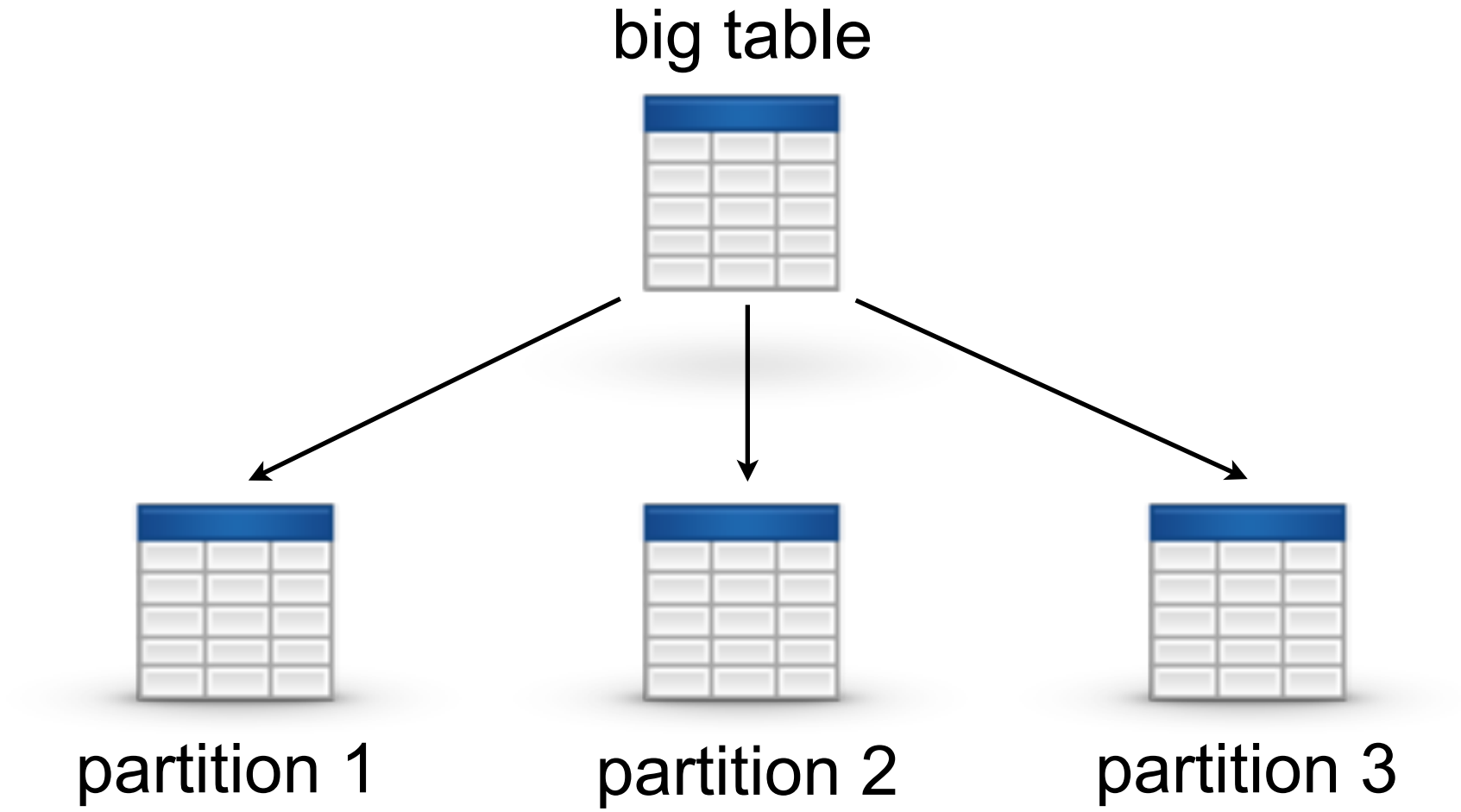
Секционирование без границ

Мусин Ильдар

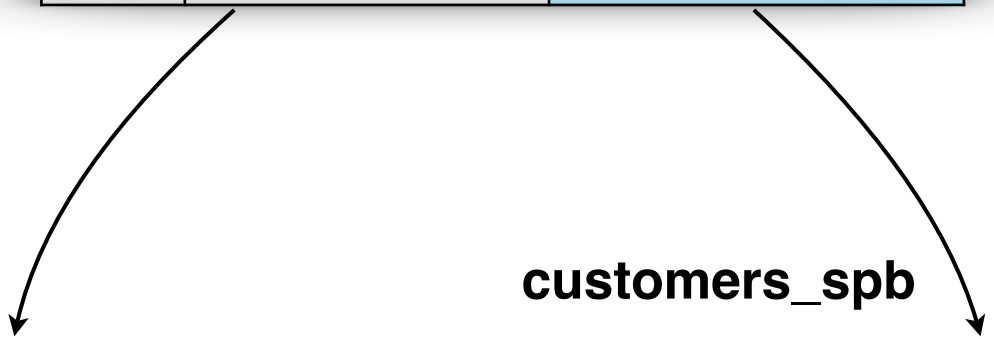
Секционирование



Пример секционирования

customers

id	name	city
1	Иванов И.	Москва
2	Сидоров А.	Санкт-Петербург
3	Петров В.	Москва
4	Григорьев С.	Санкт-Петербург



customers_msc

id	name	city
1	Иванов И.	Москва
3	Петров В.	Москва

customers_spb

id	name	city
2	Сидоров А.	Санкт-Петербург
4	Григорьев С.	Санкт-Петербург

Преимущества секционирования

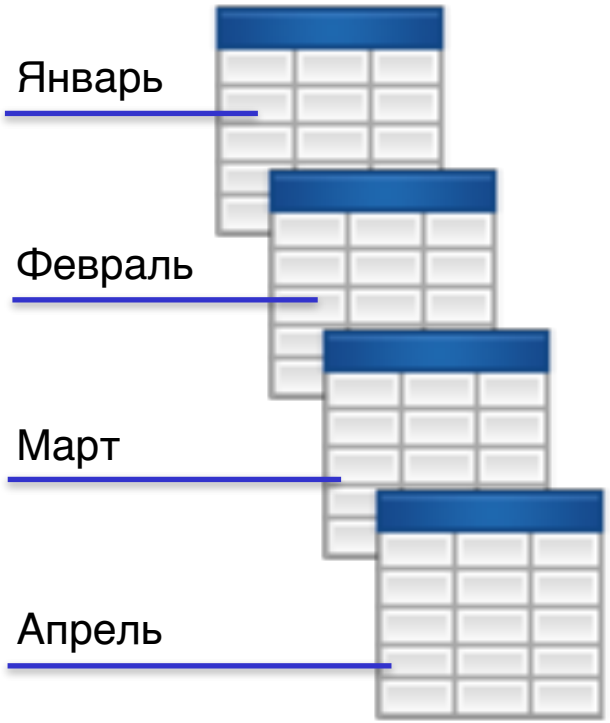
- Рост производительности при условии, что наиболее часто используемые данные находятся в небольшом числе секций.
- Использование последовательного доступа к небольшим секциям вместо индекса или случайного доступа к одной большой таблице.
- Управление большими объемами данных.
- Редко используемые данные могут быть вынесены на более дешевые и медленные носители.

Когда нужно секционировать?

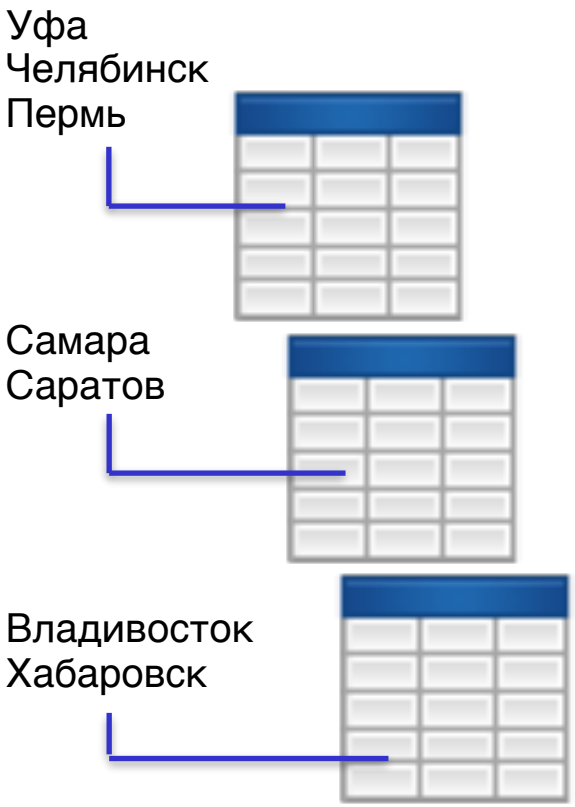
- Эмпирическое правило: размер таблицы превышает размер физической памяти.
- Таблица содержит исторические данные, а новые записи добавляются в последнюю секцию.
- Содержимое таблицы должно быть распределено между носителями информации или между серверами.

Виды секционирования

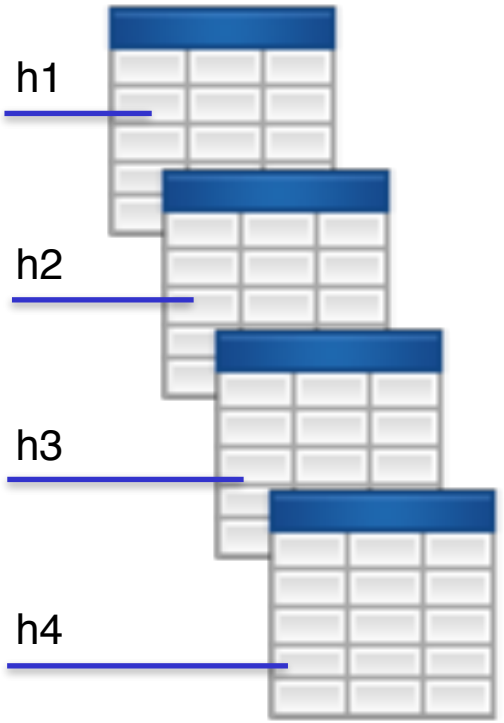
RANGE



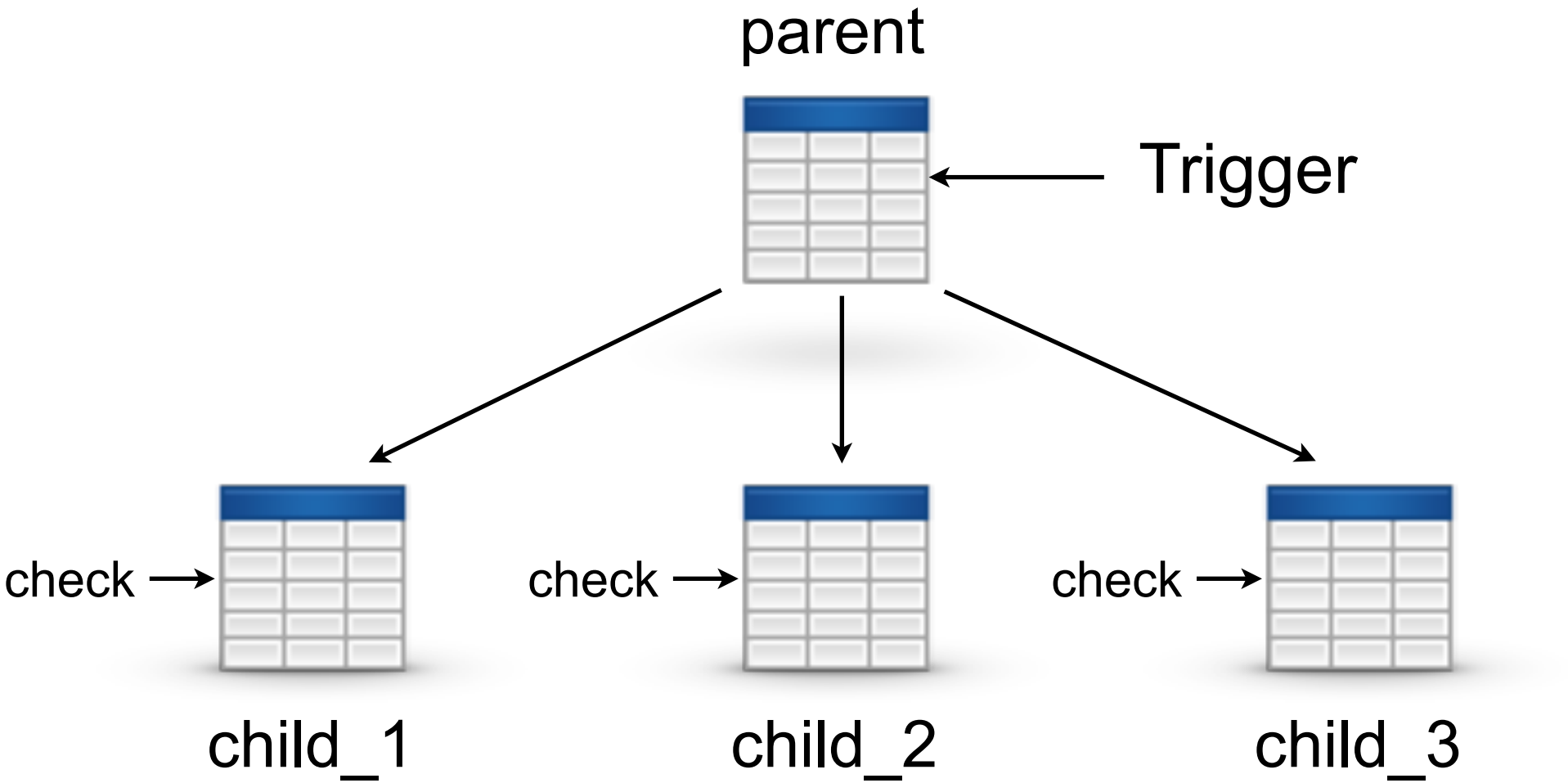
LIST



HASH



Секционирование в PostgreSQL



Виды секционирования в PostgreSQL

- **RANGE**

  CHECK(dt_created >= '2015-01-01' AND
  dt_created < '2016-01-01')

- **LIST**

  CHECK(city IN (‘Уфа’, ‘Оренбург’, ‘Челябинск’))

- **HASH** (?)

  CHECK(id % 10 = 0)

  …

  CHECK(id % 10 = 9)

  SELECT * FROM my_table
  WHERE id = 5 AND (id % 10) = (5 % 10)

Проблемы

- Полный перебор при поиске подходящих секций
- Нет встроенной поддержки HASH-секционирования
- Ручное управление секциями

Решение

Определить структуры данных, над которыми можно выполнять эффективный поиск

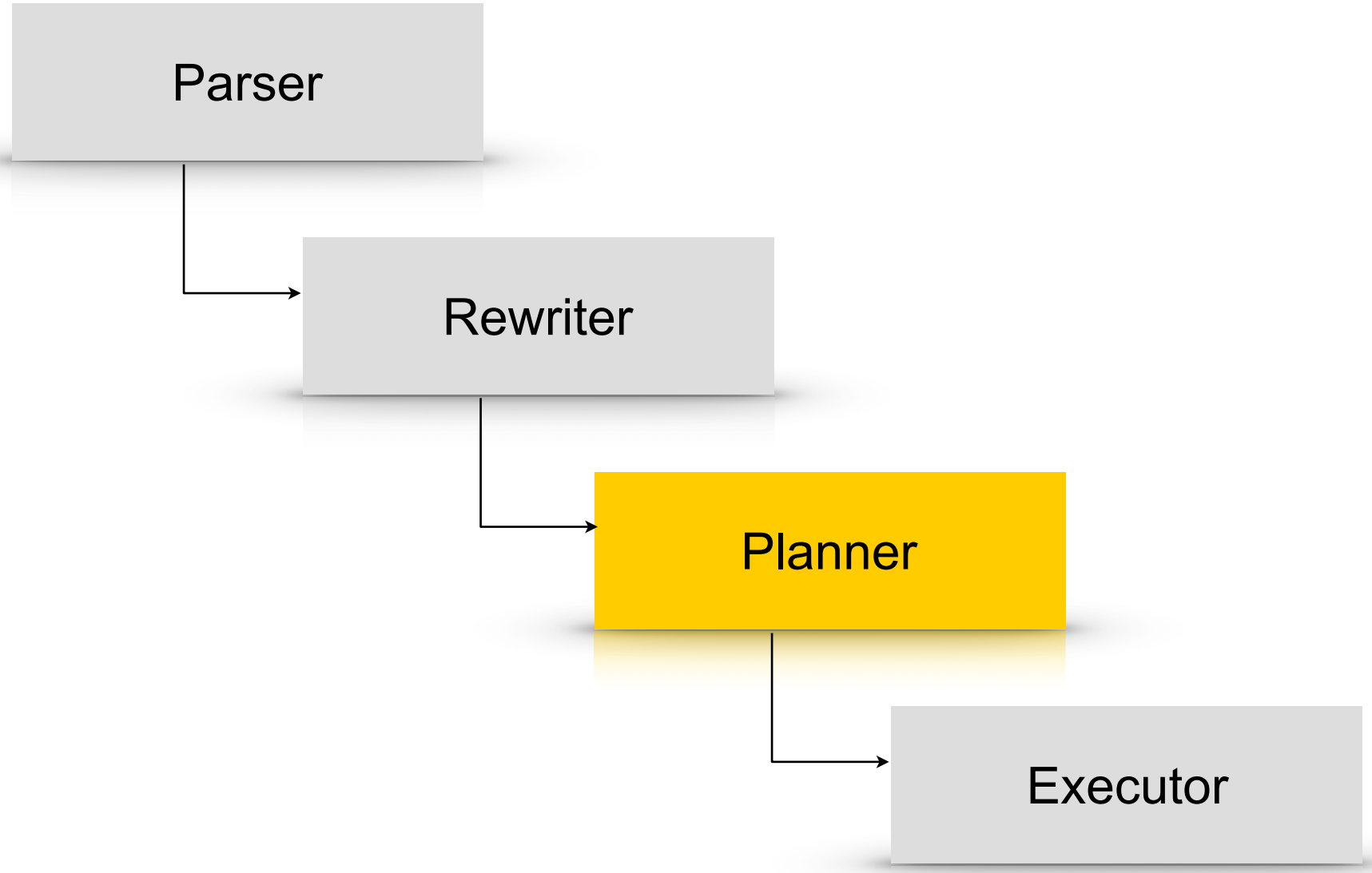
RANGE

- бинарный поиск в сортированном массиве
- поиск в бинарном дереве $O(\log_2 N)$

HASH, LIST

- хеш-таблица $\approx O(1)$

Этапы обработки запроса



Используемые средства

Hook-функции — лазейки, используемые для внедрения в код СУБД:

- planner_hook
- set_rel_pathlist_hook

Path — прототип планов. Для каждой таблицы, используемой в запросе, строятся всевозможные способы обхода, из которых выбирается лучший. На его основе строится узел плана.

Инициализация кэша

parent OID
partitioning type
attribute
attribute type
children

hash

child OID	486876	486883	486890	486898

range

сортировка по возрастанию min

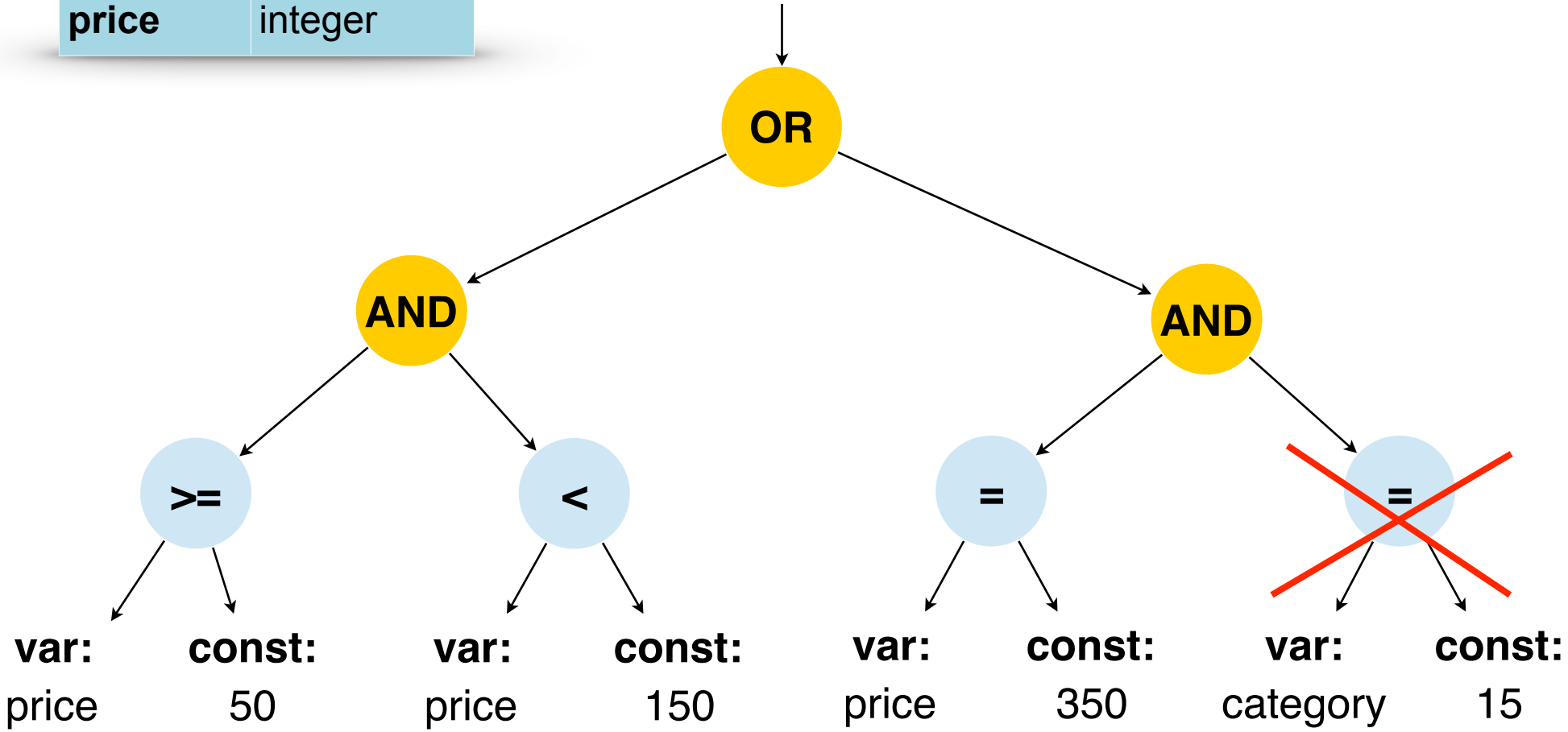
min	0	100	200	300
max	100	200	300	400
child OID	486876	486883	486890	486898

Дерево выражений

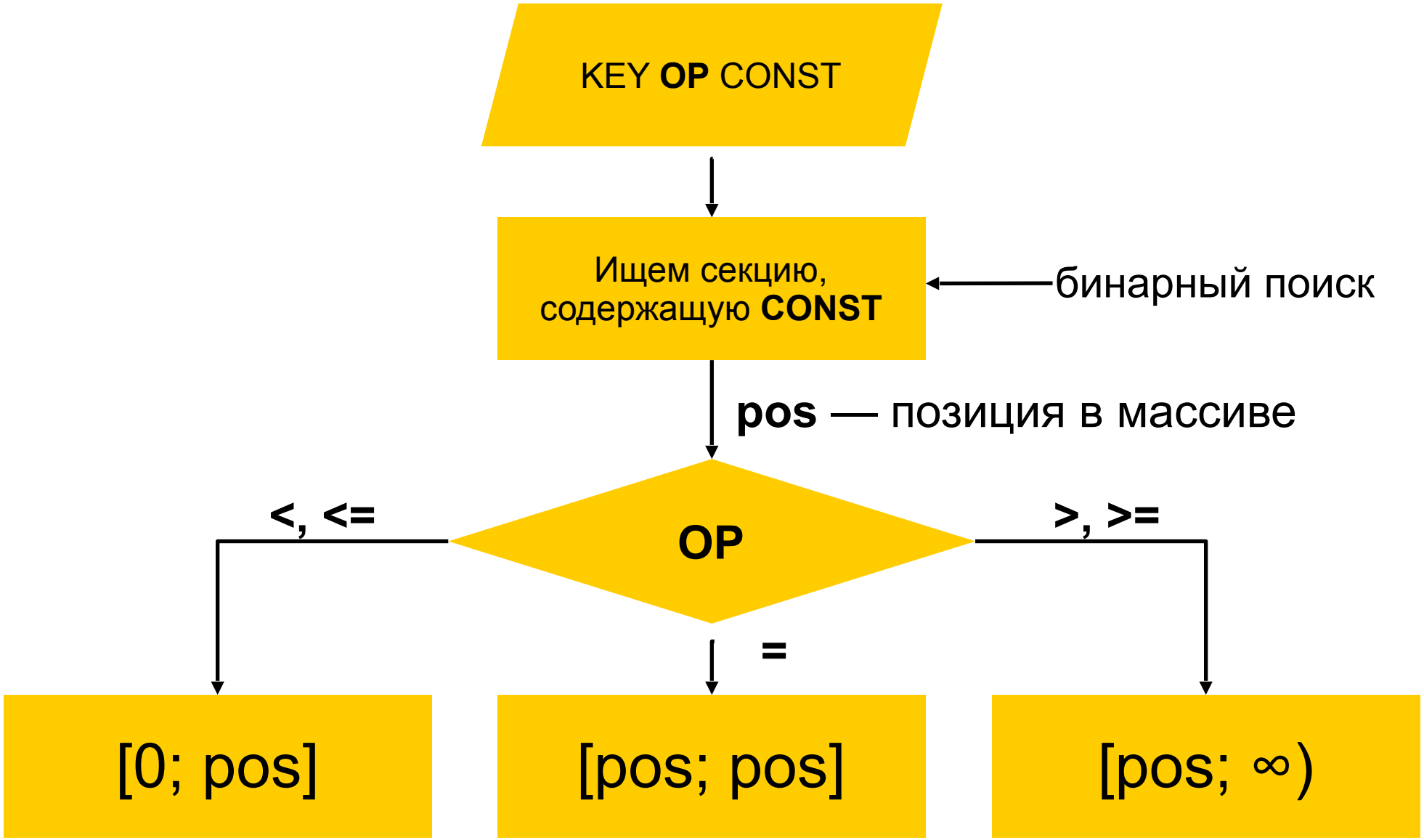
products

id	integer
category	integer
name	varchar(32)
price	integer

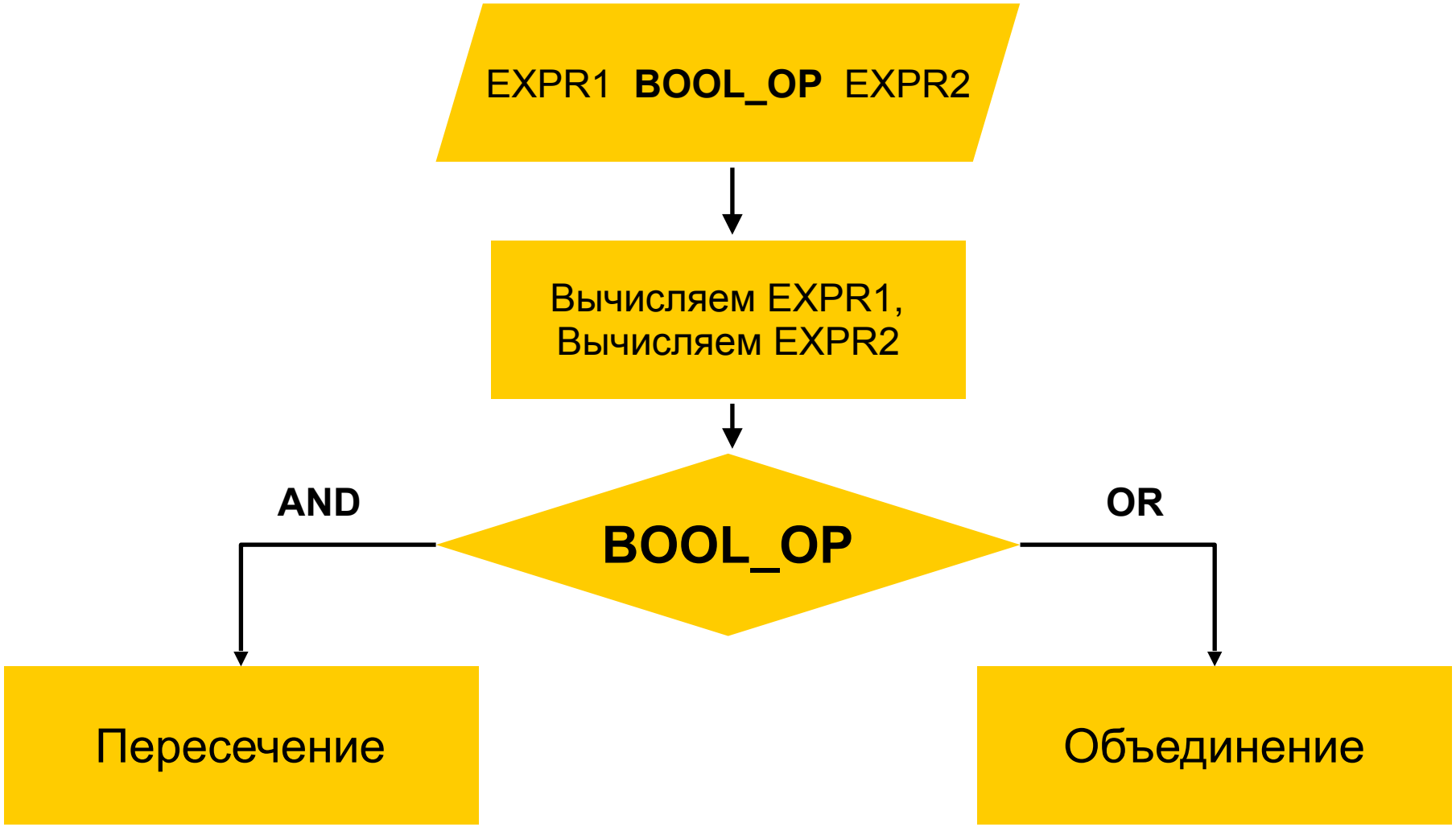
```
SELECT *
FROM products
WHERE (price >= 50 AND price <= 150)
      OR (price = 350 AND category=15)
```



Поиск RANGE-секции

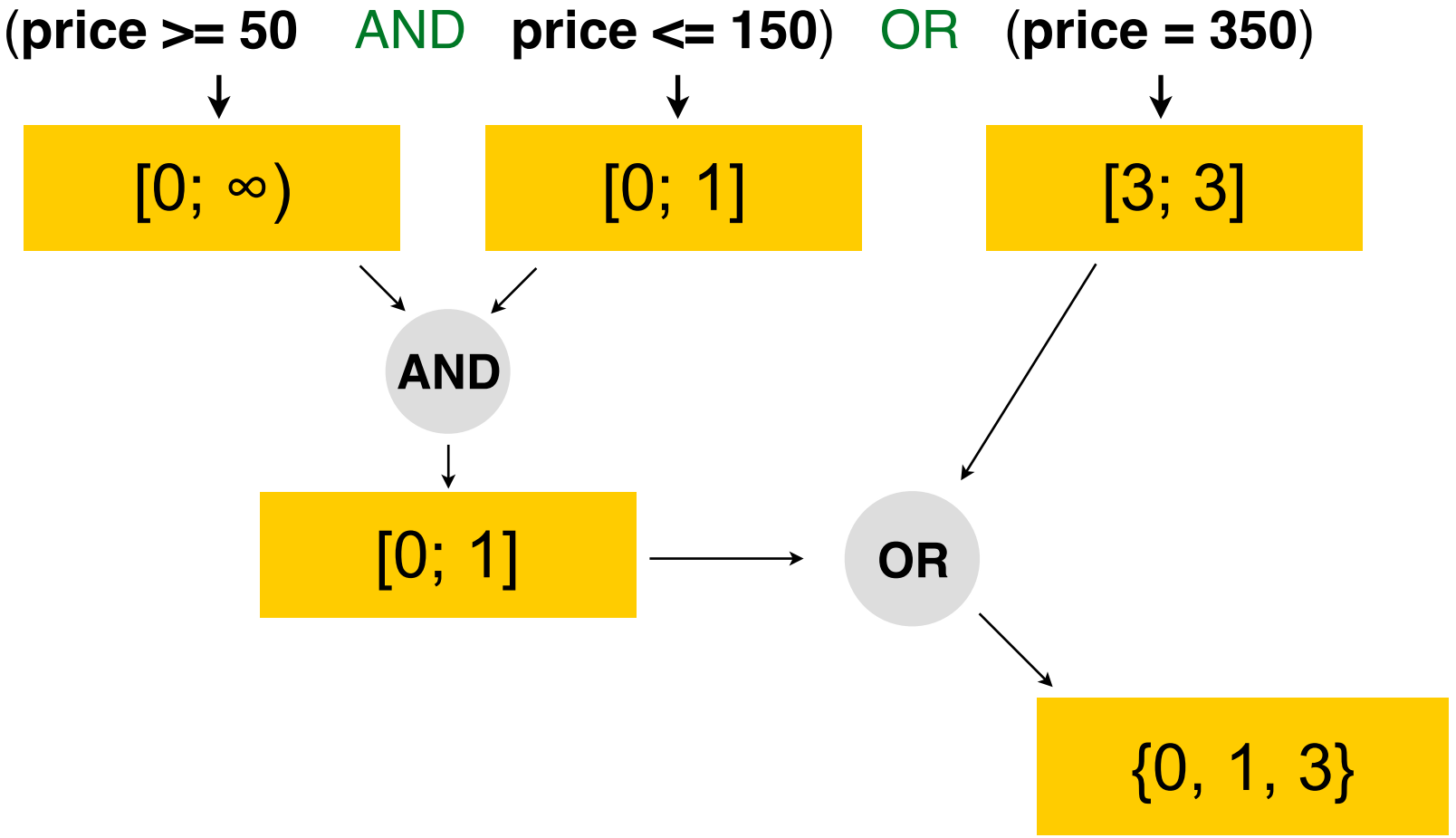


Обработка булевых операторов

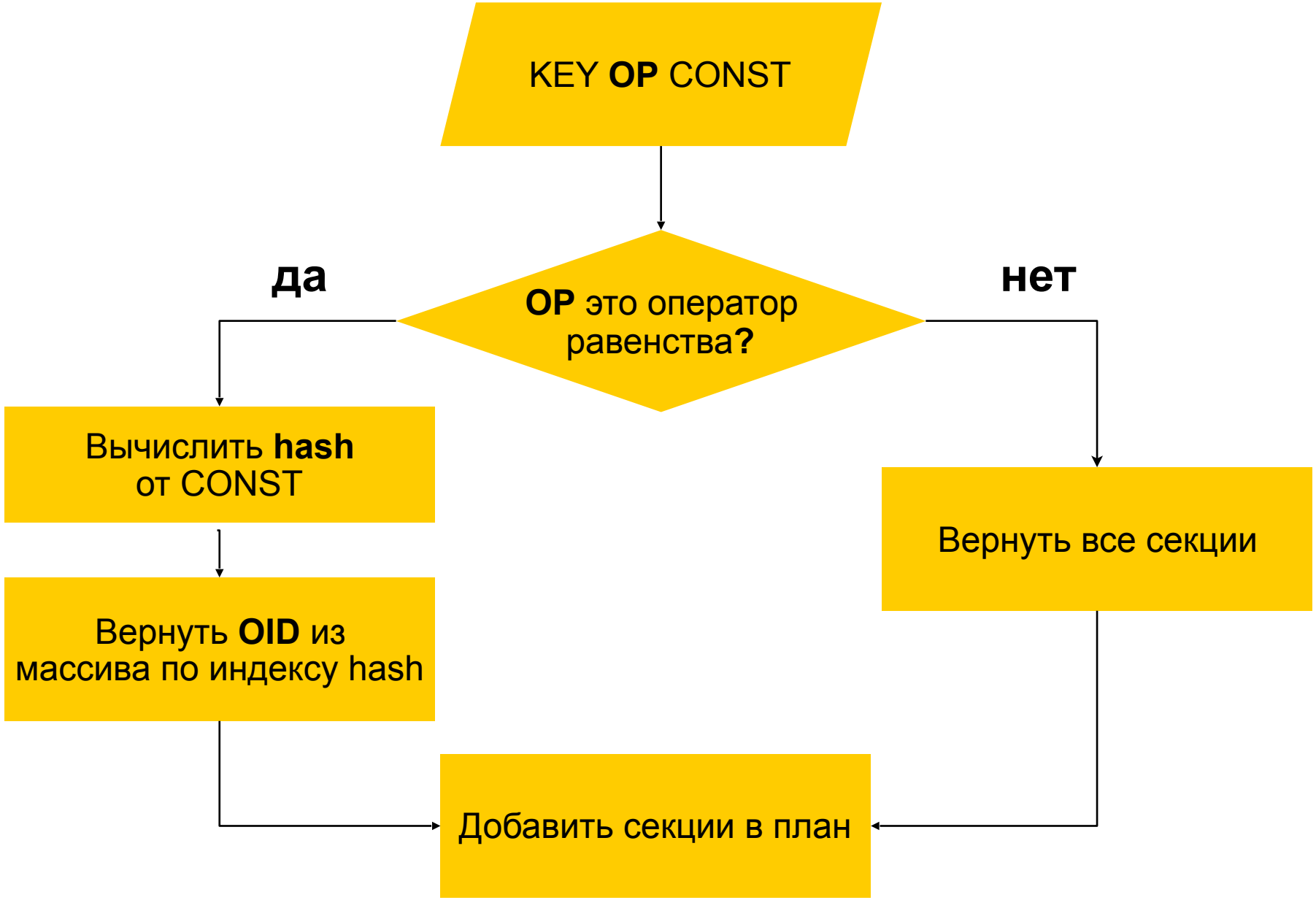


Пример

	0	1	2	3
min	0	100	200	300
max	100	200	300	400
child OID	486876	486883	486890	486898



Поиск HASH-секций



Расширение pg_pathman

- Оптимизированный механизм построения планов для секционированных таблиц
- Функции для управления секциями
- Триггеры на вставку

Эксперимент для HASH

Время планирования в зависимости от количества секций в разбиении

Таблица:

```
CREATE TABLE hash_test(id INTEGER);
```

Количество созданных секций:

2, 4, 8, 16, 32, 64, 128, 512, 1024, 2048

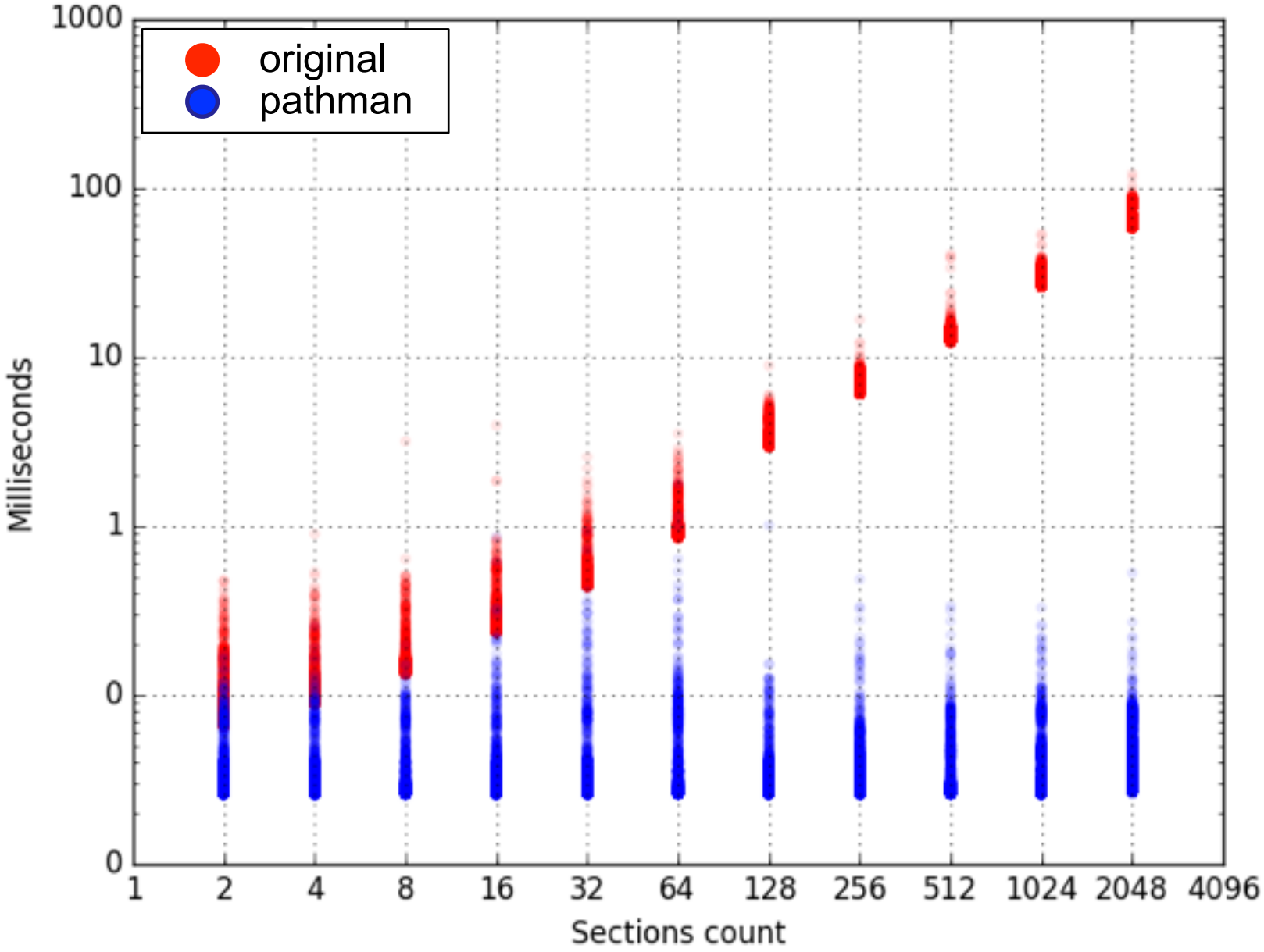
Количество запросов:

10000 на каждой конфигурации

```
EXPLAIN ANALYZE
SELECT * FROM hash_test
WHERE id = 10 AND (id%<N> = 10%<N>);
```

(где <N> - количество секций)

Эксперимент для HASH



Эксперимент для RANGE

Время планирования в зависимости от количества секций в разбиении

Таблица:

```
CREATE TABLE range_test(id INTEGER);
```

Количество созданных секций:

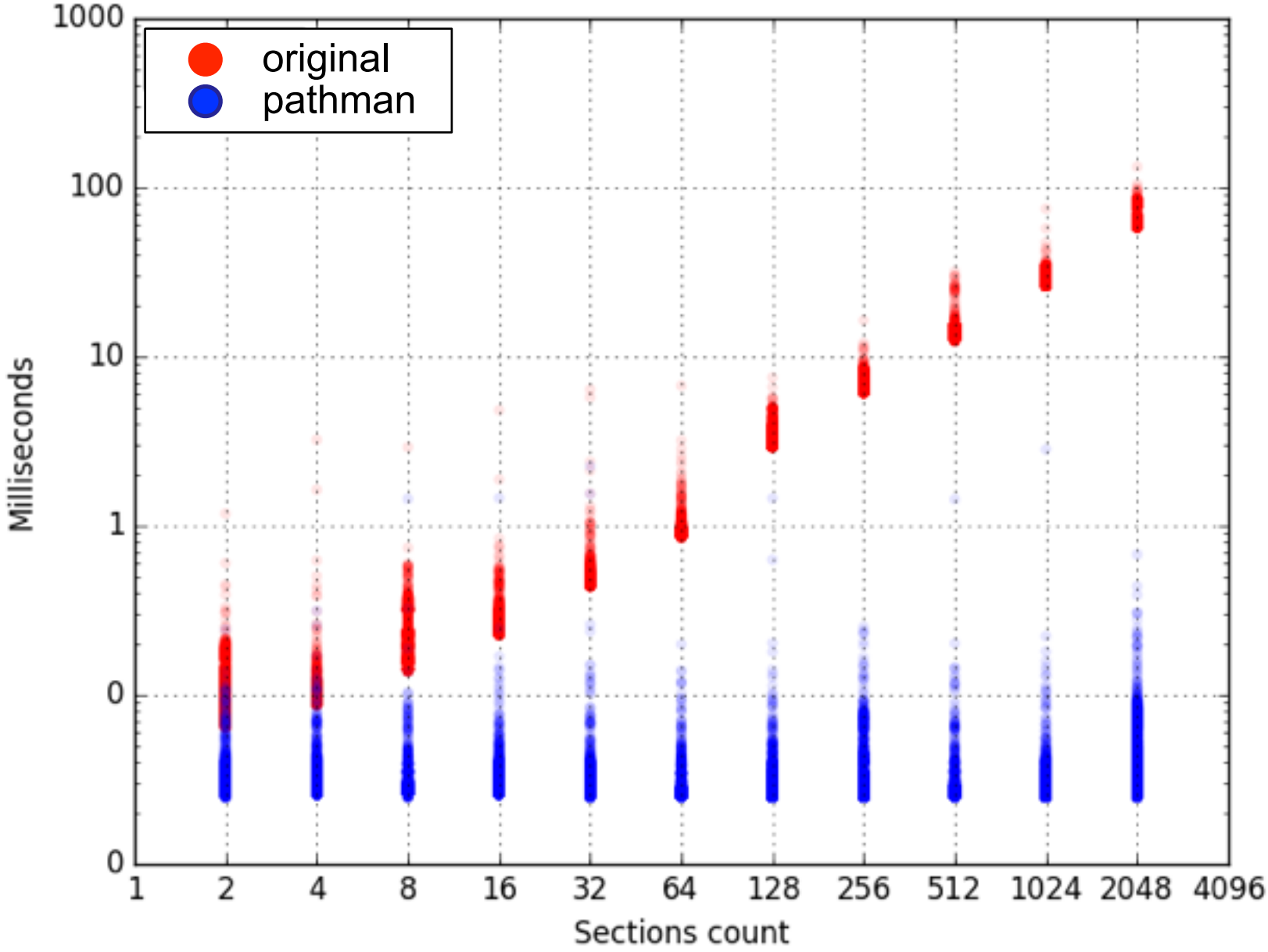
2, 4, 8, 16, 32, 64, 128, 512, 1024, 2048

Количество запросов:

10000 на каждой конфигурации

```
EXPLAIN ANALYZE
SELECT * FROM range_test
WHERE id = 10;
```

Эксперимент для RANGE



Эксперимент для RANGE

Время планирования в зависимости от количества секций в плане

Таблица:

```
CREATE TABLE range_test(id INTEGER);
```

Количество созданных секций:

512

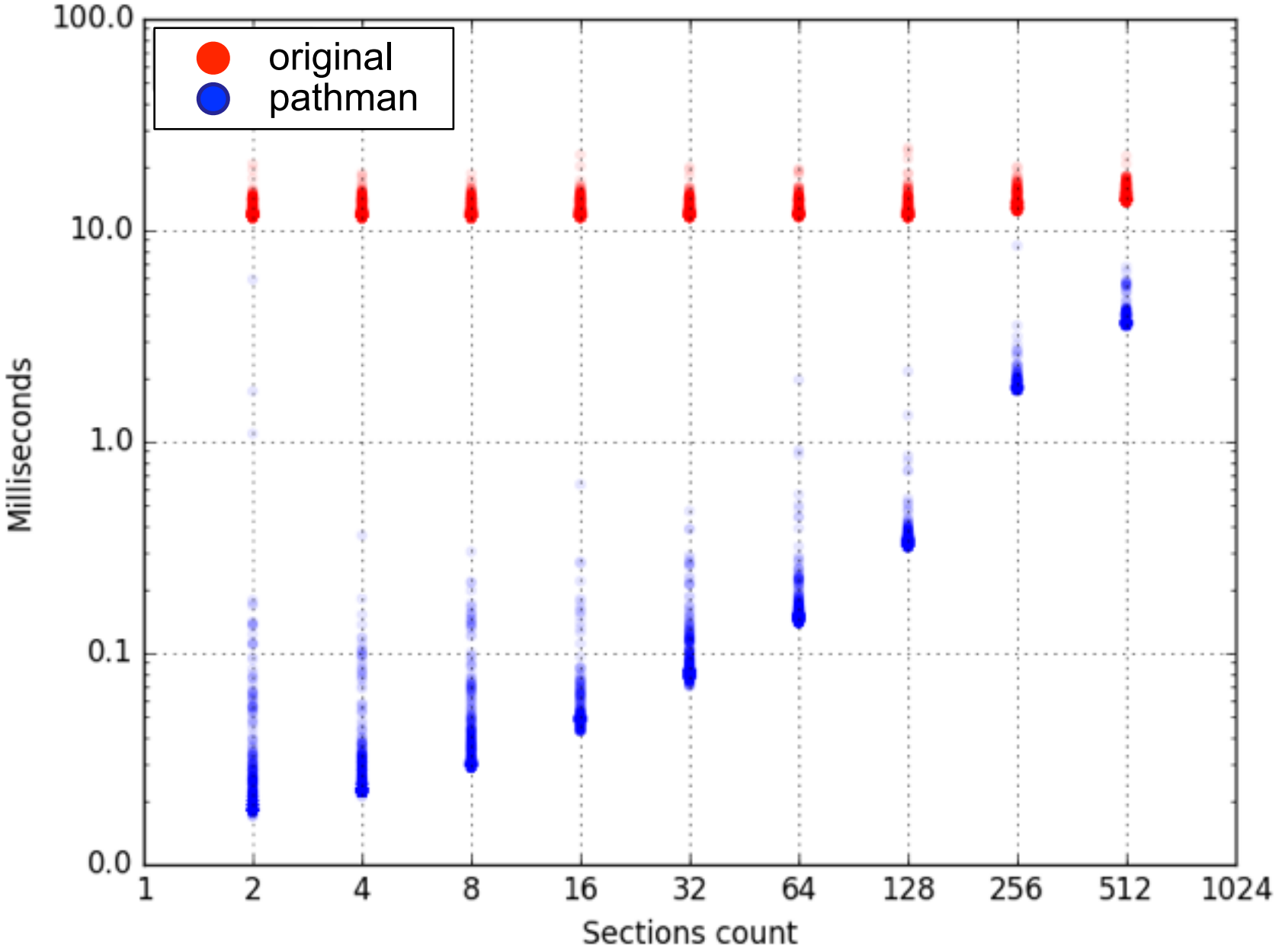
Количество секций, попадающих в план:

2, 4, 8, 16, 32, 64, 128, 256, 512

Количество запросов:

1000 на каждой конфигурации

Эксперимент для RANGE



Postgres PROFESSIONAL

Спасибо за внимание!

github.com/zilder/pg_pathman

Контакты:

i.musin@postgrespro.ru

+7 929 929 6075