможно

Дополнительные материалы

- Изоляция транзакций в PostgreSQL
  https://postgrespro.ru/docs/postgrespro/9.6/transaction-iso
- Уровни изоляции транзакций с примерами на PostgreSQL
  https://habrahabr.ru/post/317884/
- Проверки целостности данных на стороне приложения
  https://postgrespro.ru/docs/postgrespro/9.6/applevel-consistency
- Лекция про транзакции (слайды к http://db-book.com/)
  http://codex.cs.yale.edu/avi/db-book/db6/slide-dir/PDF-dir/ch14.pdf

Concurrency control

- Механизм управления конкурентным доступом - реализация уровней изоляции
- Компромисс между степенью конкурентности и издержками на реализацию и работу алгоритма
- Варианты реализации
  - Lock-based
  - Timestamp-based
  - MVCC & Snapshot Isolation

Lock-based protocol (1)

- Для выполнения любого действия с объектом БД на него должна быть взята блокировка
- Режимы блокировки:
  - X - exclusive (монопольные, эксклюзивные) - блокировка на запись
  - S - shared (разделяемые) - блокировка на чтение
- Таблица совместимости блокировок
  - транзакция может получить блокировку на объект, если она совместима с другими существующими блокировками этого объекта
- Если блокировка не может быть получена, транзакция будет ожидать освобождения конфликтующих блокировок

	S	**X**
S	+	-
X	-	-

Lock-based protocol (2)

T1	T2
read (A)	
A: = A - 50	
write (A)	**lock-S (A)**
	read (A)
	unlock (A)
	lock-S (B)
	read (B)
	unlock (B)
read (B)	print (A+B)
B: = B + 50	
write (B)	

- одних только блокировок в данном примере недостаточно для того, чтобы гарантировать сериализуемость

Two-phase locking protocol (1)

- 1 фаза - нарастание блокировок
  - только взятие блокировок
  - работа с заблокированными объектами
- 2 фаза - снятие блокировок
  - во время этой фазы блокировки только снимаются
  - работа с ранее заблокированными данными может продолжаться
- Доказано, что при использовании двухфазного протокола, транзакции могут быть сериализованы в порядке получения последней блокировки.
- Но есть последовательные расписания, которые не могут быть получены в этом протоколе.

Two-phase locking protocol (2)

- Strict two-phase locking (S2PL)
  - транзакция должна держать **все эксклюзивные блокировки** до своего завершения
  - помогает избежать каскадных откатов в случае разрешения взаимных блокировок
- Rigorous two-phase locking
  - транзакция должна держать **все блокировки** до своего завершения
  - В этом протоколе транзакции могут быть сериализованы в порядке коммита

Реализация менеджера блокировок

- транзакции отправляют lock и unlock запросы к менеджеру блокировок
- затем менеджер выделяет блокировку или отправляет транзакции команду rollback (в случае разрешения взаимной блокировки)
- для обработки выделенных блокировок и запросов существует структура **lock table**, которая обычно реализуется в виде хэш-таблицы в памяти

Deadlock

Система находится в состоянии **взаимной блокировки**, когда есть набор транзакций, каждая из которых ожидает освобождения блокировки другой транзакцией.

- предотвращение возникновения взаимных блокировок:
  - predeclaration - каждая транзакция блокирует все необходимые данные перед началом выполнения
  - установить частичный порядок на данных и разрешить брать блокировки только в этом порядке

Обработка взаимных блокировок

- timeout-based
  - если транзакция ожидает получения блокировки дольше определенного времени, она откатывается
  - легко реализовать
  - возможно “голодание”
- deadlock-detection
  - граф ожидания (wait-for graph)
  - если в графе ожидания транзакций имеется цикл, система находится в состоянии взаимной блокировки
  - одну из транзакций, попавших в цикл, необходимо откатить

Материалы

- Лекция про управление конкурентным доступом (слайды к http://db-book.com/)
  http://codex.cs.yale.edu/avi/db-book/db6/slide-dir/PPT-dir/ch15.ppt

На следующей лекции

- MVCC и Snapshot Isolation

Вопросы и ответы.

- a.lubennikova@postgrespro.ru
- a.alekseev@postgrespro.ru
- Telegram: https://t.me/dbmsdev