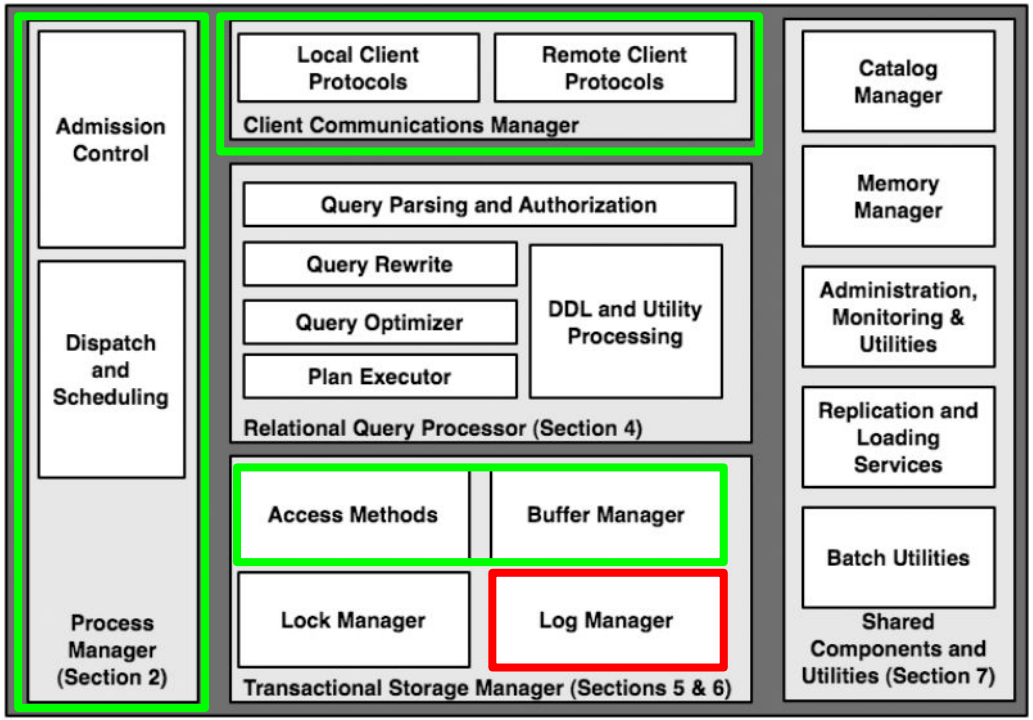
Технологии и разработка СУБД

Лекция 6. Журнал транзакций

Анастасия Лубенникова
Александр Алексеев

После прошлых занятий



Лекция 6

- Часть 1: ACID
- Часть 2: Восстановление после сбоев
- Часть 2: Журнал транзакций

ACID

- Atomicity (Атомарность)
  - никакая транзакция не может быть зафиксирована в системе частично
- Consistency (Согласованность)
  - результат успешно завершенной транзакции не нарушает ограничения схемы базы данных
- Isolation (Изолированность)
  - параллельные транзакции не влияют друг на друга
- Durability (Устойчивость)
  - изменения, сделанные успешно завершённой транзакцией, остаются сохранёнными после возвращения системы в работу

Атомарность и Устойчивость

- Транзакция может завершиться в следующих состояниях
  - commit - команда commit после выполнения всех действий
  - abort - команда rollback, аварийное отключение системы, конфликты
- Два важных свойства
  - Атомарность (Atomicity)
  - Устойчивость (Durability)
- Их выполнение обеспечивается **логированием**:
  - UNDO - действия прерванных (aborted / failed) транзакций
  - REDO - действия успешно завершившихся (commited) транзакций, которые ещё не доехали до диска

Некоторые типичные сценарии

- ошибка транзакции
  - нарушение ограничения уникальности
  - взаимная блокировка
  - транзакцию нужно откатывать
- системная ошибка
  - например, отключение питания
  - потеря данных, которые были в кэше на момент падения
  - что-то нужно либо откатить, либо докатить
- ошибка диска
  - потеря данных на диске → нужно восстанавливаться из бэкапа
  - не в этой лекции :)

Вводные знания. Вопросы.

- Что такое страница?
- Что такое буфер?
- Основная задача buffer manager?
- Что такое dirty page?
- С помощью каких команд можно синхронизировать данные с диском?
- В каких состояниях может находиться транзакция?

Вводные знания. Ответы.

- Что такое страница?
  - минимальная единица ввода/вывода
- Что такое буфер?
  - область памяти, доступная для хранения копии дисковой страницы
- Основная задача buffer manager?
  - синхронизация данных в памяти с диском, уменьшение I/O
- Что такое dirty page?
  - страница в кэше, изменившаяся по сравнению с её состоянием на диске
- С помощью каких команд можно синхронизировать данные с диском?
  - fsync(), msync()
- В каких состояниях может находиться транзакция?
  - commited, aborted, in progress

Допущения

- Изменение данных производится “in place” в страницах разделяемой памяти (не в копиях страниц)
- Все вопросы, связанные с изоляцией транзакций (блокировками) отсутствуют
  - Можно поверить, что кто-то сделал это правильно до нас;
  - Можно просто представить однопользовательскую систему;

- Придумайте самый простой способ гарантировать атомарность и устойчивость.
  - Как обрабатывать прерывание транзакции?
  - Как обрабатывать коммит?

Политики управления кэшем (1 / 2)

- FORCE
  - Это когда все изменения должны быть записаны на диск до фиксации транзакции;
  - Подход обеспечивает устойчивость без REDO лога;
  - Как можно догадаться, не самый эффективный подход;
  - И кстати, чтобы нормально восстановиться, данные на диске нельзя писать in-place;
- NO-FORCE
  - Это когда можем сказать commit до того, как сбросили все страницы данных на диск;
  - Куда более эффективный подход;
  - Но чтобы обеспечить durability, надо вести REDO лог и форсить его сброс на диск до возврата ответа "Commited";

Политики управления кэшем (2 / 2)

- STEAL
  - Это когда страницы с незакоммиченными изменениями можно вытеснять на диск;
  - Очевидно, такой подход лучше для buffer manager'a;
  - Но теперь нужен UNDO лог или его аналог, потому что, если мы упали, на диске окажется промежуточная версия данных, а старую версию надо где-то добыть;
- NO-STEAL
  - Это когда страницы с незакоммиченными изменениями нельзя вытеснять на диск;
  - Такое условие сразу дает атомарность: если упали - просто зачитываем всё с диска и вжух, всё консистентное;
  - Как можно догадаться, не самый эффективный подход;

Логирование

- Последовательная запись
- Часто лог выносят на отдельный более быстрый диск
- В лог пишутся только изменившиеся данные → лог более компактен
  - кстати, любой лог можно сжимать алгоритмами сжатия без потерь

Типы логирования

- физический
  - запись в логе содержит: старое значение (для UNDO), новое значение (для REDO) и позицию в файле данных
- логический
  - запись в логе содержит исходный запрос
  - занимает меньше места
  - REDO - повторное выполнение запроса. Это неидемпотентная операция. Почему это плохо?
  - проблемы с атомарностью
  - UNDO - нужна логика для инвертирования запроса
- **смешанный**
  - запись в логе содержит информацию только об изменениях (diff'ax) внутри одной страницы; формат может быть более компактным, чем в физическом логе

WAL (Write Ahead Log)

Алгоритм ARIES (Algorithms for Recovery and Isolation Exploiting Semantics).
Использует подход **STEAL NO-FORCE**.

1. Запись в логе должна быть сброшена на диск до того, как туда попадет страница данных.
   - Атомарность
2. Перед коммитом транзакции все записи в логе, относящиеся к ней, должны быть синхронизированы с диском.
   - Устойчивость

WAL (Write Ahead Log)

- У каждой записи в логе есть порядковый номер - **LSN** (Log Sequence Number)
  - LSN монотонно возрастает
- На каждой странице данных записан **pageLSN**
  - LSN последней записи в логе, относящейся к этой странице
- В системе хранится **flushedLSN**
  - последний LSN, синхронизированный с диском

- Перед тем, как вытеснить страницу из буфера, нужно убедиться, что pageLSN <= flushedLSN

Структуры данных

- Transaction Table
  - Одна запись для каждой активной транзакции
  - XID, status, lastLSN (LSN последнего действия в этой транзакции)
- Dirty Page Table
  - Одна запись для каждой “грязной” страницы в кэше
  - recoveryLSN - LSN **первого** изменения, после которого страница стала dirty

Картина целиком

WAL	**DB**	**Memory**
Log records	Data pages - (pageLSN)	Transaction Table Dirty Page Table flushedLSN Shared Buffers Log Tail

Успешная транзакция

1. Запись в лог: commit
2. Сбрасываем лог на диск, чтобы обеспечить условие flushedLSN >= lastLSN
3. Возвращаем ответ: commited
4. Запись в лог: end2

Теперь для всех страниц, измененных в транзакции, верно условие pageLSN <= flushedLSN. И мы можем вытеснять их в любой момент.

Checkpoint

- Чтобы уменьшить количество вычислений, необходимое при восстановлении, в журнал периодически записывается информация о состоянии системы. Это действие называют checkpoint.
- Реализации чекпойнтов могут быть разными:
  - В ARIES в лог записываются только метаданные: Transaction Table и Dirty Page Table
  - В других реализациях при чекпойнте могут синхронизироваться с диском все данные из буферного кэша

Алгоритм восстановления в ARIES

- Анализ
  - Проходим по логу в прямом направлении от последнего чекпойнта до конца
  - Реконструируем Transaction Table и Dirty Page Table
  - Наименьший recoveryLSN в Dirty Page Table - **firstLSN** точка, от которой нужно начинать REDO
- REDO
  - Проходим по логу в прямом направлении от firstLSN до конца, проигрывая REDO записи
  - Не логируем никакие действия
- UNDO
  - Проходим по логу в обратном направлении (до LSN старта самой старой in-progress транзакции на момент падения), проигрывая UNDO записи
  - Каждое действие логируем “компенсирующей” записью, чтобы успешно обрабатывать падение во время процесса восстановления (подробнее в статье из доп материалов)

Особенности реализации ARIES в PostgreSQL

- Из-за особенностей реализации транзакций (MVCC) в PostgreSQL есть только REDO лог:
  - допущение про то, что записи обновляются in-place здесь не выполняется
  - при любом обновлении создается новая версия строки и записывается рядом
  - как результат, нам не нужен UNDO лог, потому что новые данные не затирают старые
  - но нужен сборщик мусора для версий строк, которые уже не видны ни одной из транзакций (об этом на следующих лекциях)
- Во время чекпойнтов на диск вытесняются все “грязные” страницы
  - Есть отдельный процесс - checkpointer, который выполняет это по таймауту или объему изменений
  - Также можно вызвать команду CHECKPOINT

Ротация логов

- Если не удалять старые логи, рано или поздно они займут все свободное место на диске.
- Тут главное не удалить логи, которые могут потребоваться:
  - Для восстановления, если база сейчас упадет;
  - Для получения данных, видимых в активных транзакциях (для UNDO-лога);
  - Для реплик (о репликации - отдельная лекция);

Дополнительные материалы

- **Concurrency control and recovery. By Michael J. Franklin (только разделы про recovery)**
  zoo.cs.yale.edu/classes/cs637/franklin97concurrency.pdf
- Как работает реляционная база данных
  https://makeomatic.ru/blog/2015/11/24/relational_database_3/
- The internals of PostgreSQL. Write Ahead Log
  http://www.interdb.jp/pg/pgsql09.html
- Corruption War Stories
  https://www.pgcon.org/2017/schedule/events/1048.en.html

Дополнительные материалы: по сжатию

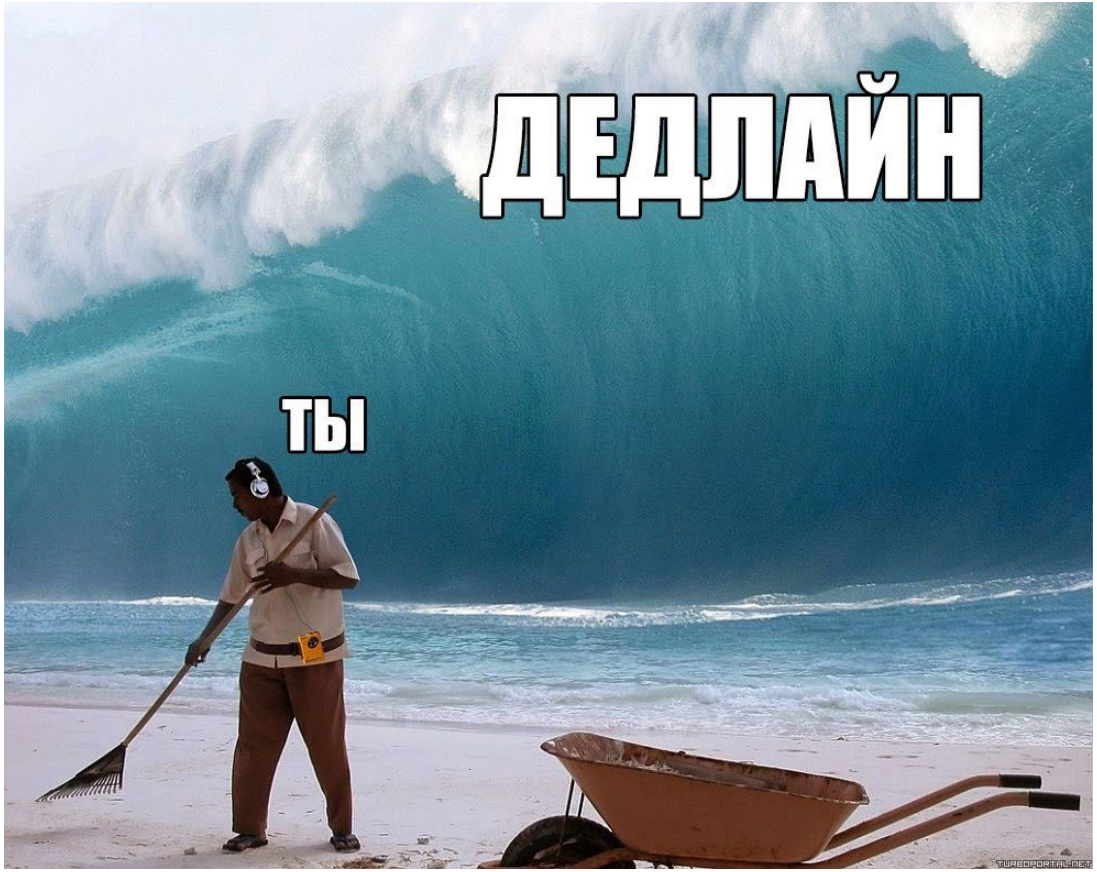
- Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео https://www.ozon.ru/context/detail/id/1096097/
- Сжатие данных, изображений и звука https://www.ozon.ru/context/detail/id/3250274/

Об-суж-дение!

- Вопрос залу:
  - Когда может быть **оправдано** сделать fsync = off?

Об-суж-дение!

- Вопрос залу:
  - Давайте подумаем, как undo log помогает с версионируемостью данных?
  - Hint: PITR
  - Почему нельзя так же сделать с redo log’ом?


ДЕДЛАЙН
ТЫ

Вопросы и ответы.

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