B-tree индексы: архитектура и новые возможности

Анастасия Лубенникова

Индексы

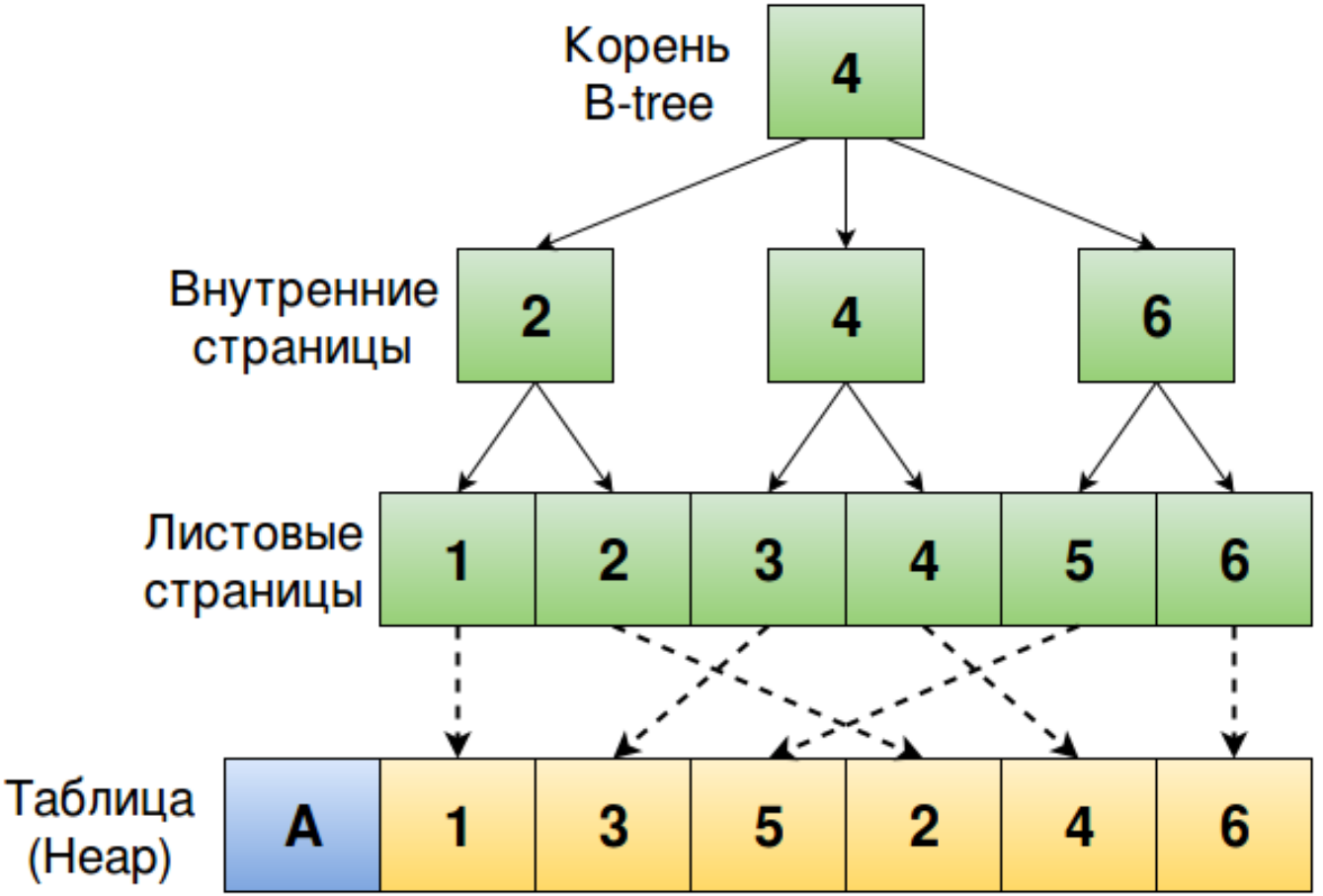
- Индекс - это вспомогательная структура данных, предназначенная для ускорения выполнения некоторых запросов.

- Ускорение поиска, сортировки
- Оптимизация соединения таблиц
- Поддержание ограничений целостности
  - unique
  - primary key

Индексы в PostgreSQL

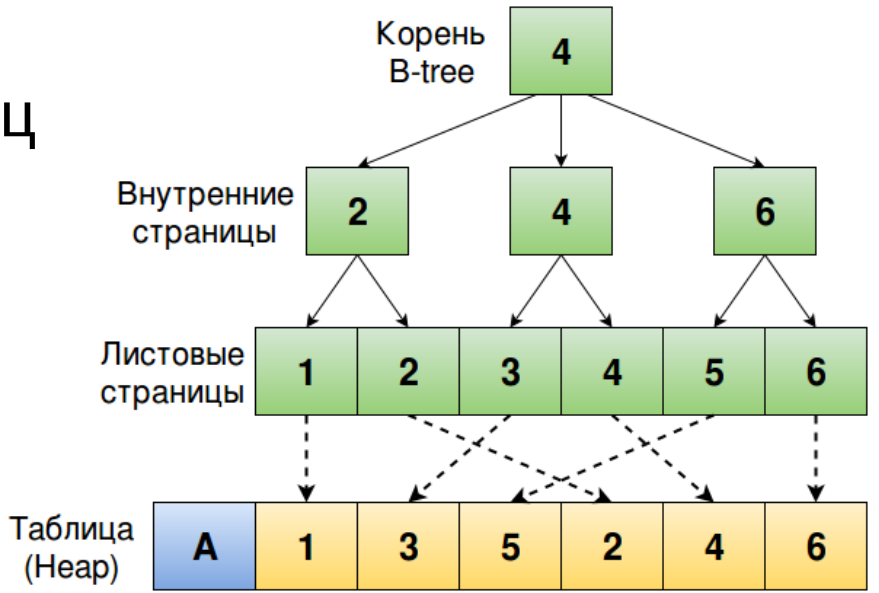
- Вторичные
- Расширяемые
  - методы доступа
  - операторы классов
- Не содержат информации о видимости данных

B-tree индекс



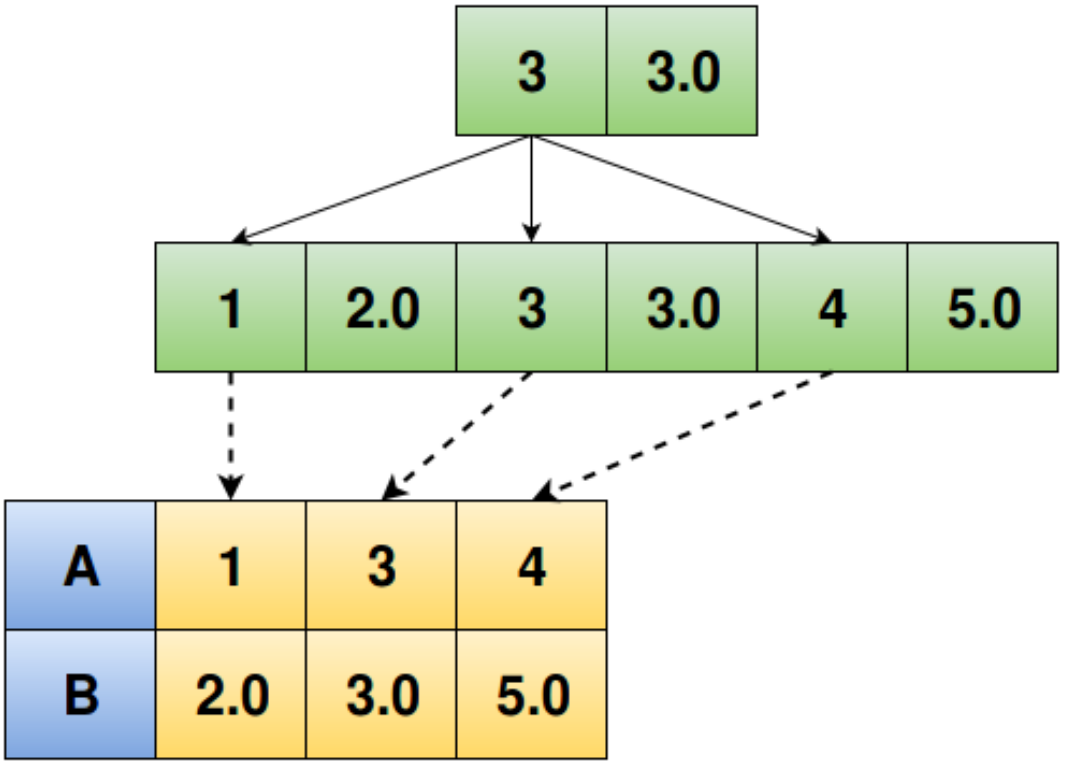
B-tree индекс

- Единственный индекс для ограничений целостности
  - UNIQUE
  - PRIMARY KEY
- Индекс по умолчанию
- Индексы всех системных таблиц



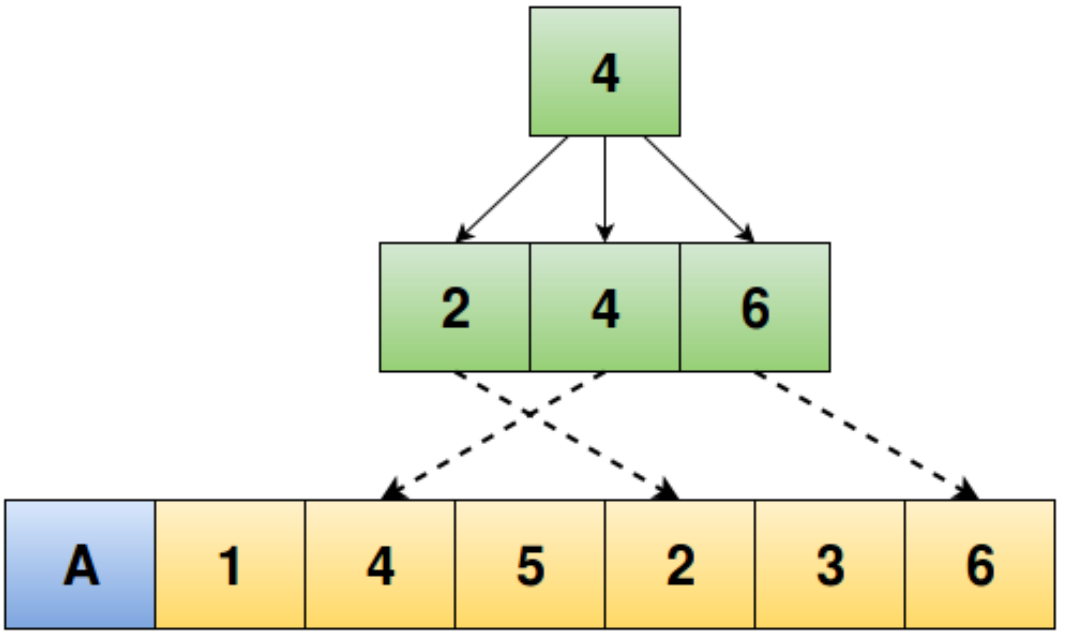
Составной индекс

- CREATE INDEX ON tbl (a, b);



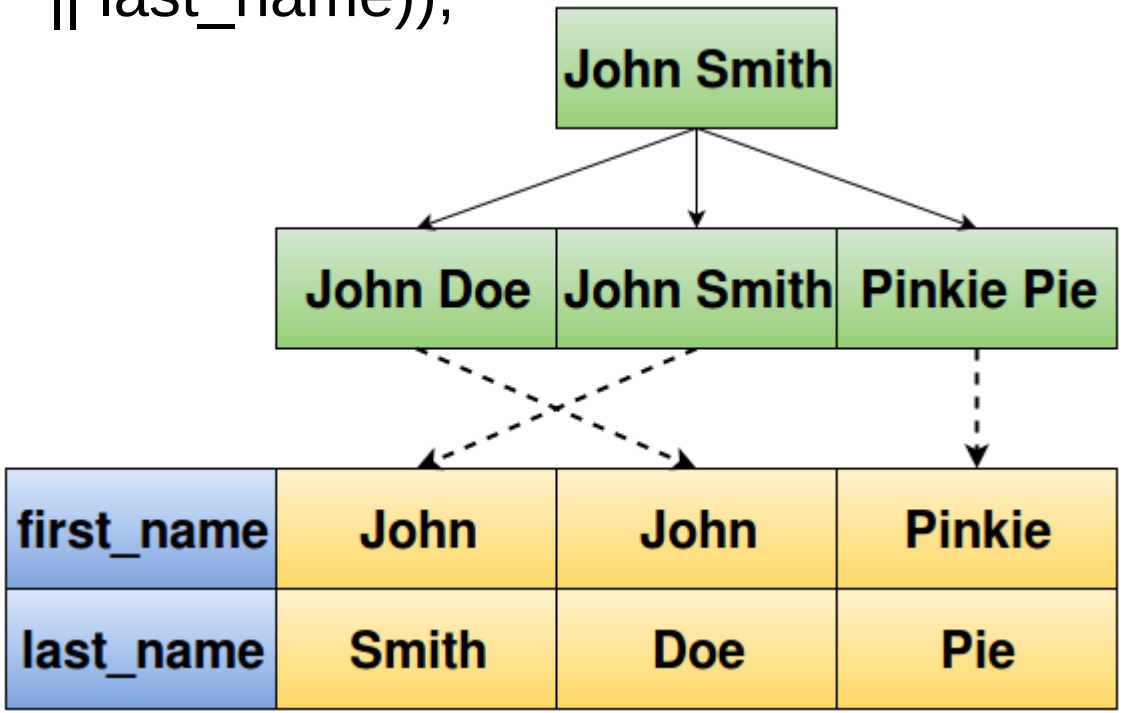
Частичный индекс

- CREATE INDEX ON tbl (a) WHERE a%2 = 0;

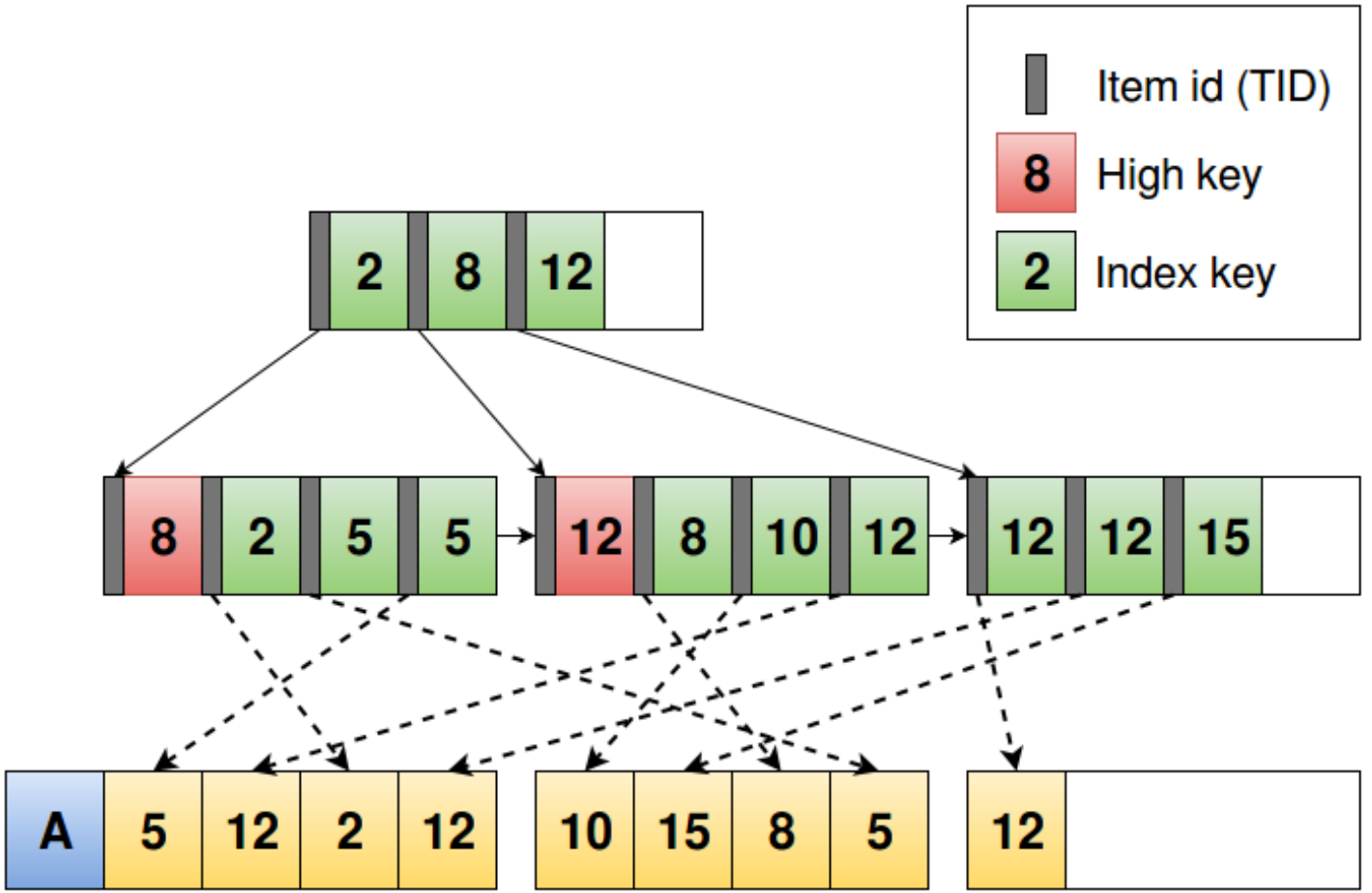


Индекс по выражению

- CREATE INDEX people_names
  ON people ((first_name || ' ' || last_name));

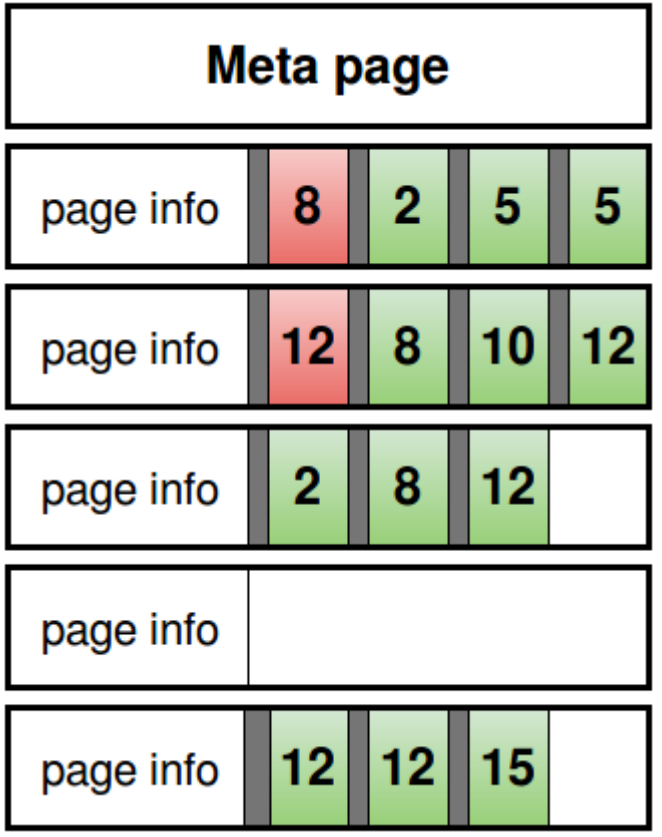


Строение B-tree индекса

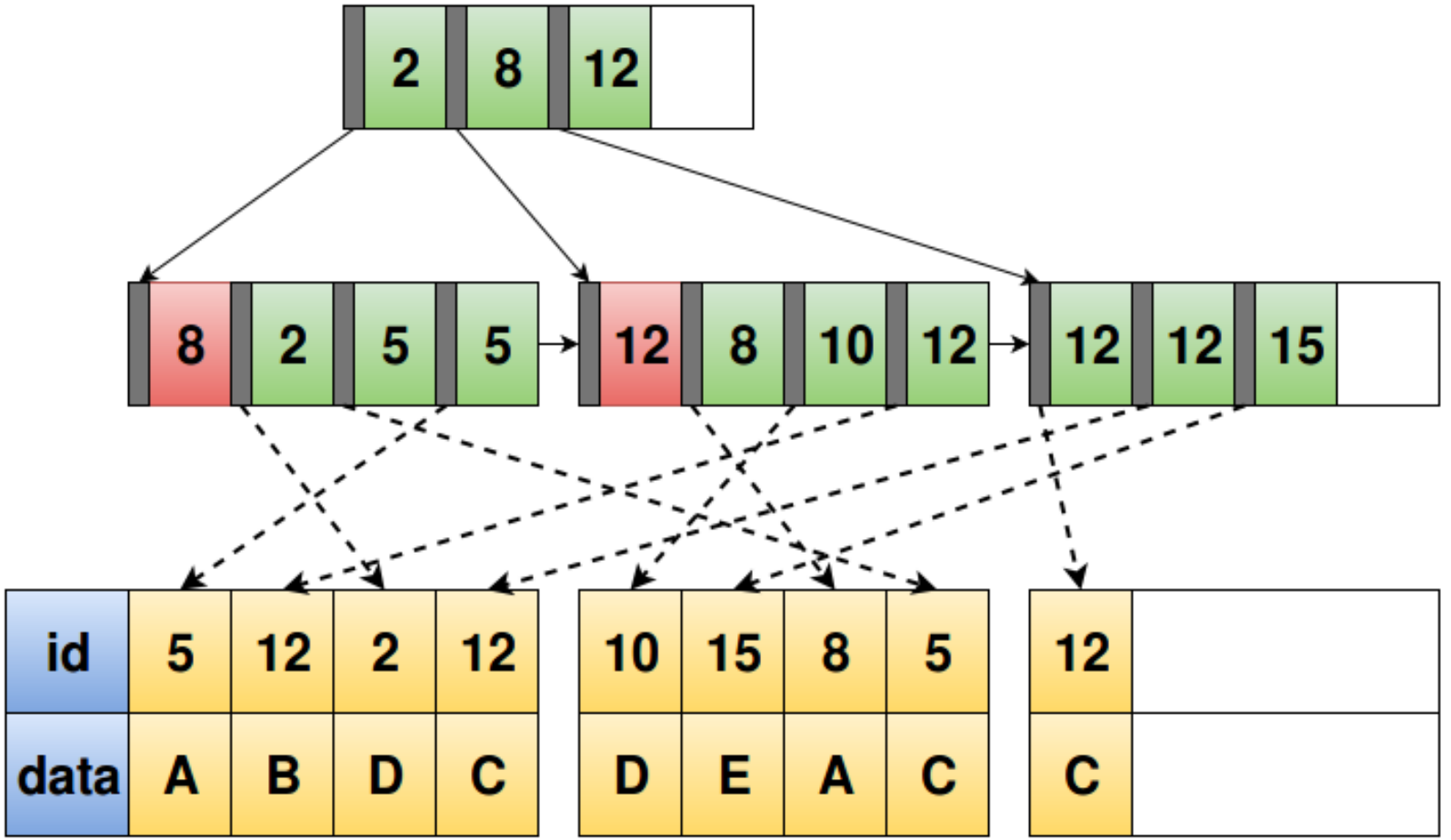


Файл индекса

- Один или несколько непрерывных файлов
- Страницы по 8 Кб
- Пустые страницы не удаляются
  - Free Space Map

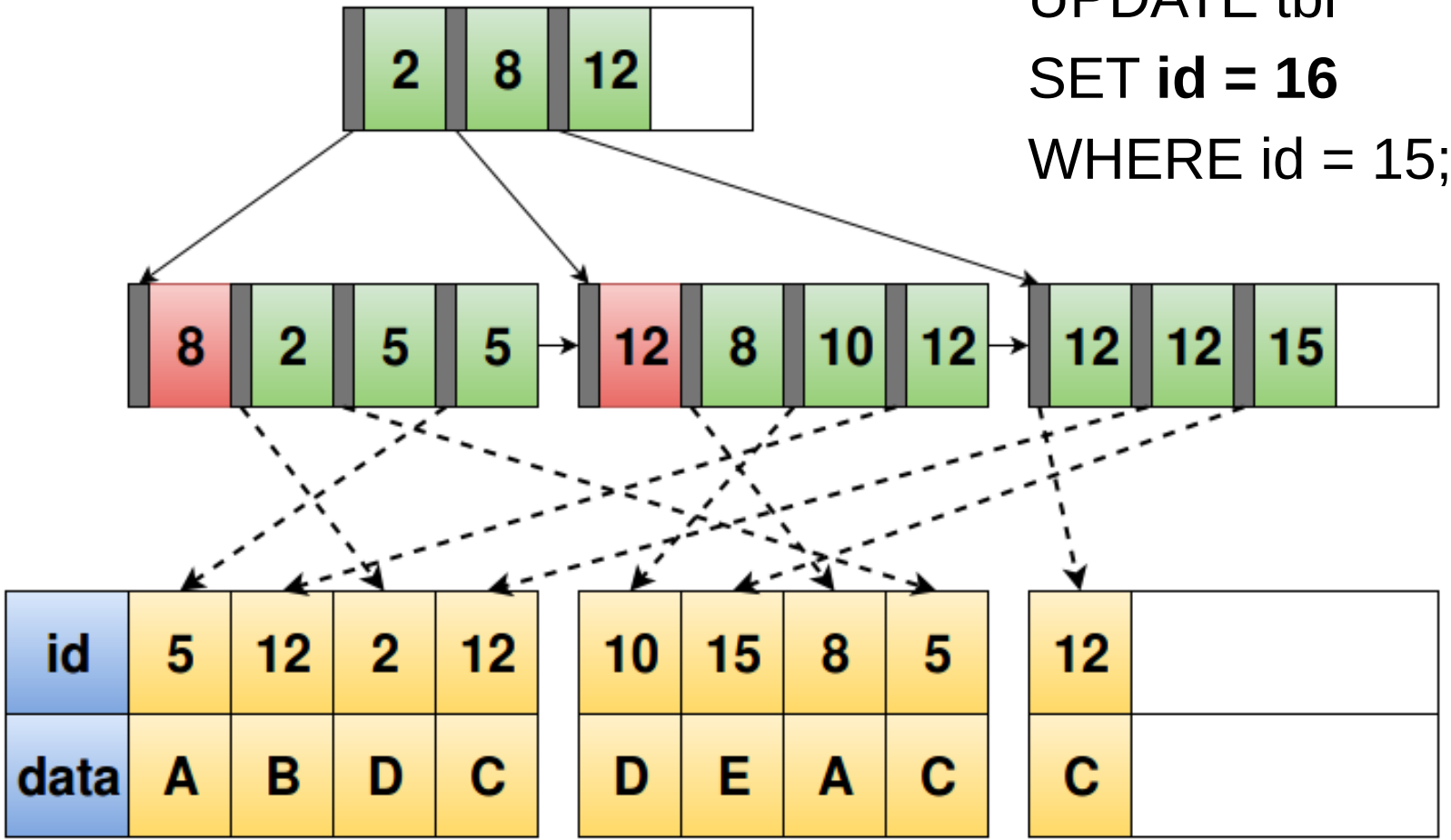


MVCC update (1)



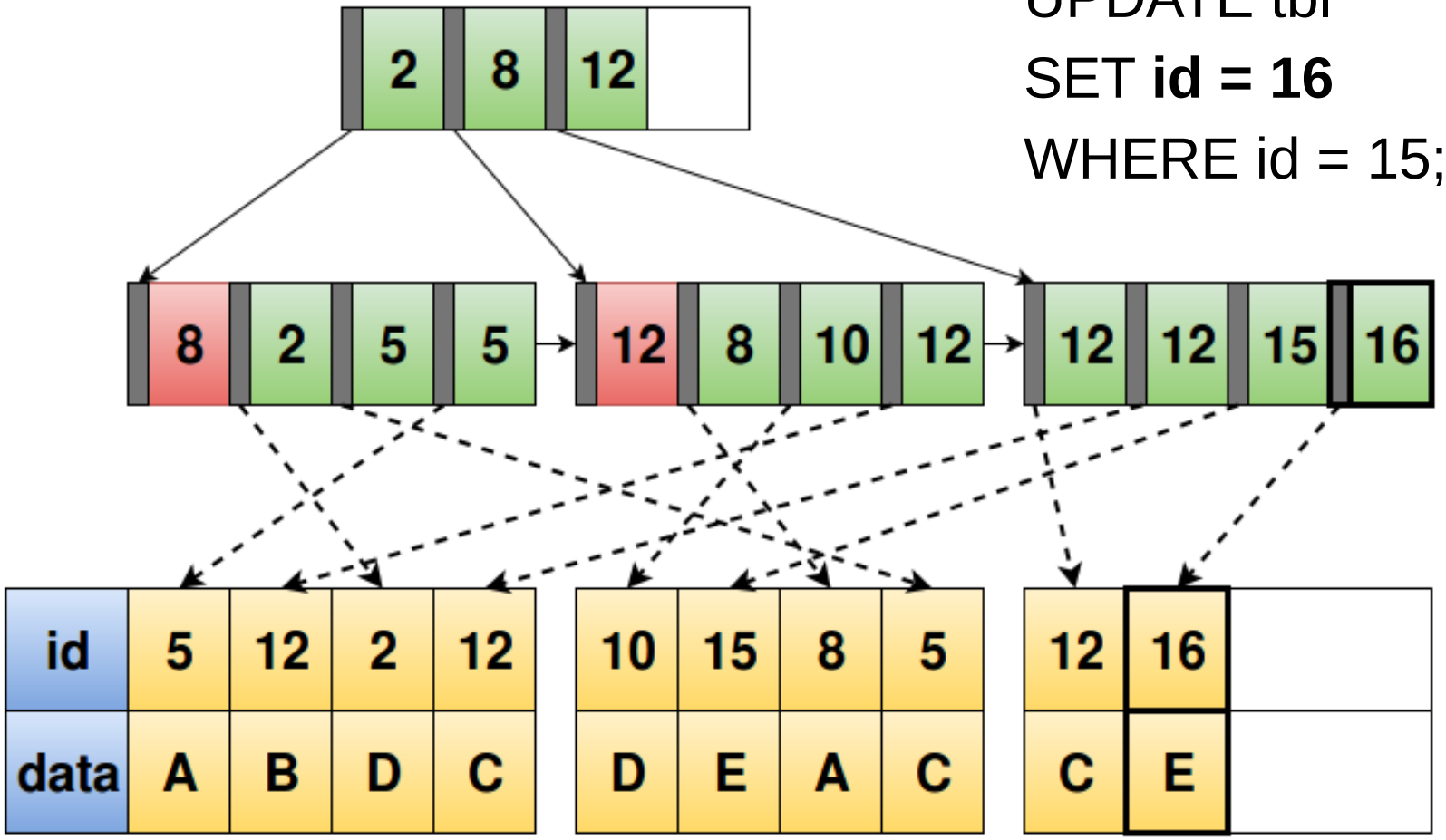
MVCC update (1)

UPDATE tbl
SET **id = 16**
WHERE id = 15;



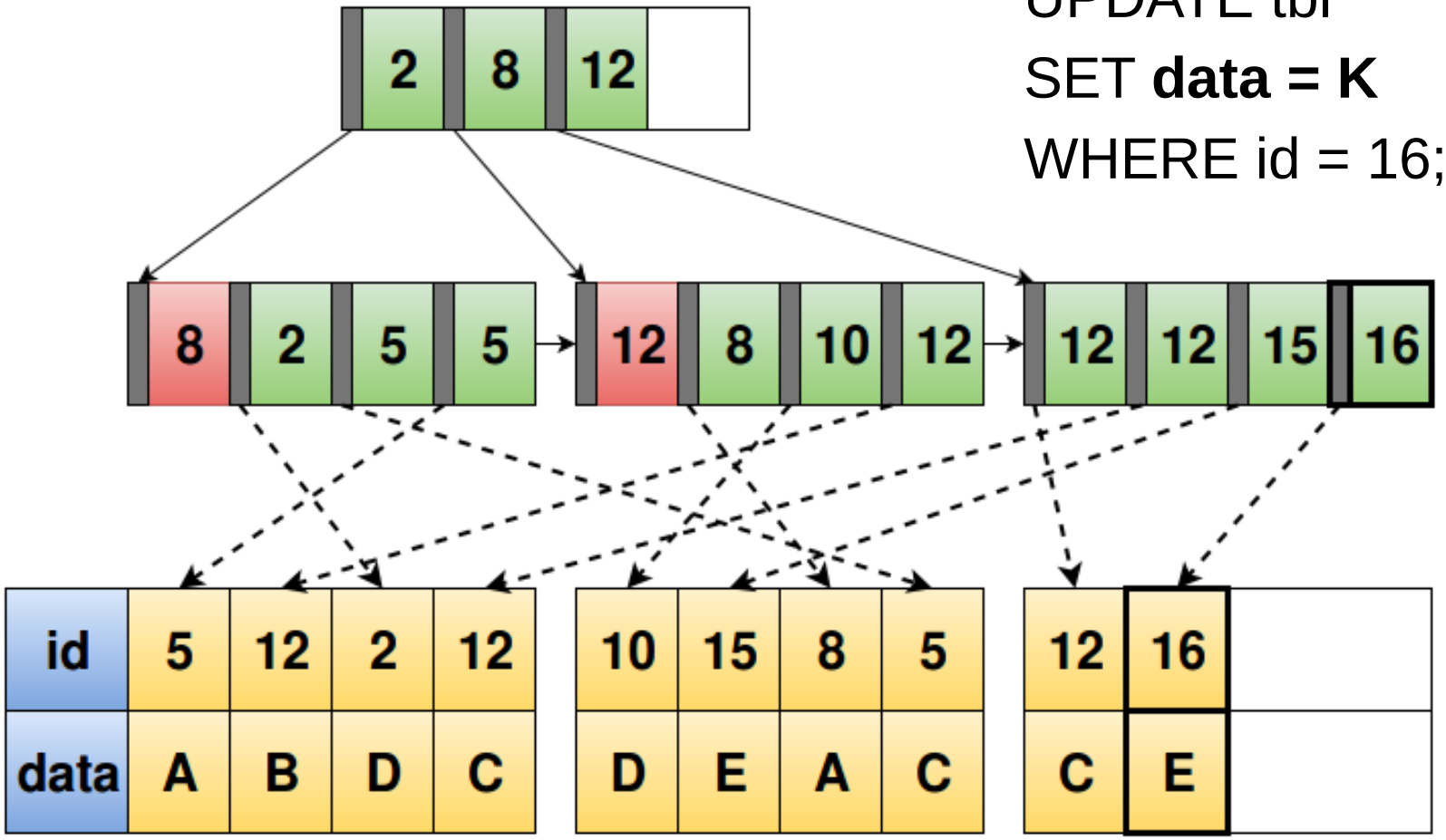
MVCC update (2)

UPDATE tbl
SET **id = 16**
WHERE id = 15;

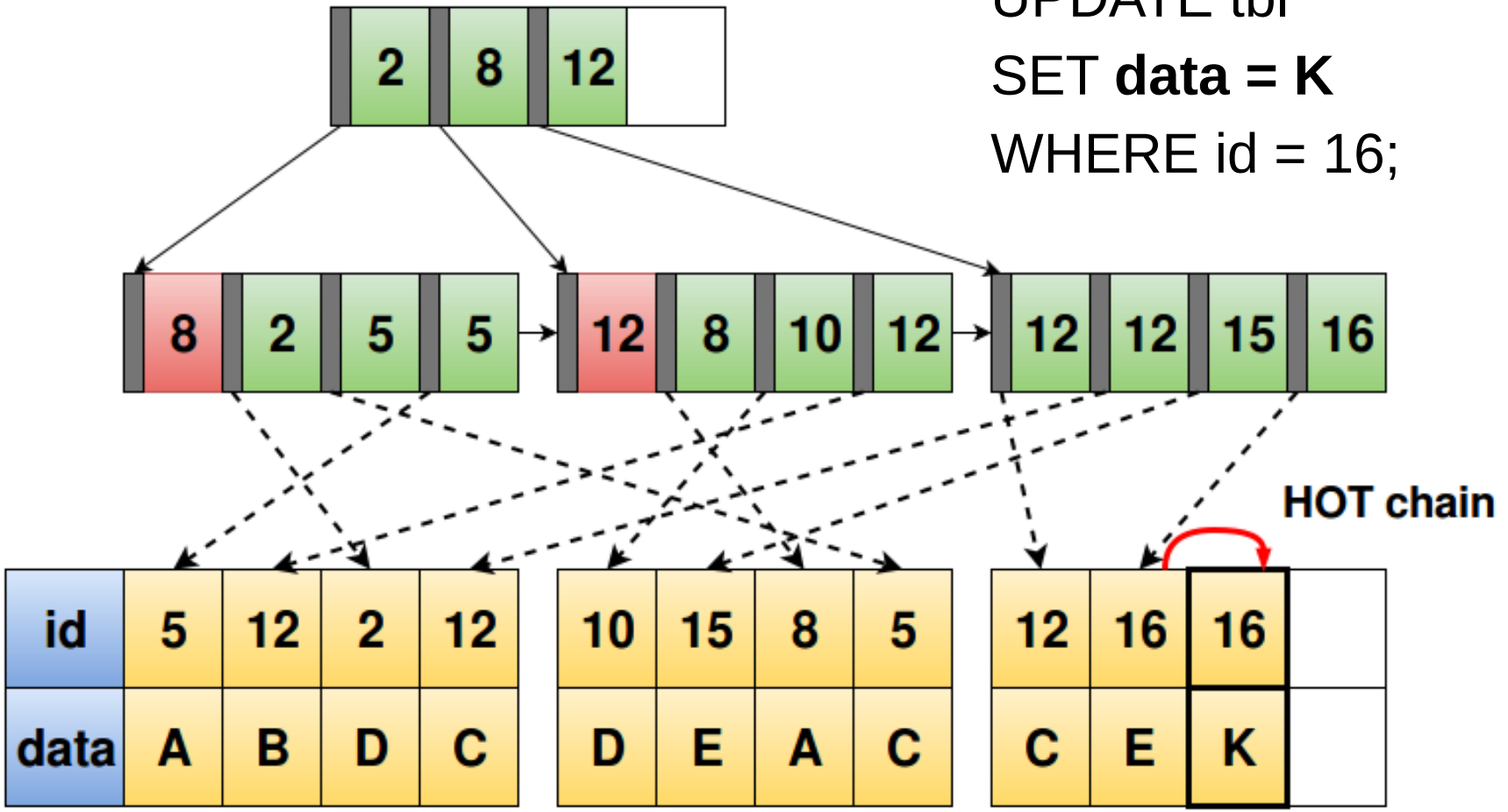


HOT update

UPDATE tbl
SET **data = K**
WHERE id = 16;



HOT update


UPDATE tbl
SET **data = K**
WHERE id = 16;
2
8
12
8
2
5
5
12
8
10
12
12
12
15
16
HOT chain
id
5
12
2
12
10
15
8
5
12
16
16
data
A
B
D
C
D
E
A
C
C
E
K

Издержки использования индексов

- MVCC update → Большой размер
  - VACUUM
  - Мониторинг индексов

Издержки использования индексов

- MVCC update → Большой размер
  - VACUUM
  - Мониторинг индексов
- Вставка каждой записи в индекс → Замедление insert/update
  - Удаление неиспользуемых индексов

Издержки использования индексов

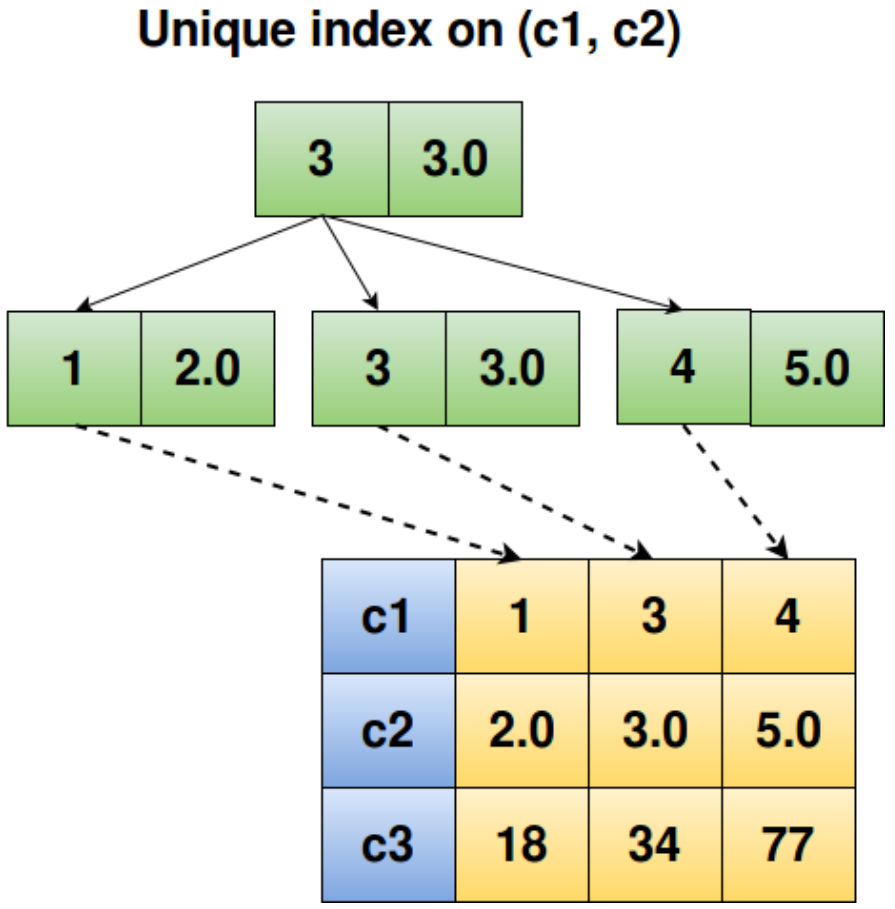
- MVCC update → Большой размер
  - VACUUM
  - Мониторинг индексов
- Вставка каждой записи в индекс → Замедление insert/update
  - Удаление неиспользуемых индексов
- Повторное использование страниц → Фрагментация индексов
  - REINDEX
  - CREATE INDEX CONCURRENTLY

Издержки использования индексов

- MVCC update → Большой размер
  - VACUUM
  - Мониторинг индексов
- Вставка каждой записи в индекс → Замедление insert/update
  - Удаление неиспользуемых индексов
- Повторное использование страниц → Фрагментация индексов
  - REINDEX
  - CREATE INDEX CONCURRENTLY
- Случайное чтение диска
  - Создавать индекс после вставки данных (если это возможно)

Новые возможности B-tree

Уникальный индекс (1)



Уникальный индекс (2)

- CREATE TABLE tbl (c1 int, c2 int, c3 int);
- CREATE UNIQUE INDEX idx ON tbl (c1, c2);
- SELECT * FROM tbl WHERE c1 > 10000 and c1 < 11000;

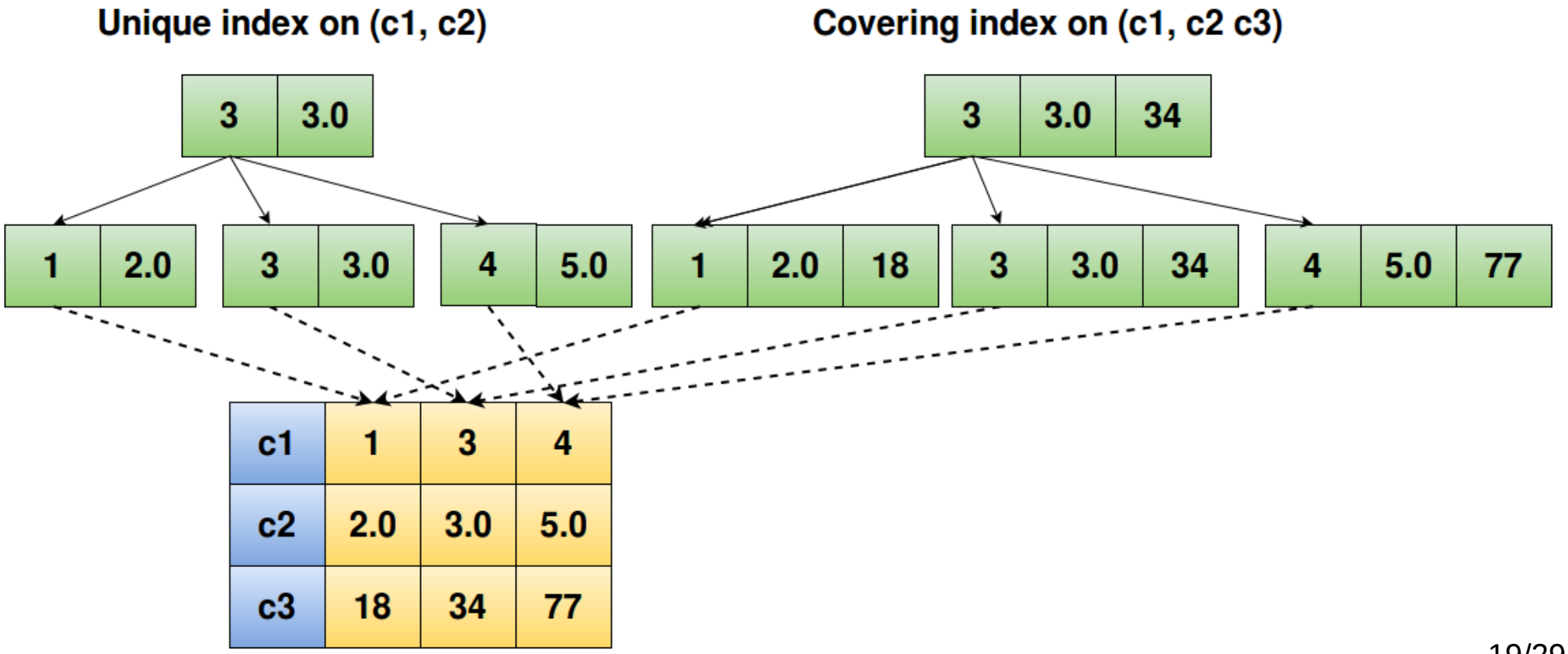
Bitmap Heap Scan on tbl (cost=5.37..354.89 rows=91 width=12) (actual time=0.046..0.161 rows=101 loops=1)
  Recheck Cond: ((c1 > 10000) AND (c1 < 11000))
  Heap Blocks: exact=101
  -> **Bitmap Index Scan** on **idx** (cost=0.00..5.34 rows=91 width=0) (actual time=0.028..0.028 rows=101 loops=1)
      Index Cond: ((c1 > 10000) AND (c1 < 11000))
Planning time: 0.105 ms
Execution time: **0.183 ms**

Уникальный и покрывающий индексы (2)

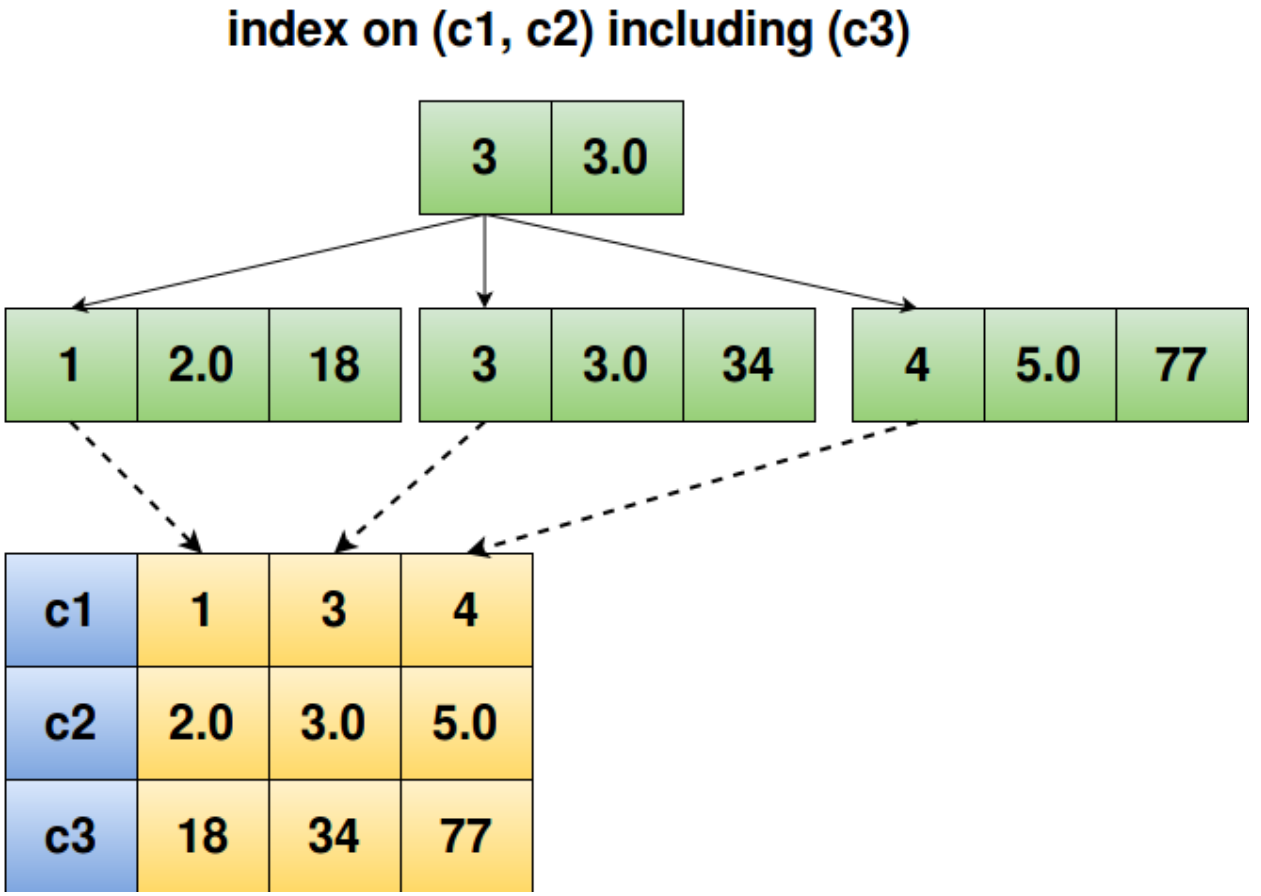
- CREATE TABLE tbl (c1 int, c2 int, c3 int);
- CREATE UNIQUE INDEX idx ON tbl (c1, c2);
- CREATE INDEX idx_covering ON tbl (c1, c2, c3);
- SELECT * FROM tbl WHERE c1 > 10000 and c1 < 11000;

Index Only Scan using **idx_covering** on tbl (cost=0.43..6.43 rows=100 width=12) (actual time=0.011..0.047 rows=124 loops=1)
Index Cond: ((c1 > 10000) AND (c1 < 11000))
Heap Fetches: 0
Planning time: 0.132 ms
Execution time: **0.074 ms**

Уникальный и покрывающий индексы (1)



Индекс с включенными столбцами (1)



Индекс с включенными столбцами (2)

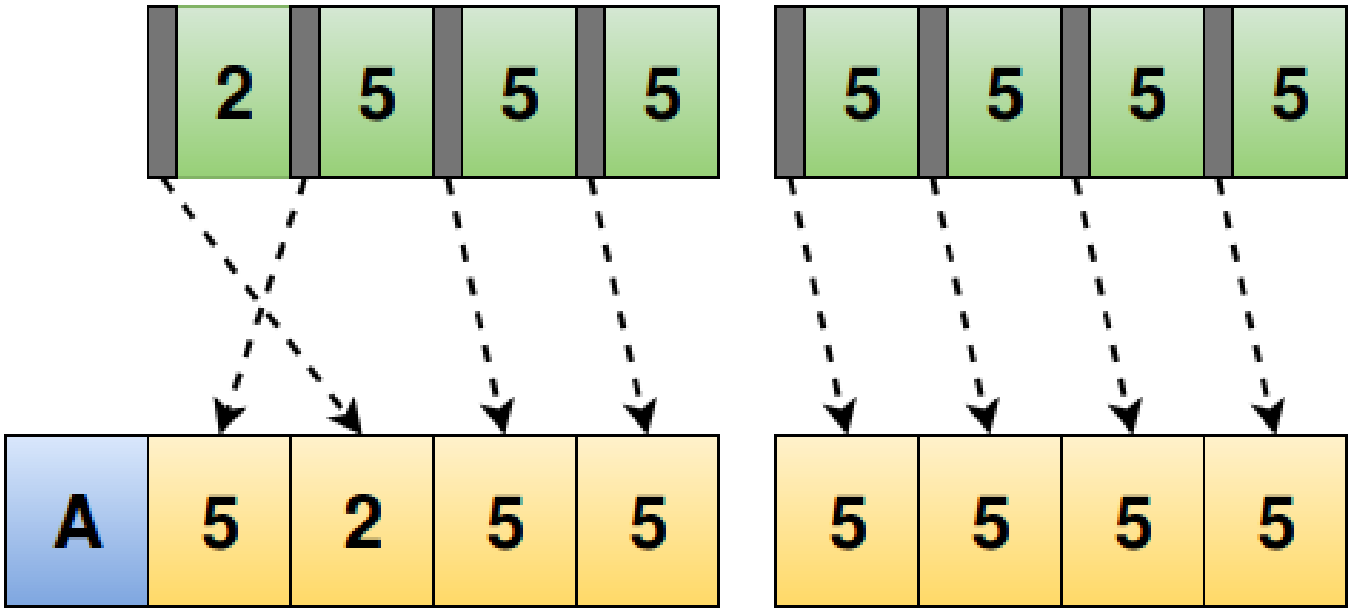
- CREATE TABLE tbl (c1 int, c2 int, c3 int);
- CREATE UNIQUE INDEX idx ON tbl (c1, c2) **INCLUDING** (c3);
- SELECT * FROM tbl WHERE c1 > 10000 and c1 < 11000;

Index Only Scan using **idx** on tbl (cost=0.43..6.53 rows=105 width=12) (actual time=0.007..0.034 rows=113 loops=1)
Index Cond: ((c1 > 10000) AND (c1 < 11000))
Heap Fetches: 0
Planning time: 0.095 ms
Execution time: **0.055 ms**

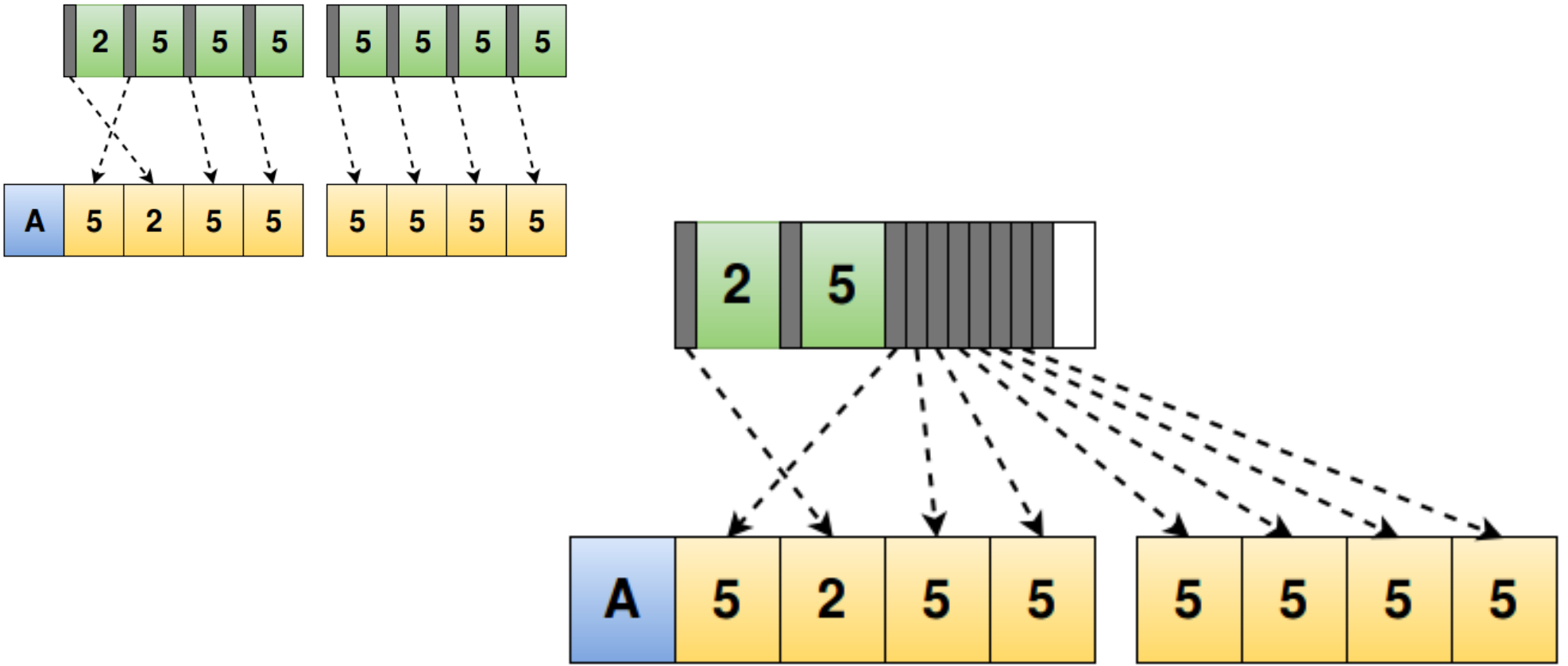
Индексы с включенными столбцами

- Меньше размер индекса
- Меньше издержек на обновление
- Быстрее планирование и поиск
- Для включенных столбцов не нужен opclass
- Фильтр по включенным столбцам

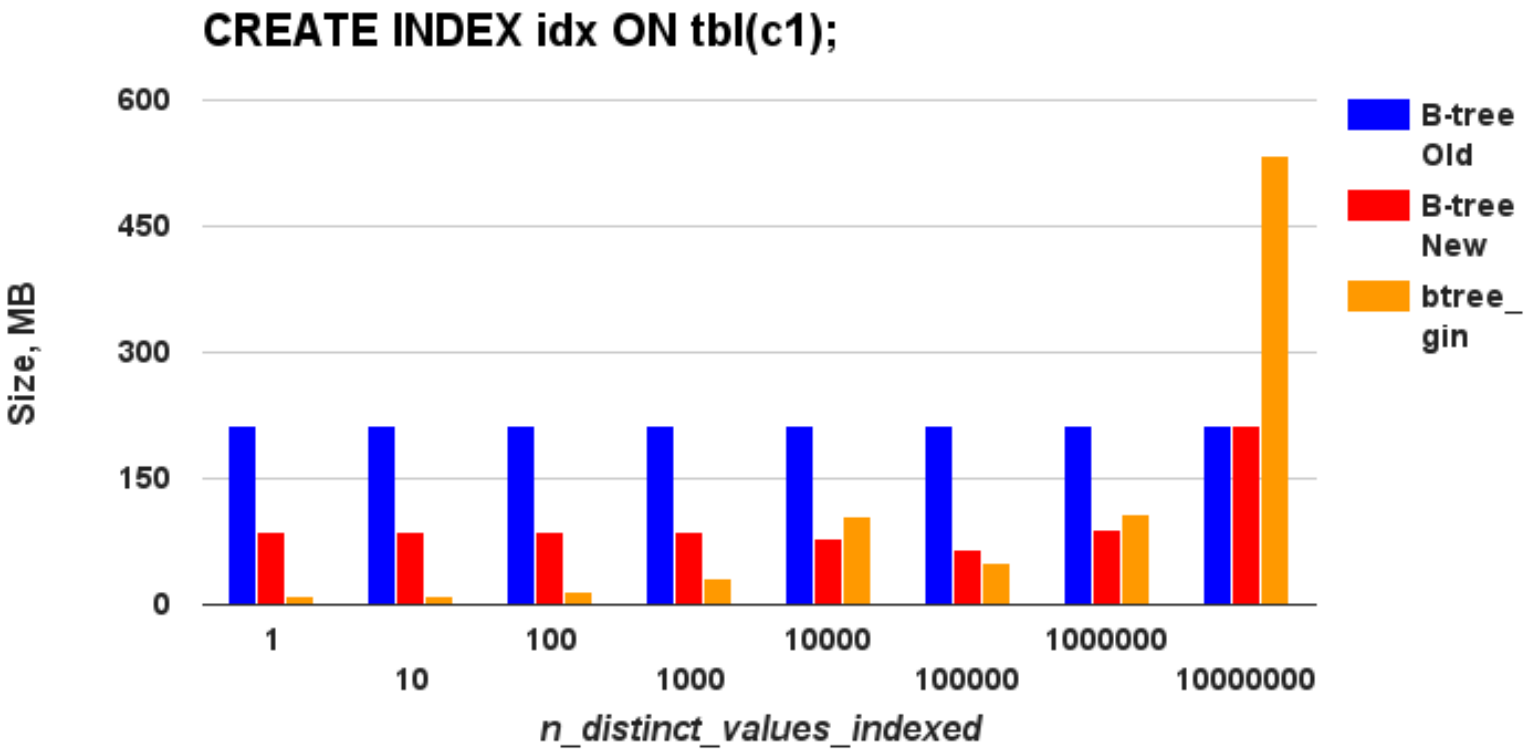
Сжатие B-tree индексов



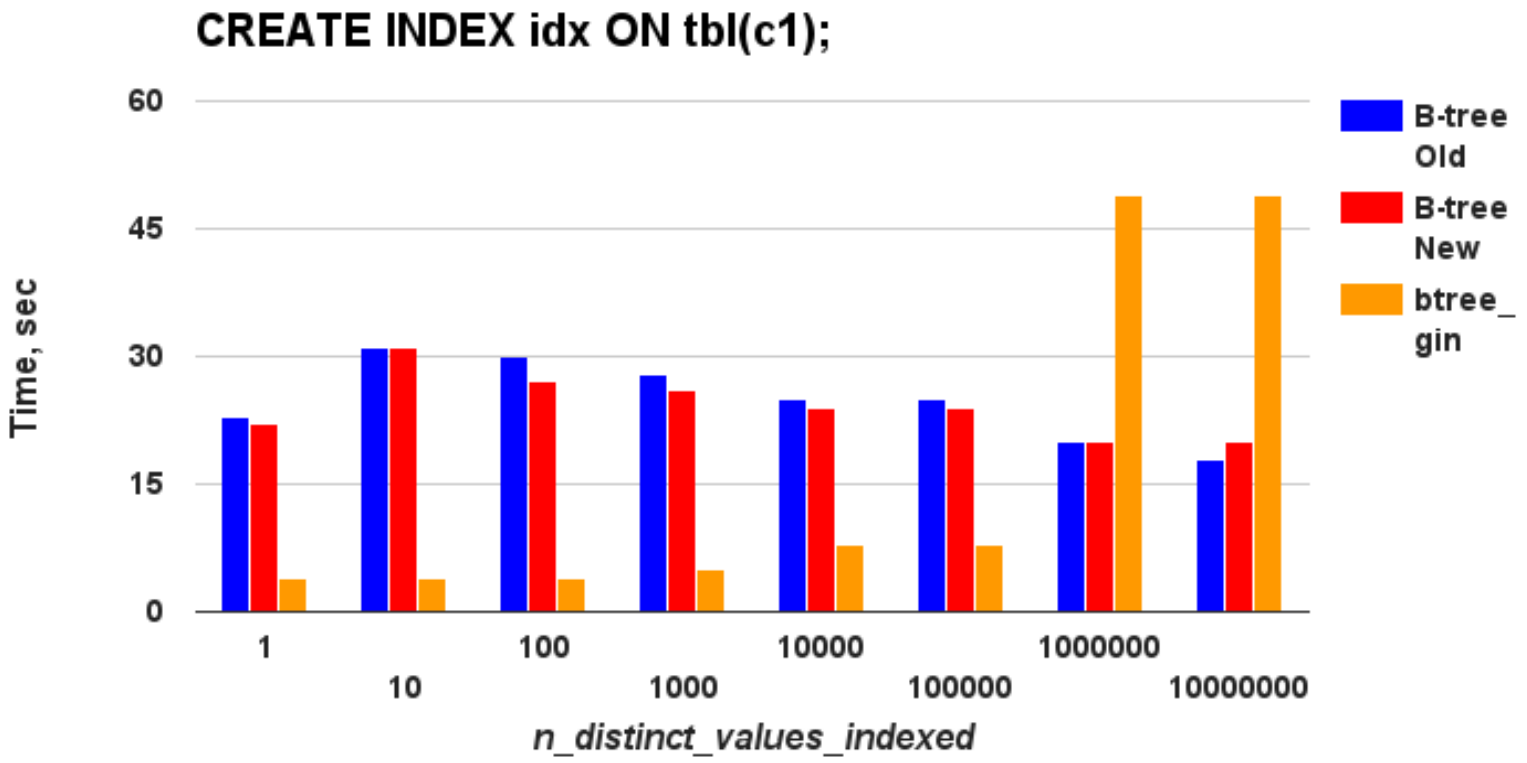
Сжатие B-tree индексов



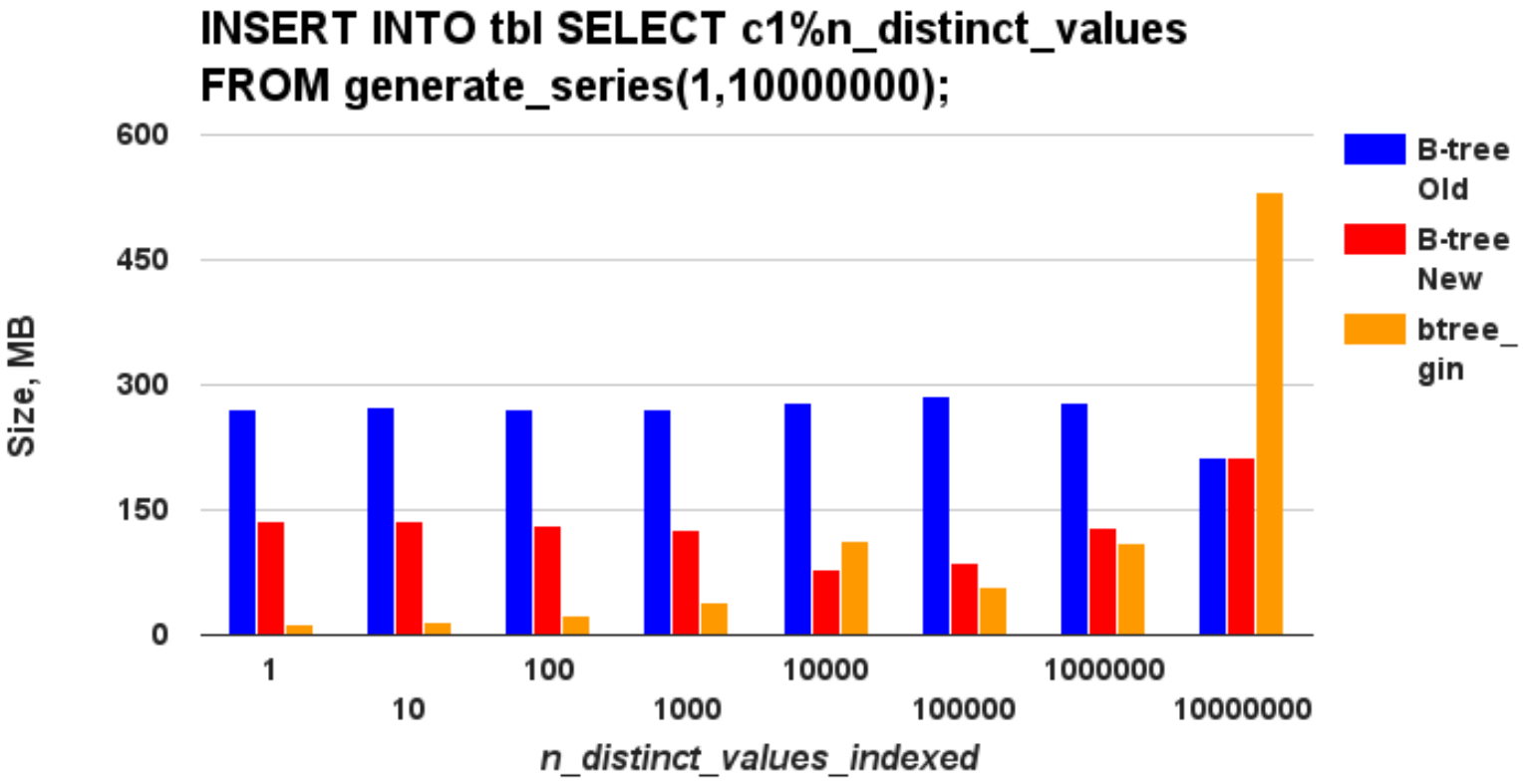
Сжатие B-tree индексов



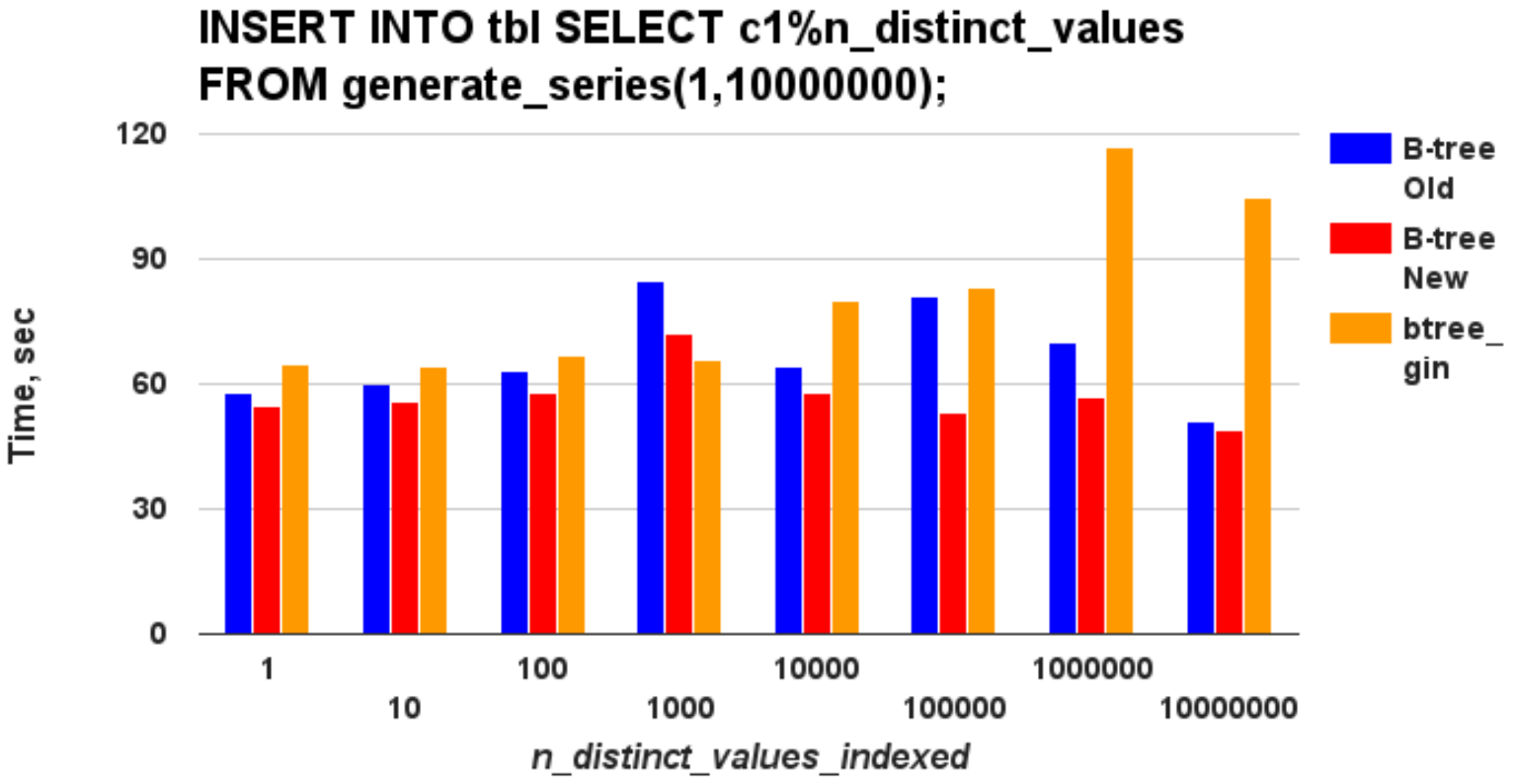
Сжатие B-tree индексов



Сжатие B-tree индексов



Сжатие B-tree индексов



Postgres PROFESSIONAL

Спасибо за внимание!

a.lubennikova@postgrespro.ru