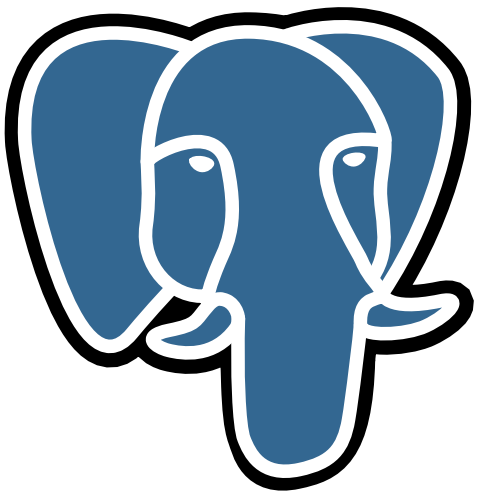
2018-02-02

GIN в Postgres и много ядер

Константин Пан
kvapen@gmail.com

PostgreSQL



Серверная СУБД

- с 1995 года
- open source, сообщество
- SQL, GIS, JSON
- куча расширений

GIN назван в честь

GIN назван в честь

Должно бы быть GENIE

GIN назван в честь

GIN назван в честь

Другие индексы в Postgres

GIN назван в честь

Другие индексы в Postgres: VODKA

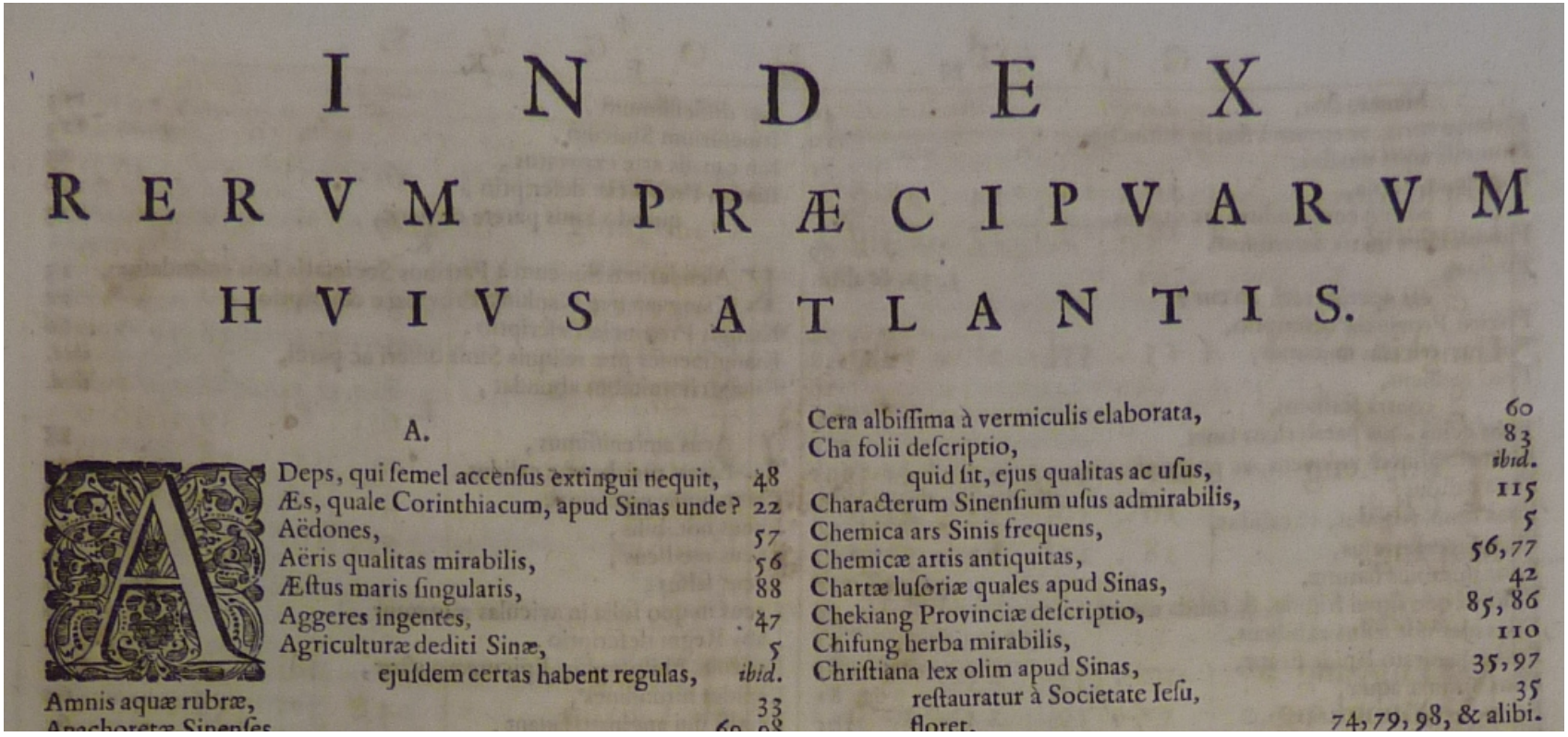
GIN назван в честь

Другие индексы в Postgres: VODKA, RUM

Generalized Inverted Index

- **Generalized**, потому что работает с абстрактным типом данных
- **Inverted**, потому что данные вывернуты наизнанку

GIN индексирует составные значения



INDEX
RERVM PRÆCIPVARVM
HVIVS ATLANTIS.

A.

GIN используется

В полнотекстовом поиске:

```
select ... where descr like '%linux%'
```

GIN используется

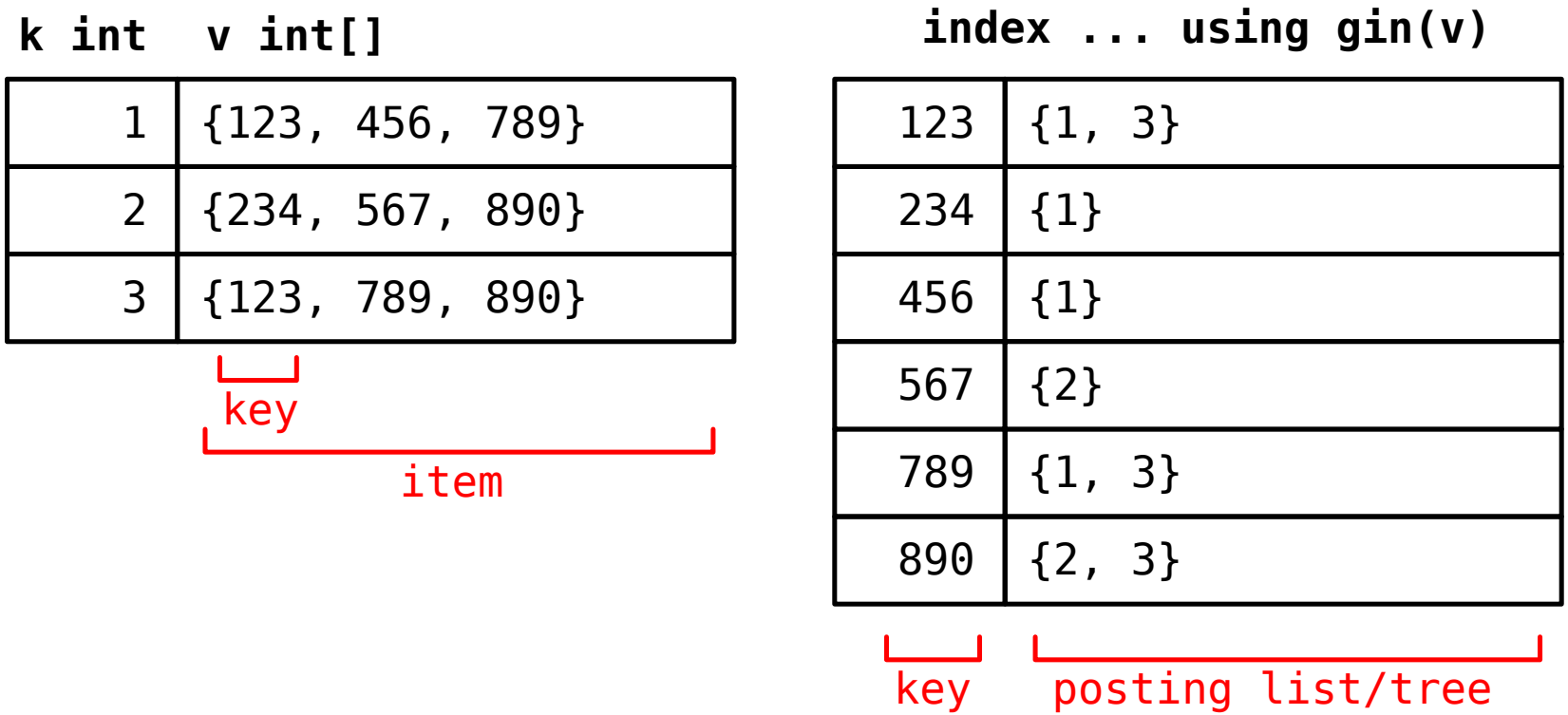
В полнотекстовом поиске:

```
select ... where descr like '%linux%'
```

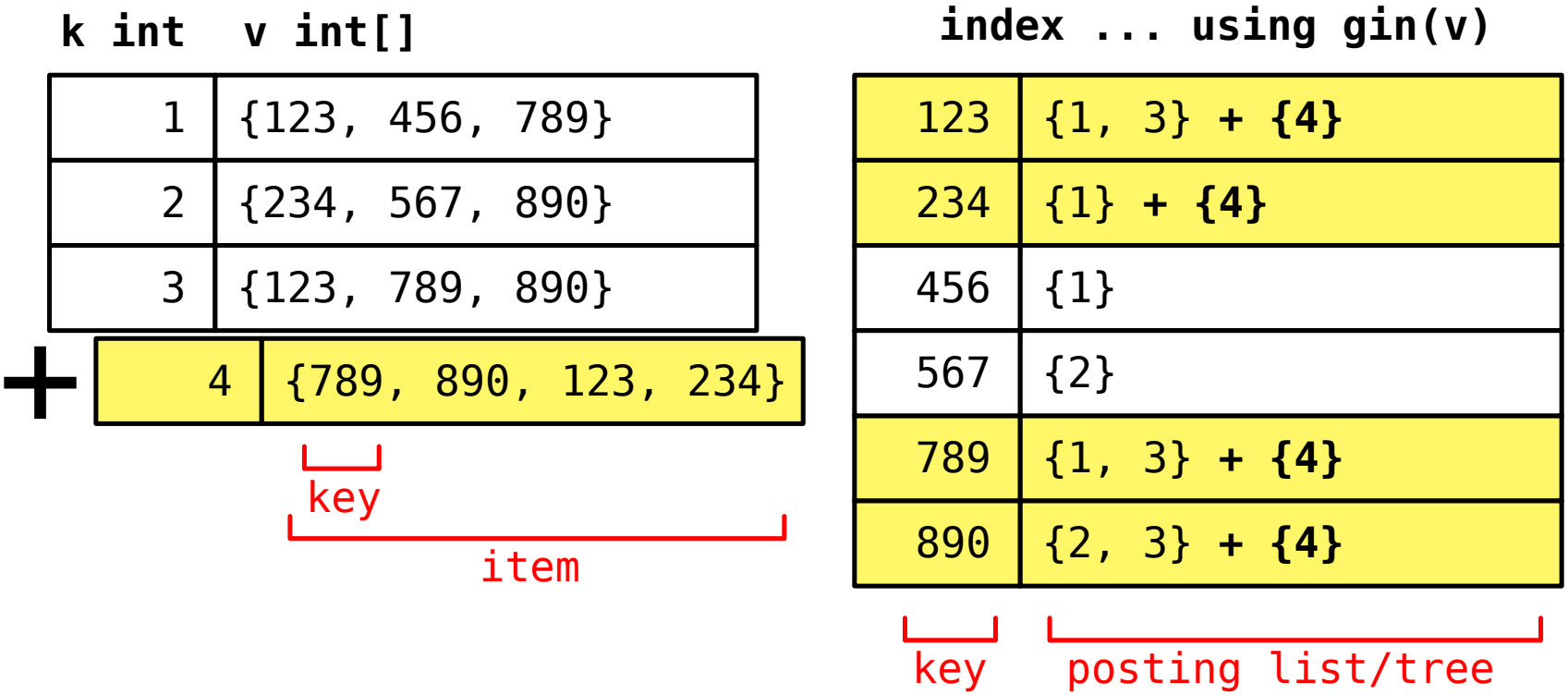
Превращается в:

```
select ... where descr_ts @@ 'linux'::tsquery
```

Структура индекса

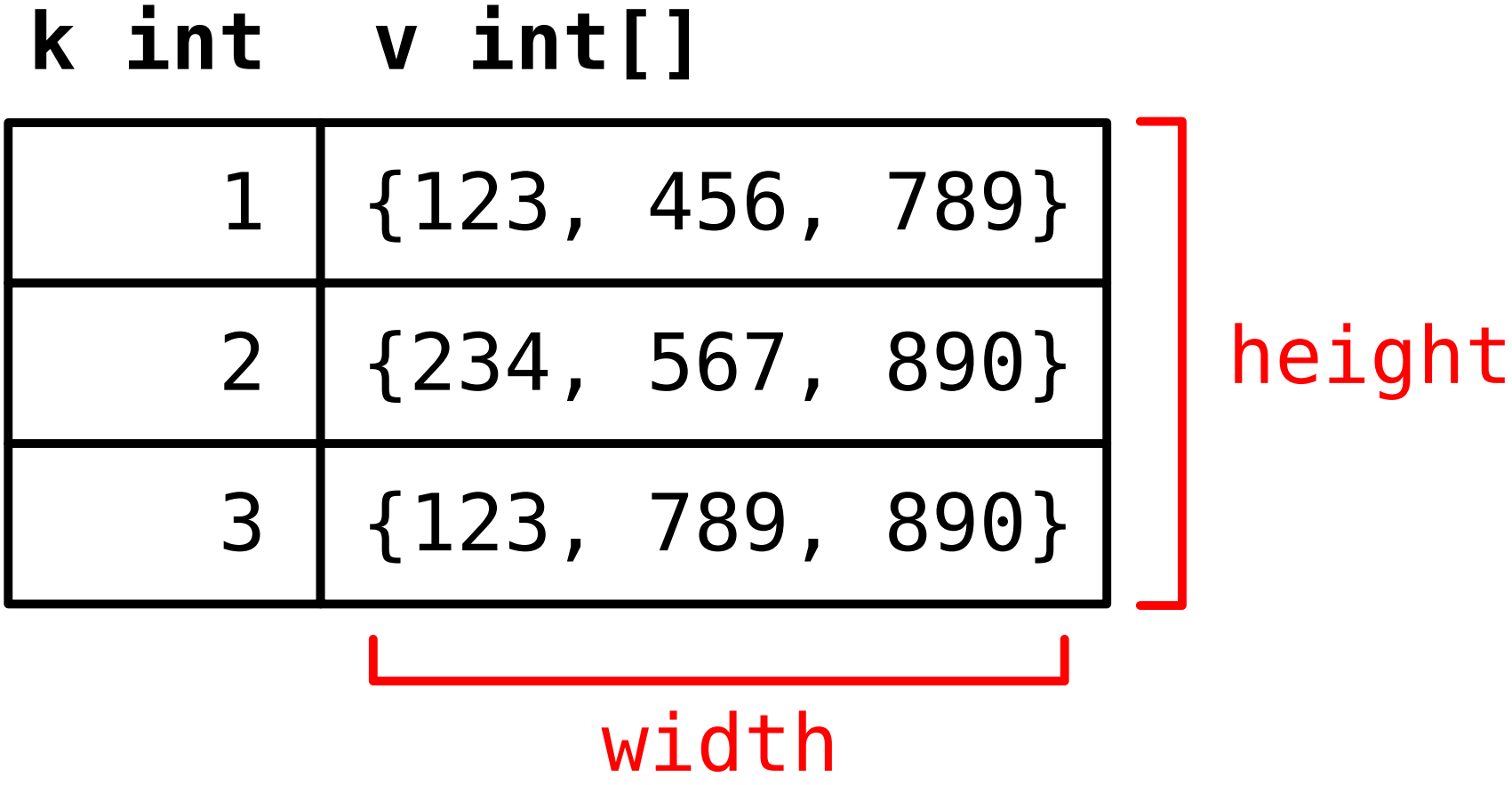


Проблема с update



Ещё есть pending list.

Проблема с create index



Проблема с create index

```
create index ... using gin(v)
```

width × height	relsize	idxsize	time
1000 × 10000	53 MiB	26 MiB	7.997 s
2000 × 10000	81 MiB	46 MiB	15.779 s
3000 × 10000	132 MiB	44 MiB	24.230 s
1000 × 20000	107 MiB	52 MiB	16.015 s
2000 × 20000	161 MiB	91 MiB	33.310 s
3000 × 20000	265 MiB	92 MiB	51.584 s
1000 × 30000	159 MiB	53 MiB	25.204 s
2000 × 30000	241 MiB	96 MiB	52.028 s
3000 × 30000	397 MiB	171 MiB	81.731 s

Проблема с `create index`

- $t \sim \text{width} \times \text{height}$
- Упирается в CPU 100%
- Долго

Много ядер

- CPU 100500%
- Быстро

Как работает `create index`

1. Построить частичный индекс

Как работает `create index`

1. Построить частичный индекс

2. Сбросить на диск

Как работает `create index`

1. Построить частичный индекс
2. Сбросить на диск
3. Повторять пока есть строки

Как работает `create index`

1. Построить частичный индекс
   - 1.1 Прочитать строку таблицы
   - 1.2 Разбить на элементы
   - 1.3 Добавить элементы в частичный индекс
   - 1.4 Повторять пока есть память
2. Сбросить на диск
3. Повторять пока есть строки

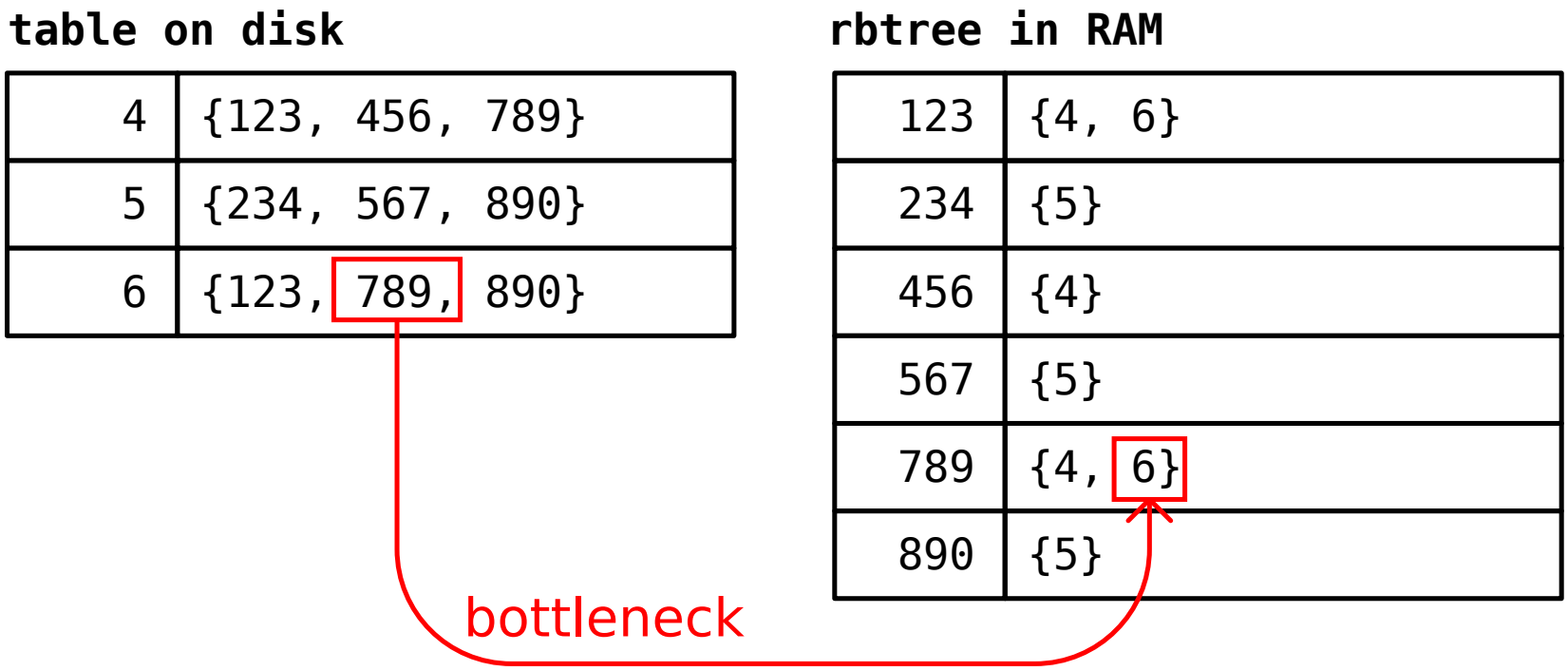
Как работает `create index`

1. Построить частичный индекс
   - 1.1 Прочитать строку таблицы
   - 1.2 Разбить на элементы
   - 1.3 Добавить элементы в частичный индекс
   - 1.4 Повторять пока есть память
2. Сбросить на диск
   - 2.1 Взять элемент из индекса
   - 2.2 Прочитать с диска существующий список
   - 2.3 Объединить с новым списком
   - 2.4 Записать на диск
   - 2.5 Повторять пока есть элементы
3. Повторять пока есть строки

Как работает `create index`

1. Построить частичный индекс
   - 1.1 Прочитать строку таблицы
   - 1.2 Разбить на элементы
   - 1.3 Добавить элементы в частичный индекс
     - Красно-чёрное дерево
     - Bottleneck
   - 1.4 Повторять пока есть память
2. Сбросить на диск
   - 2.1 Взять элемент из индекса
   - 2.2 Прочитать с диска существующий список
   - 2.3 Объединить с новым списком
   - 2.4 Записать на диск
   - 2.5 Повторять пока есть элементы
3. Повторять пока есть строки

Построить частичный индекс



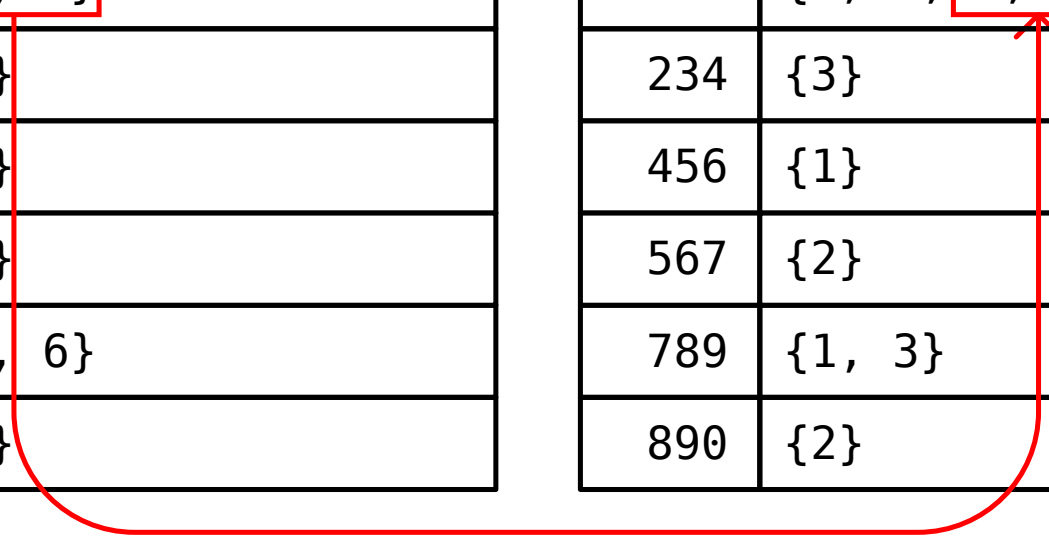
Сбросить на диск

rbtree in RAM

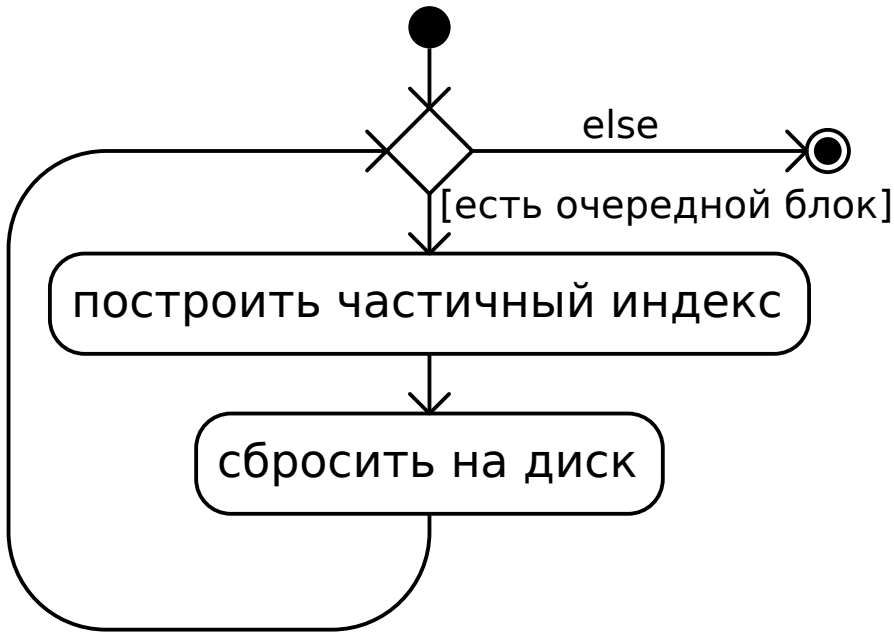
123	{4, 6}
234	{5}
456	{4}
567	{5}
789	{4, 6}
890	{5}

index on disk

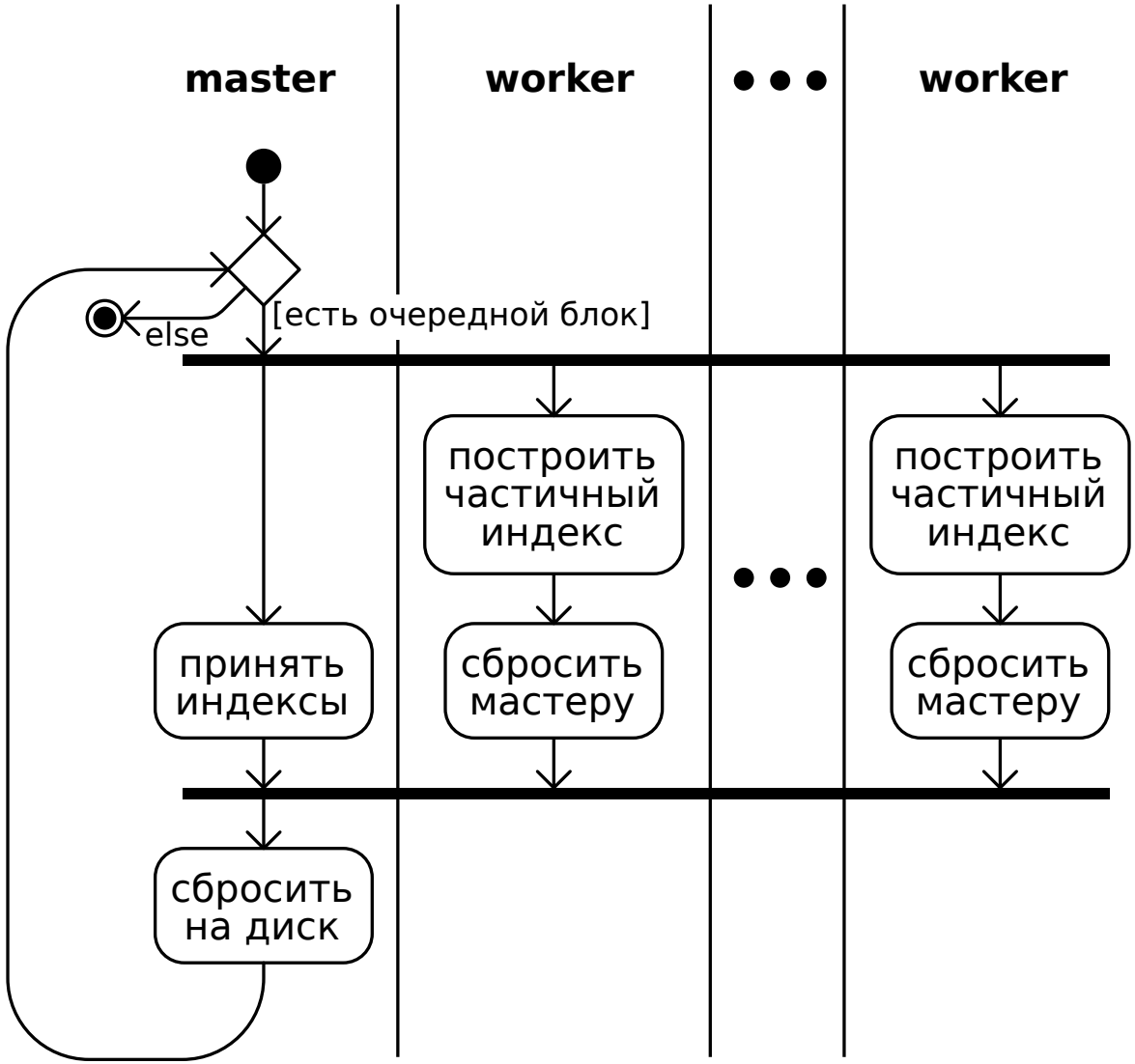
123	{1, 2, 4, 6}
234	{3}
456	{1}
567	{2}
789	{1, 3}
890	{2}



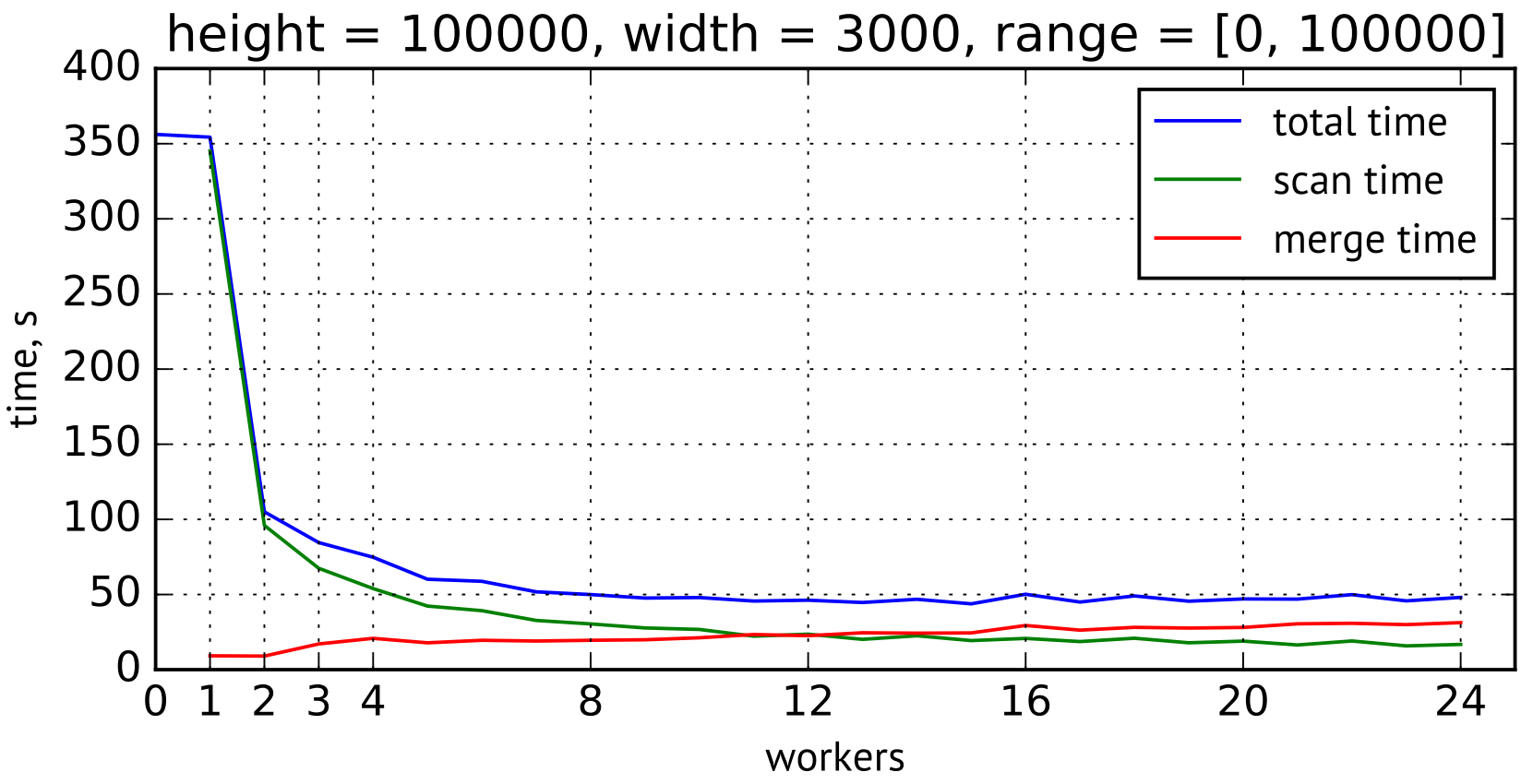
Построение индекса



Параллельное построение индекса



Ускорение


height = 100000, width = 3000, range = [0, 100000]
time, s
workers
total time
scan time
merge time
0
50
100
150
200
250
300
350
400
0
1
2
3
4
8
12
16
20
24

Закон Амдала

$$S_p \leq \frac{1}{\alpha + \frac{1-\alpha}{p}}$$

Если последовательный код выполнялся 5% времени, то нельзя ускорить выполнение больше чем в 20 раз. Даже с бесконечным количеством процессоров.

Спасибо

Константин Пан <kvapen@gmail.com>

Ссылка на обсуждение в сообществе:
https://www.postgresql.org/message-id/flat/20160116013839.57cfcb37@thought