

Авторы: Егор Рогов, Павел Лузанов



**Обратная связь**
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

# Темы

Сканирование по битовой карте

Сравнение эффективности разных методов доступа

# Битовая карта

Построение битовой карты (Bitmap Index Scan)

Сканирование по битовой карте (Bitmap Heap Scan)

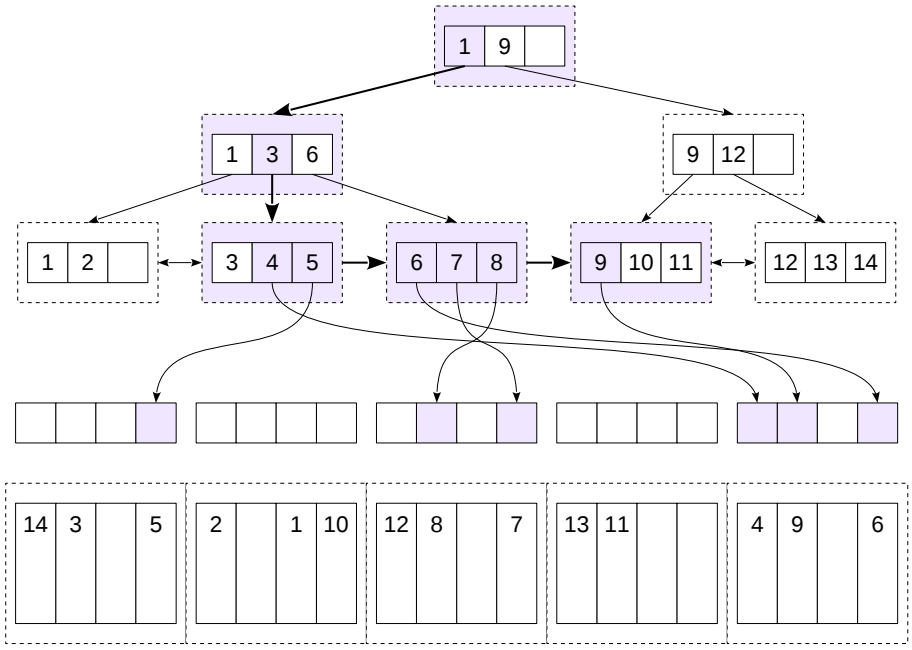
Использование памяти

Параллельное сканирование (Parallel Bitmap Heap Scan)

Объединение битовых карт

Кластеризация

# Bitmap Index Scan



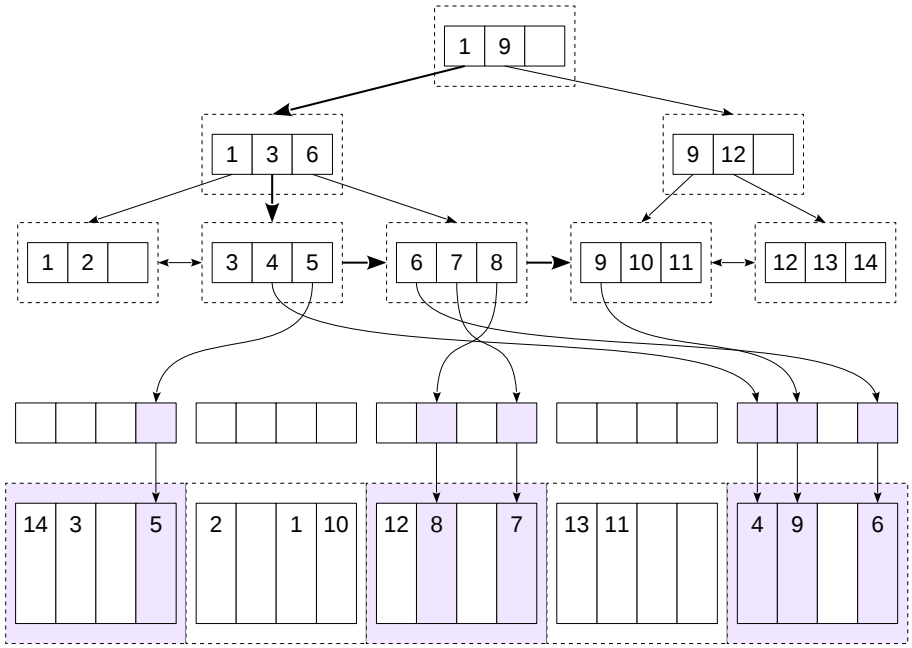



Просмотр одних и тех же табличных страниц по нескольку раз крайне неэффективен. Даже в лучшем случае, если нужная страница находится в буферном кэше, ее нужно найти и заблокировать (см. курс DBA2: тема «Буферный кэш» модуля «Журналирование»), а в худшем приходится иметь дело с произвольными чтениями с диска.

Чтобы не тратить ресурсы на повторный просмотр табличных страниц, применяется еще один способ доступа — сканирование по битовой карте. Он похож на обычный индексный доступ, но происходит в два этапа.

Сначала сканируется индекс (Bitmap Index Scan) и в локальной памяти процесса строится битовая карта. В этой карте отмечаются те строки, которые должны быть прочитаны.

# Bitmap Heap Scan






Когда индекс просканирован и битовая карта готова, начинается сканирование таблицы (Bitmap Heap Scan). При этом:

- страницы читаются последовательно (увеличивается шанс воспользоваться кэшем ОС);

- каждая страница просматривается ровно один раз.

# Использование памяти

## Битовая карта без потери точности

пока размер карты не превышает *work_mem*,
информация хранится с точностью до строки

## Битовая карта с потерей точности

если память закончилась, происходит огрубление
части уже построенной карты до отдельных страниц

при чтении табличной страницы требуется перепроверка условий

требуется примерно 1 МБ памяти на 64 ГБ данных;
ограничение памяти может быть превышено



Логически битовую карту можно представлять себе, как битовый массив, в котором каждой табличной строке соответствует отдельный бит. На самом деле карта поделена на фрагменты; для каждого фрагмента указана начальная табличная страница и далее следует не очень большой битовый массив. За счет этого битовая карта, в которой отмечено всего несколько строк, будет занимать немного места.

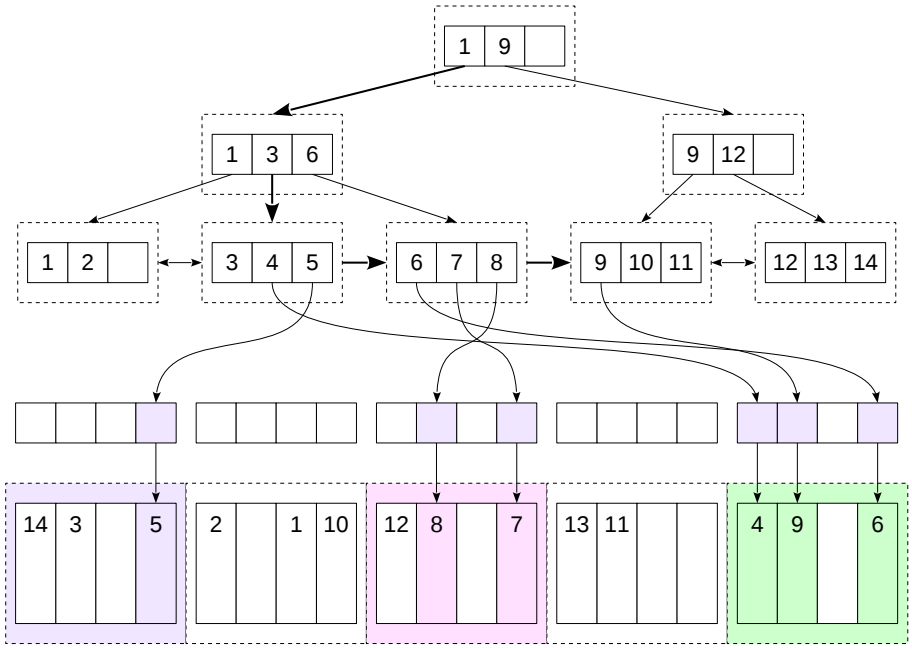
Битовая карта хранится в локальной памяти обслуживающего процесса и под ее хранение выделяется *work_mem* байтов. Временные файлы никогда не используются.

Если карта перестает помещаться в *work_mem,* часть ее фрагментов «огрубляется» — каждый бит начинает соответствовать целой странице, а не отдельной строке (lossy bitmap). Освободившееся место используется для того, чтобы продолжить строить карту.

В принципе, при сильно ограниченном *work_mem* и большой выборке, битовая карта может не поместиться в памяти, даже если в ней совсем не останется информации на уровне строк. В таком случае ограничение *work_mem* нарушается — под карту будет дополнительно выделено столько памяти, сколько необходимо.

При потере точности требуется перепроверка условий при чтении табличных страниц, а это сказывается на производительности. Если две битовые карты объединяются, причем фрагмент хотя бы одной из карт неточный, то и результирующий фрагмент тоже вынужденно будет неточным. Поэтому размер *work_mem* играет большую роль для этого способа доступа.

# Parallel Bitmap Heap Scan



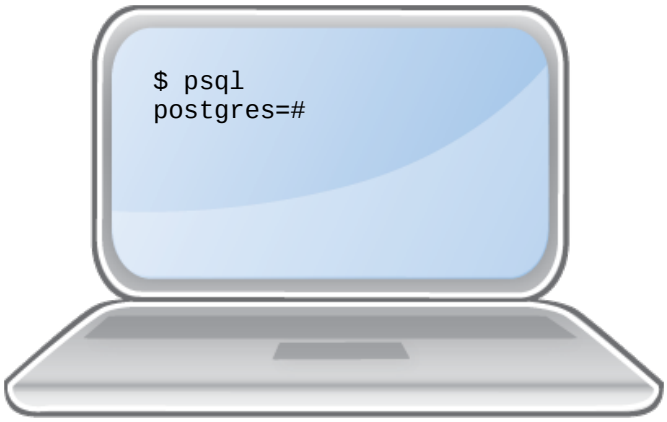



Сканирование по битовой карте может выполняться параллельно.

Первый этап — сканирование индекса — всегда выполняется последовательно ведущим процессом.

А второй этап — сканирование таблицы — выполняется рабочими процессами параллельно. Это происходит аналогично параллельному последовательному сканированию.

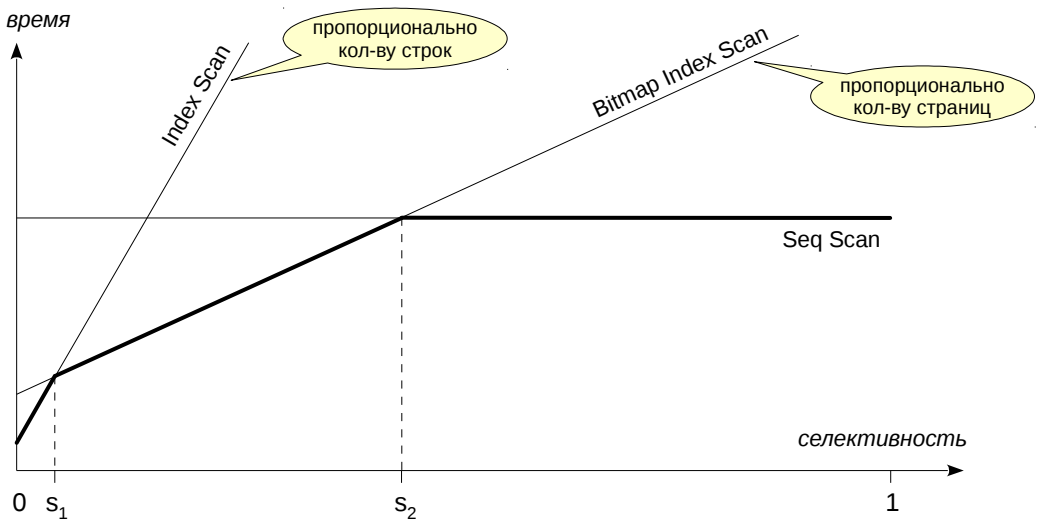
# Демонстрация



# Сравнение эффективности

Сравнение различных методов доступа

# Сравнение эффективности




значения селективностей $s_1$ и $s_2$ зависят от таблицы и индекса



Индексное сканирование лучше всего работает при очень высокой селективности, когда по индексу выбирается одно или несколько значений.

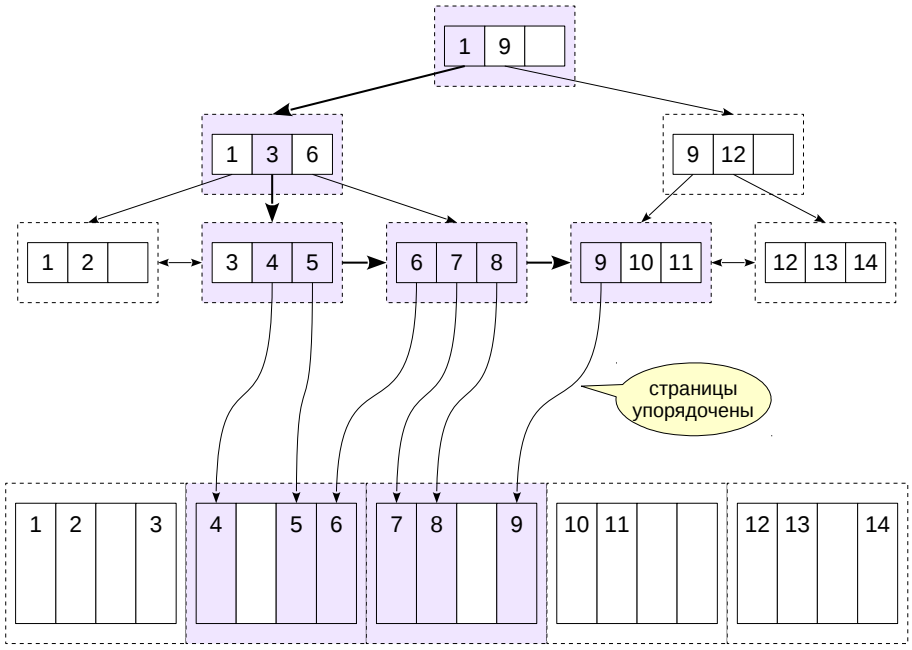
При средней селективности лучше всего показывает себя сканирование по битовой карте. Оно работает лучше индексного сканирования, поскольку обходится без повторных чтений одних и тех же страниц. Однако сканирование по битовой карте имеет накладные расходы на построение карты, поэтому при высокой селективности оно проигрывает.

При низкой селективности лучше всего работает последовательное сканирование: если надо выбрать все или почти все табличные строки, обращение к индексным страницам только добавляет накладные расходы. Этот эффект усиливается в случае вращающихся дисков, где стоимость произвольного чтения существенно выше стоимости чтения последовательно расположенных страниц.

Конкретные значения селективностей, при которых становится выгодно переключиться на другой метод доступа, сильно зависят от конкретной таблицы и конкретного индекса. Планировщик учитывает множество параметров, чтобы выбрать наиболее подходящий способ.

Еще одно замечание: при индексном доступе результат возвращается отсортированным, что в ряде случаев увеличивает привлекательность такого доступа даже при низкой селективности.

# Index Scan / Bitmap Scan

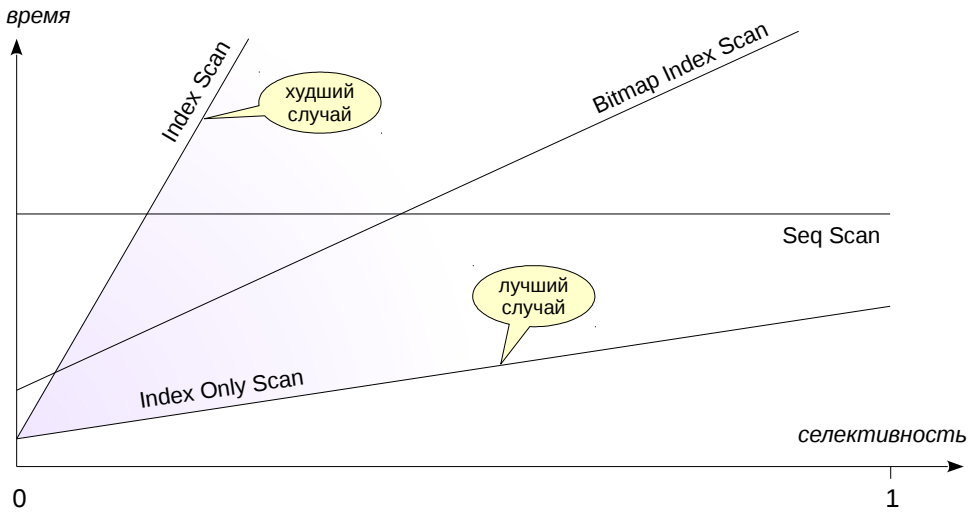





Если данные в таблице физически упорядочены, обычное индексное сканирование не будет возвращаться к одной и той же табличной странице повторно. В таком (нечастом на практике) случае метод сканирования по битовой карте теряет смысл, проигрывая обычному индексному сканированию.

Разумеется, планировщик это тоже учитывает (как именно это работает, говорится в теме «Статистика»).

# Сравнение эффективности




эффективность Index Only Scan зависит от карты видимости



Метод исключительно индексного сканирования сильно зависит от актуальности карты видимости (и от того, насколько много страниц действительно содержат только актуальные версии строк).

В лучшем случае этот метод доступа может быть эффективней последовательного сканирования даже при низкой селективности (если индекс меньше, чем таблица, и особенно на SSD-дисках).

В худшем случае, когда требуется проверять видимость каждой строки, этот метод доступа вырождается в обычное индексное сканирование.

Поэтому планировщику приходится учитывать состояние карты видимости: при неблагоприятном прогнозе исключительно индексное сканирование не применяется из-за опасности получить замедление вместо ускорения.

# Итоги

## Сканирование по битовой карте

- исключает повторное чтение табличных страниц
- позволяет объединять несколько индексов
- эффективно при средней селективности

## Несколько методов доступа позволяют планировщику выбрать лучший, учитывая различные факторы:

- селективность условия
- необходимость обращаться к таблице и карта видимости
- соответствие порядка данных в таблице и в индексе
- необходимость получить отсортированную выборку
- потребность быстрого получения первых результатов
- и другие

# Практика

1. Создайте индекс по столбцу amount таблицы перелетов (ticket_flights).
2. Напишите запрос, находящий количество перелетов стоимостью более 180 000 руб. (менее 1 % строк). Какой метод доступа был выбран? Сколько времени выполняется запрос?
3. Запретите выбранный метод доступа, снова выполните запрос и сравните время выполнения. Прав ли был оптимизатор?
4. Повторите пункты 2 и 3 для стоимости менее 44 000 руб. (чуть более 90 % строк).



3. Для запрета метода доступа установите один из параметров в значение off:

- enable_seqscan,

- enable_indexscan,

- enable_bitmapscan.

Например:

```
SET enable_seqscan = off;
```

4. Не забудьте восстановить значение измененного параметра, например:

```
RESET enable_seqscan;
```