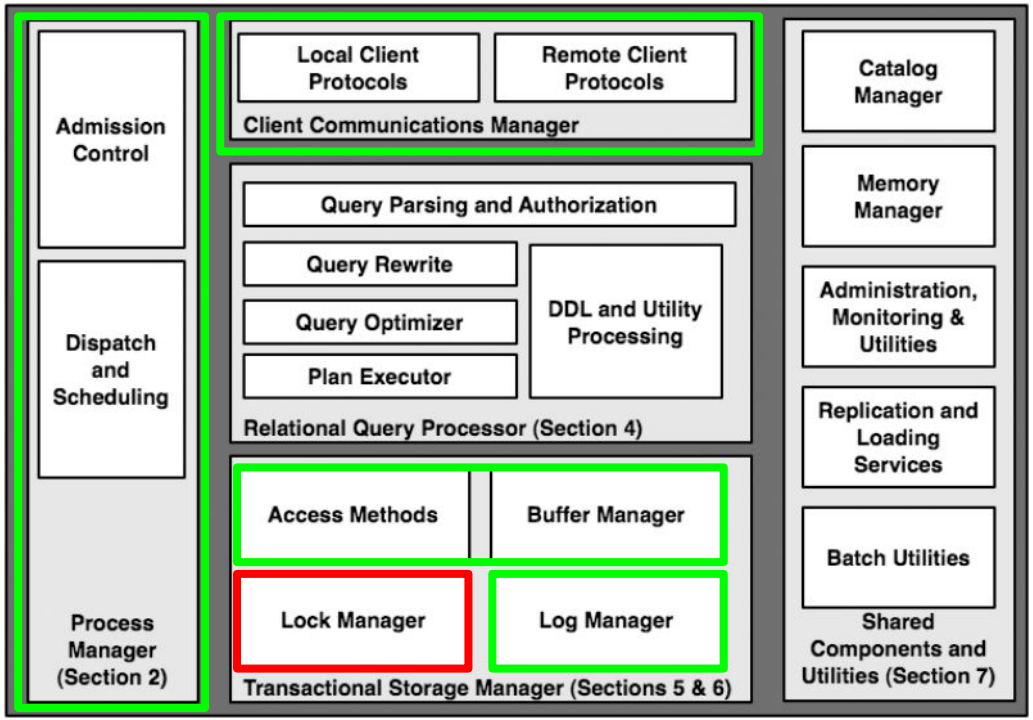
Технологии и разработка СУБД

Лекция 9. MVCC & Snapshot Isolation

Анастасия Лубенникова
Александр Алексеев

После прошлых занятий



Лекция 9

- Часть 1. Timestamp-based concurrency control
- Часть 2. MVCC
- Часть 3. Snapshot Isolation
- Часть 4. MVCC в PostgreSQL

Serializability

- **Сериализуемость** транзакций:
  - при параллельном выполнении несколько транзакций должны гарантированно выдавать такой же результат, как если бы они запускались по очереди в **некотором** порядке.

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

Мультиверсионность

- Два варианта реализации:
  - Multiversion Timestamp Ordering
  - Multiversion Two-Phase Locking
- Каждая успешная запись в таблицу приводит к созданию новой версии строки
- Для того, чтобы помечать версии строк, используются timestamps или монотонно возрастающий счетчик
- При чтении выбирается соответствующая версия строки, видимая в данной транзакции
- Читающие транзакции не ждут “писателей”

Multiversion Timestamp Ordering

- Каждая версия строки содержит 3 поля:
  - Содержимое записи
  - W-timestamp - timestamp транзакции, создавшей эту версию
  - R-timestamp - наибольший timestamp транзакции, успешно прочитавшей эту версию
- При записи новой версии, W-timestamp и R-timestamp инициализируются значениями timestamp старта транзакции
- Каждый раз при чтении R-timestamp обновляется, если TS читающей транзакции больше записанного в R-timestamp

Multiversion Timestamp Ordering

- Чтения всегда успешны
- Запись может быть отклонена, если какая-то транзакция T2, начатая позже, уже прочитала другую версию строки
- Протокол гарантирует сериализуемость
- Проблема:
  - требуется запись даже в read-only транзакциях

Multiversion Two-phase Locking

- Различают update и read-only транзакции
- Каждая версия строки содержит только 1 дополнительное поле:
  - ts-counter
  - ts-counter увеличивается при коммите транзакции
- Update транзакции используют rigorous 2PL протокол
  - транзакция должна держать **все блокировки** до своего завершения
  - успешная запись - создание новой версии строки со значением ts-counter
- Read-only транзакции используют значение ts-counter на начало транзакции в стиле Multiversion Timestamp Ordering протокола

Multiversion Two-phase Locking

- Протокол гарантирует сериализуемость
- Проблемы:
  - плохая производительность R2PL
  - надо отличать read-only и update транзакции до начала работы

MVCC

- Storage overhead
  - требуется место для служебных полей в записях
  - дополнительные версии строк
- Garbage collection
  - Может быть удалена версия tuple, которую не видит ни одна из открытых транзакций. Для описанных выше протоколов, если ts самой старой открытой транзакции > 9, и есть строка с версиями 5 и 9, версия 5 может быть удалена.

Snapshot Isolation

- Вариант MVCC протокола: все транзакции используют “снимок” базы, операции обновления данных используют 2PL, чтобы защититься от параллельных обновлений
- Проблема:
  - не гарантируется сериализуемость. Возможны аномалии, такие как write skew
- Решение: Serializable Snapshot Isolation (SSI)

Пример аномалии: lost update

- Две транзакции обновляют одни и те же данные, но виден результат только одной из них

T1	T2
BEGIN	BEGIN
UPDATE tbl SET value = value + 100	UPDATE tbl SET value = value - 100
COMMIT;	COMMIT;

Snapshot Isolation

- “first committer wins”
  - Транзакция обновляет новые элементы в своём снимке. Конкурентные обновления не видны.
  - Транзакция может быть закоммичена, только если ни одна из конкурирующих (и уже закоммиченных) транзакций не обновляла данные, которые обновляет данная транзакция
- “first updater wins”
  - Для обновления элементов на них берется lock, который держится до того, как будут закончены **все** конкурентные транзакции

PostgreSQL: First committer wins

T1	T2
BEGIN	BEGIN
SELECT x from tbl;	SELECT x from tbl;
	UPDATE tbl set x = x-1;
UPDATE tbl set x = x+1;	
COMMIT;	
	ROLLBACK: Serializarion error

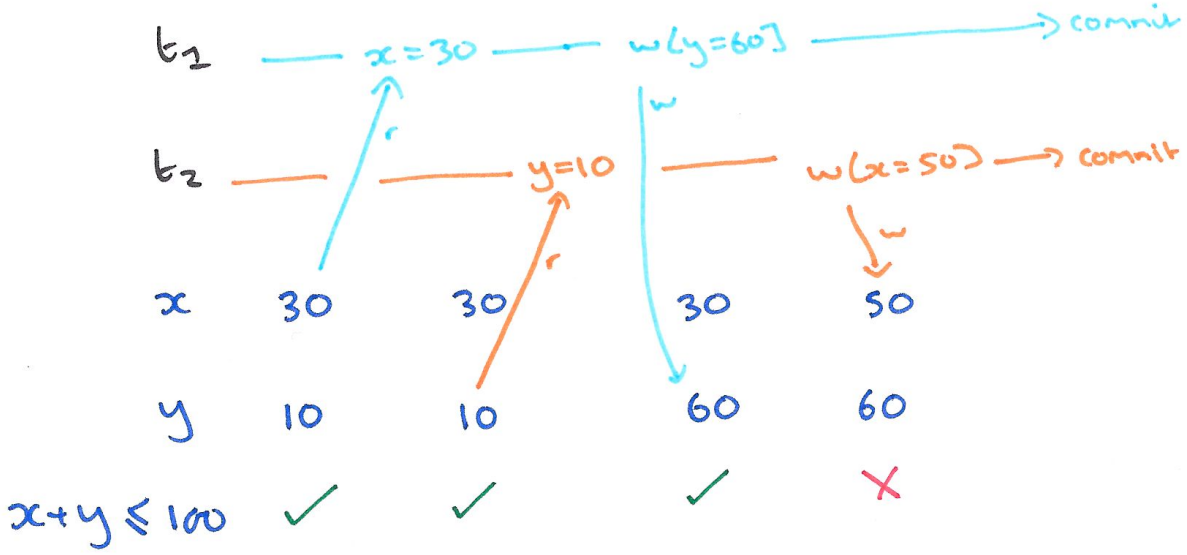
Snapshot Isolation

- Читатели никогда не блокируются и не блокируют другие транзакции
- Производительность почти как у Read Committed
- Не допускаются следующие аномалии:
  - dirty read
  - non-repeatable read
  - phantom read
  - lost update
- Проблема:
  - Не гарантирует сериализуемость
  - В сериализуемом расписании одна из транзакций видит эффекты другой
  - В SI - они полностью независимы

SI in Oracle and PostgreSQL

- При использовании уровня изоляции SERIALIZABLE в PostgreSQL до 9.1 и в Oracle на самом деле используется Snapshot Isolation
- В PostgreSQL >= 9.1 используется протокол Serializable Snapshot Isolation (SSI), который гарантирует сериализуемость

Пример аномалии: write skew



Write Skew

https://blog.acolyer.org/2016/02/24/a-critique-of-ansi-sql-isolation-levels/

Сериализация: пример

TABLE mytab;

```
class | value
-------+-------
    1 |    10
    1 |    20
    2 |   100
    2 |   200
```

T1	T2
BEGIN	BEGIN
SELECT SUM(value) FROM mytab WHERE class = 1;	SELECT SUM(value) FROM mytab WHERE class = 2;
INSERT INTO mytab (2, 30);	INSERT INTO mytab (1, 300);
COMMIT;	COMMIT;
	ROLLBACK: ERROR: could not serialize access due to read/write dependencies among transactions

FOR UPDATE

- Для защиты от аномалий также можно использовать SELECT … FOR UPDATE
- **Fun fact!** FOR UPDATE SKIP LOCKED (PostgreSQL >= 9.5)

MVCC in PostgreSQL: проблемы

- XID’ы 32-х битные. Нужен периодический wraparound
  - Есть патч Александра Короткова, реализующий 64-х битные XID’ы https://commitfest.postgresql.org/15/1178/
- Старые версии таплов нужно удалять, это делает Vacuum

Дополнительные материалы [1 / 2]

- Serializable Isolation Level in PostgreSQL
  https://www.postgresql.org/docs/current/static/transaction-iso.html#XACT-SERIALIZABLE
- Serializable Snapshot Isolation (SSI) and Predicate Locking in PostgreSQL
  https://github.com/postgres/postgres/blob/master/src/backend/storage/lmgr/README-SSI
- Documentation of Serializable Snapshot Isolation (SSI) in PostgreSQL
  https://wiki.postgresql.org/wiki/SSI

Дополнительные материалы [2 / 2]

- Уровни изоляции в реальных РСУБД
  https://github.com/ept/hermitage
- Подробнее об аномалиях и уровнях изоляции
  https://blog.acolyer.org/2016/02/24/a-critique-of-ansi-sql-isolation-levels/
- A Critique of ANSI SQL Isolation Levels – Berenson et al. 1995
  http://arxiv.org/pdf/cs/0701157.pdf

Семинар

- Последняя домашка принимается до понедельника
- Результаты теста

Вопросы и ответы.

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