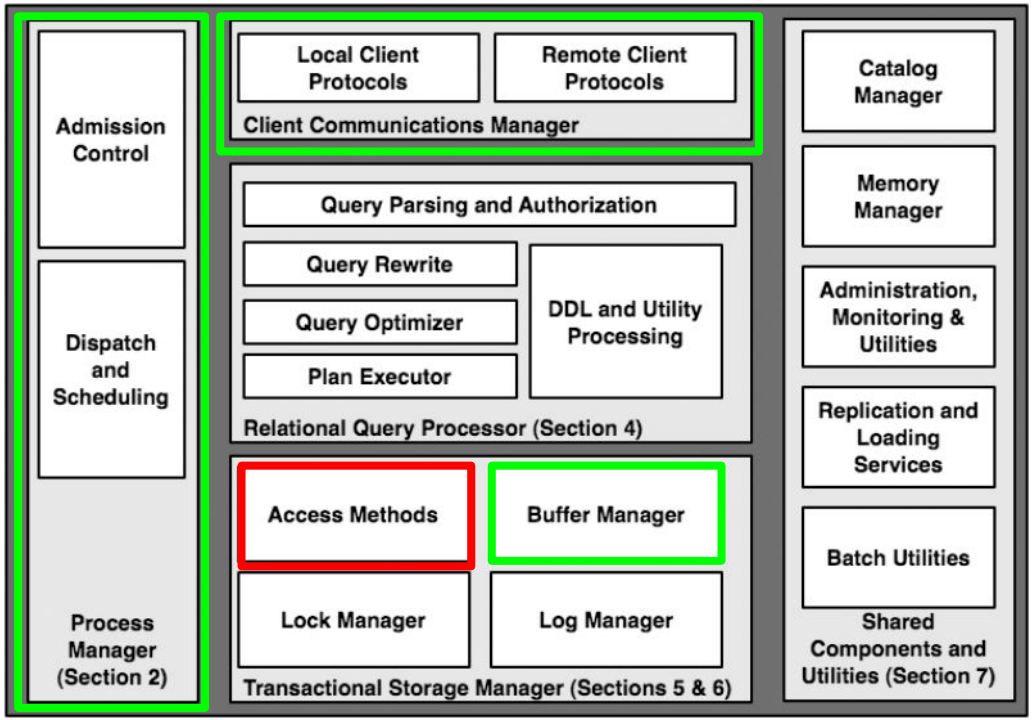
Технологии и разработка СУБД

Лекция 4. Методы доступа

Анастасия Лубенникова
Александр Алексеев

После прошлого занятия



Лекция 4

- Часть 1: Методы доступа
- Часть 2: Оценка сложности
- Часть 3: B-Tree и другие структуры

Метод доступа

- Файл базы данных (‘relation’)
  - набор страниц, каждая из которых содержит набор записей
- Метод доступа - API для работы с файлами базы данных
  - insert/delete/modify record
  - fetch record (+ здесь же применение условий поиска)
  - идентификатор записи (page id + offset) однозначно определяет положение в файле

Типы файлов БД

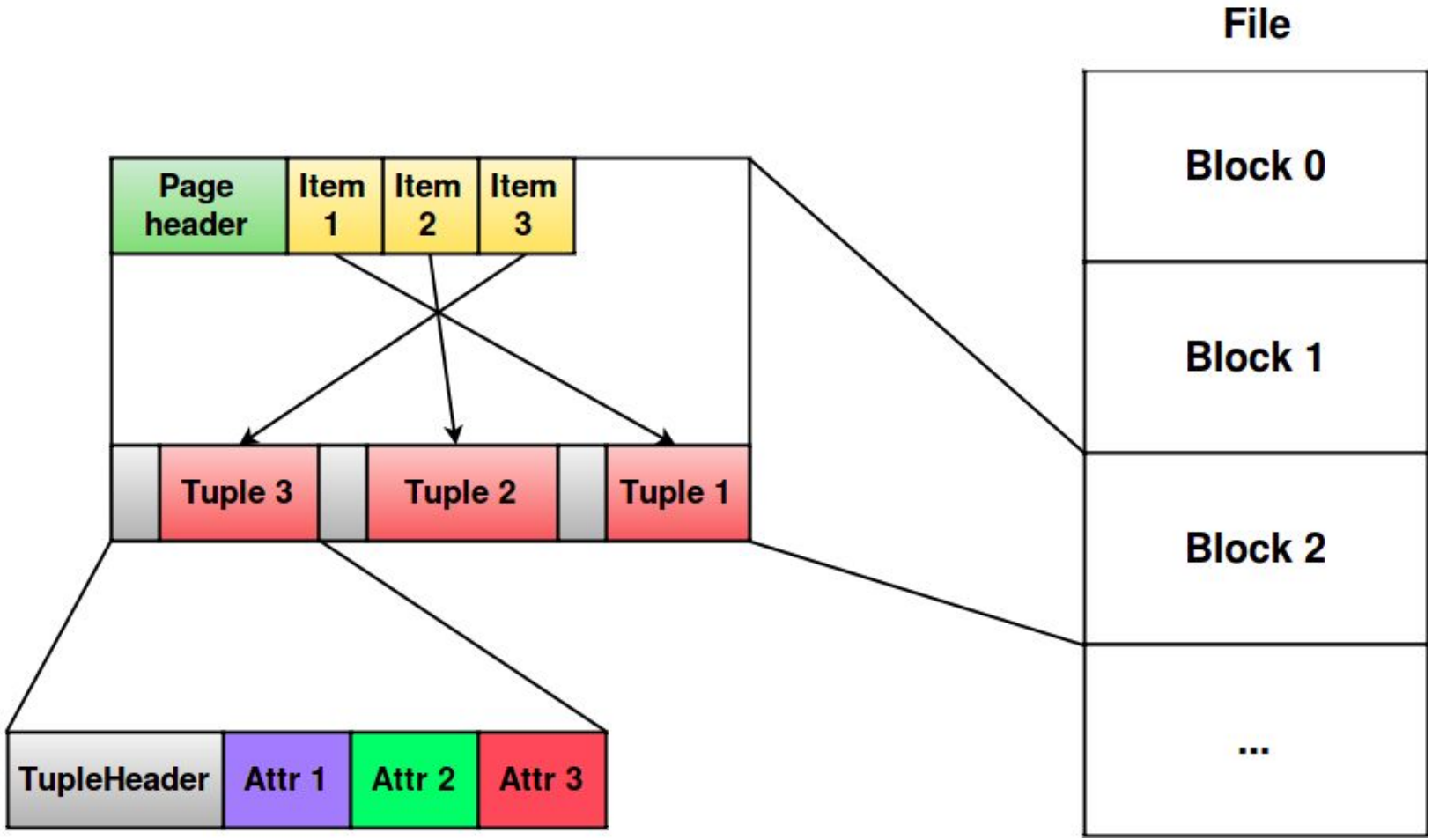
- Heap File
  - записи не упорядочены
- Sorted File
  - записи хранятся в порядке сортировки
- Index File
  - B+ Trees, Hash Tables, R-tree, LSM, GIN, RUM …

Heap File

- Вставка:
  - Всегда дописывать в конец файла. Проблема разрастания файла - bloat.
- Удаление:
  - Когда файл уменьшается нужно освобождать страницы.
  - Как отслеживать свободное место?
    - Реализация Heap на структуре данных List
    - Хранение FSM (Free Space Map)
- Чтение:
  - По идентификатору (page id + offset)
  - Sequential scan - просмотр всех записей

Page Layout

- Длина записей: фиксированная/переменная ?
- Как найти в файле запись по record_id (page_id, offset)?
- Как происходит добавление/удаление записей?


File
Block 0
Block 1
Block 2
...
Page header
Item 1
Item 2
Item 3
Tuple 3
Tuple 2
Tuple 1
TupleHeader
Attr 1
Attr 2
Attr 3

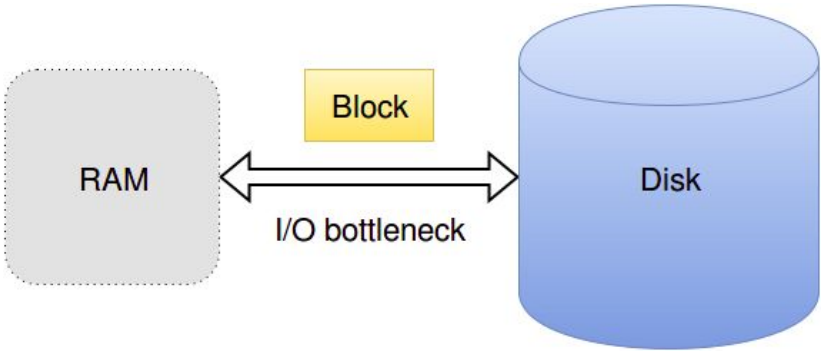
Типы файлов БД

- Heap File
  - для вставки и чтения всех записей
- Sorted File
  - для запросов с ORDER BY и поиска по диапазону (range scan)
- Index File
  - для быстрого поиска записей и эффективного изменения записей

Оценка стоимости. Общие замечания.

- B - кол-во блоков в файле
- R - кол-во записей в блоке
- D - среднее время чтения/записи блока

- Для упрощения опускаем детали
  - последовательного/случайного I/O
  - prefetching
  - стоимость операций в памяти = 0



Оценка стоимости

	Heap File	**Sorted File**
Чтение всех записей	B * D	B * D
Поиск на равенство	(0.5 * B) * D	**log2(B) * D**
Поиск по диапазону	B * D	**((log2(B)) + pages) *D**
Вставка	**2 * D**	((log2(B)) + B) * D
Удаление	(0.5 * B+1) * D	((log2(B)) + B) * D

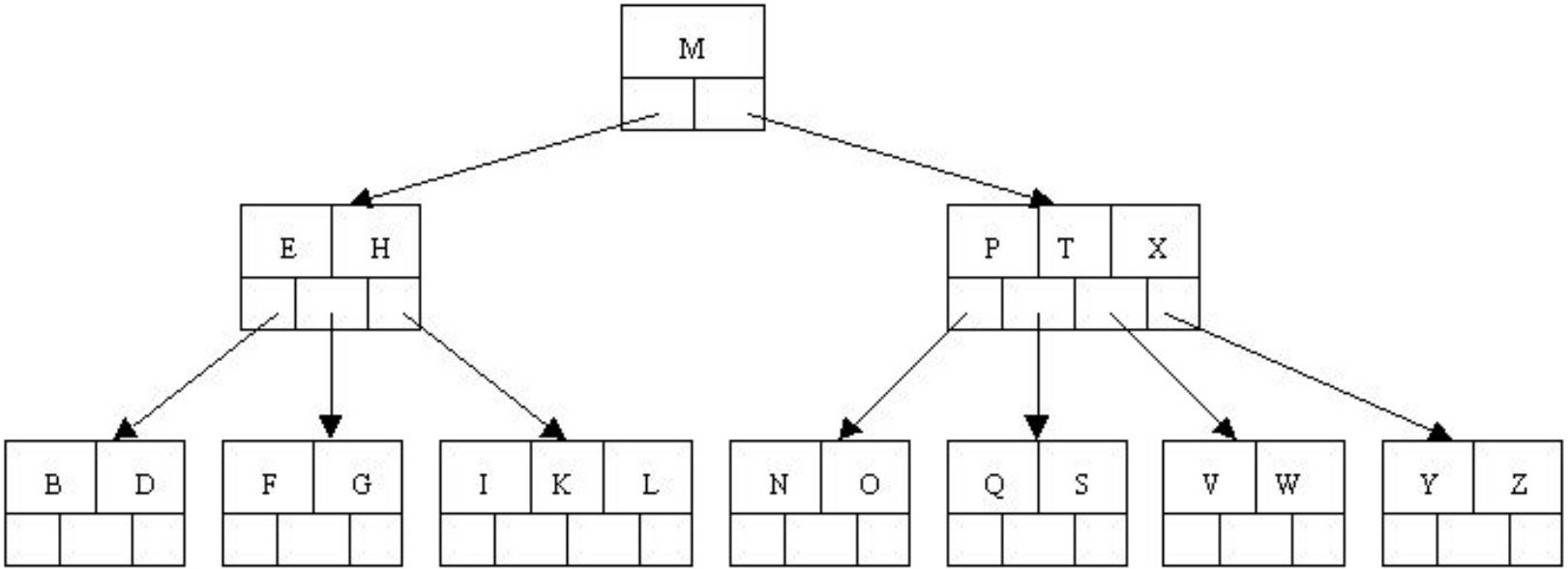
Индексы

Используются для ускорения поиска

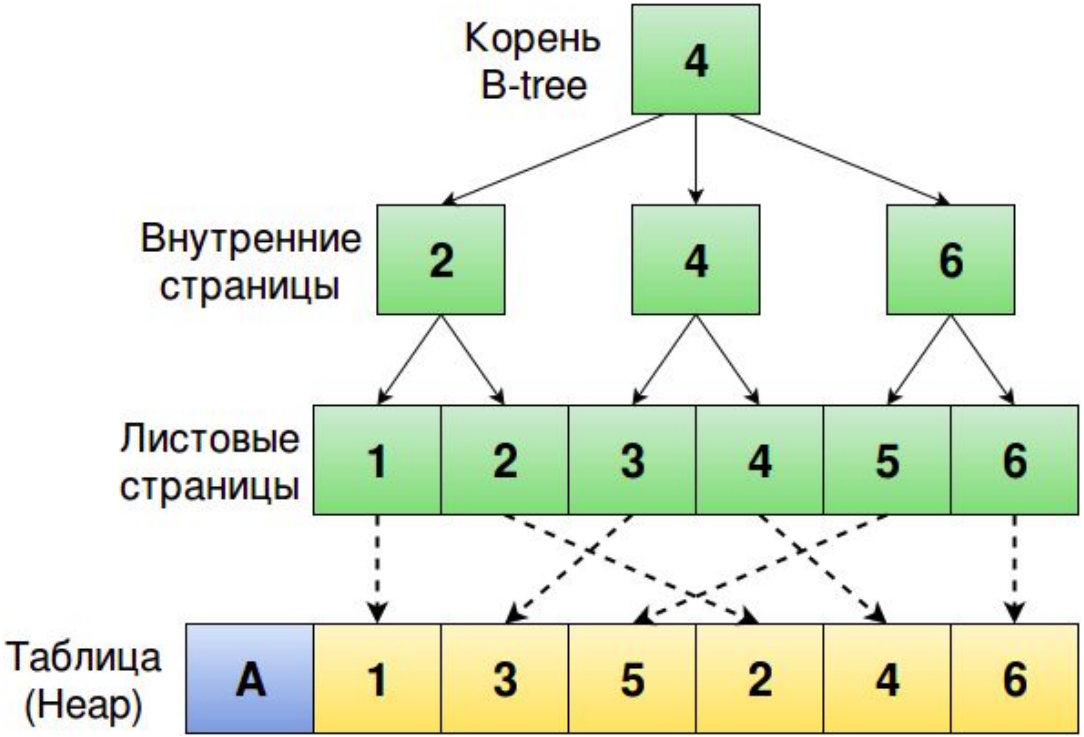
- “Первичные” индексы - Index Organized Table
  - содержат ключ и данные
- Вторичные индексы
  - содержат ключ и ссылку на запись в heap
- Еще в контексте классификации
  - функциональные, частичные, покрывающие

Важная вещь! Проверка уникальности.

B-Tree



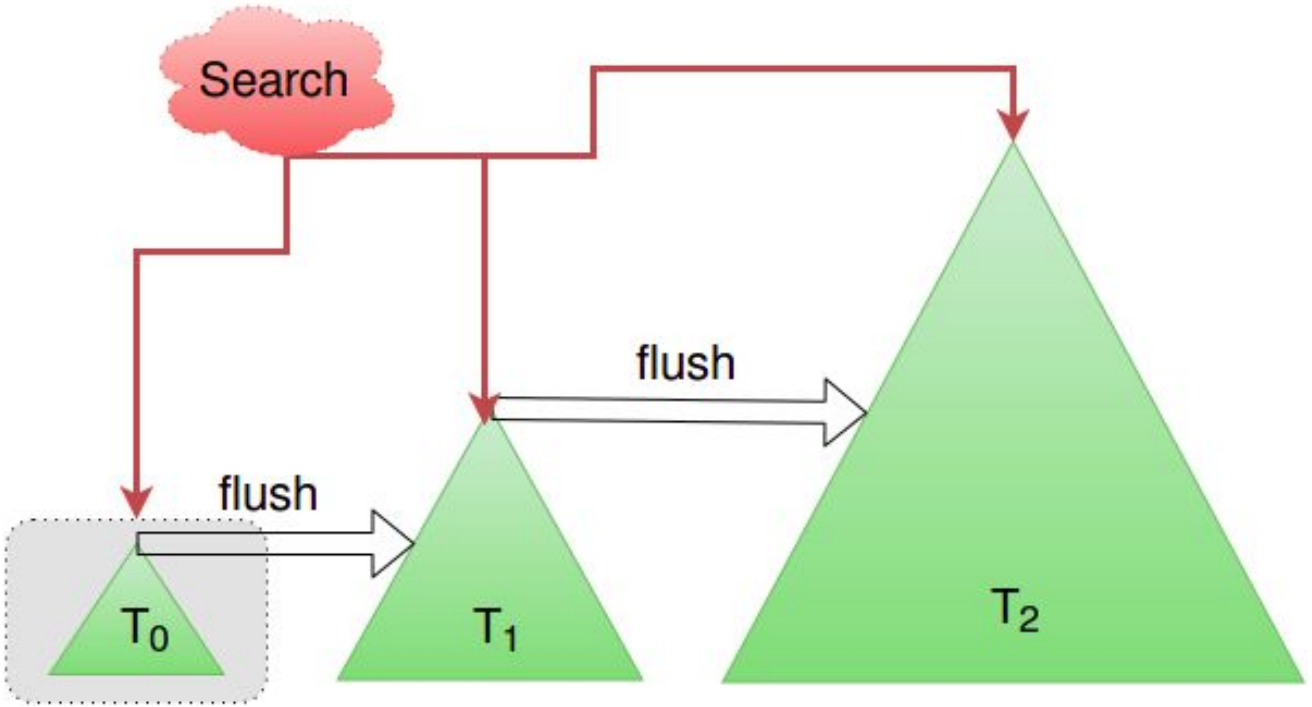
B+ tree


Корень
B-tree
4
Внутренние
страницы
2
4
6
Листовые
страницы
1
2
3
4
5
6
Таблица
(Heap)
A
1
3
5
2
4
6

Разница между B-tree и B+-tree

- B+ дерево хранит ссылки на значения только в листах
  - => больше ключей влезает в страницу
  - => меньше cache misses
- Листы B+дерева перелинкованы
  - => эффективные range scan’ы

LSM-trees

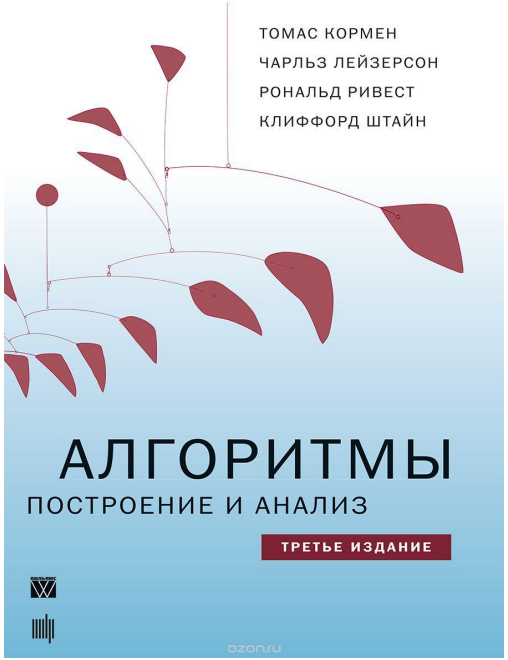


LSM-trees: коротко о главном

- LSM-tree = Log-structured merge-tree
- Работают быстрее B+деревьев при частой записи
- Доступ к данным в памяти (MemTable) осуществляется быстро
- Запись на диск происходит только последовательно
- Данные на диске (SSTable) неизменяемы
  - Меньше проблем с версионированием и получением снапшота
- Для слияния деревьев используется сортировка слиянием
- При удалении данных должен быть записан thumb stone
- Где используются: LevelDB, RocksDB, Cassandra, ...

Дополнительные материалы

- On Disk IO
  https://medium.com/@ifesdjeen/on-disk-storage-part-4-b-trees-30791060741
- Wikipedia: B-Tree
  https://en.wikipedia.org/wiki/B-tree
- Алгоритмы. Построение и анализ
  http://www.ozon.ru/context/detail/id/33769775/
- Простой пример реализации B-Tree
  https://github.com/afiskon/c-btree-example



Вопросы и ответы.

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