Пример использования GiST в решении нестандартной поисковой задачи

На примере поиска «похожих» текстов

PGConf.Russia 2020

Андрей Зубков
zubkov@moonset.ru

PARMA
technologies group

О чем это всё?

- PostgreSQL расширяемый
- GiST — не столько индекс, сколько каркас для новых стратегий поиска

Посмотрим как это работает на реальной задаче

Постановка задачи

«Сижу я, работаю, проверяю донесения министров друг на дружку.»

Евгений Шварц. «Обыкновенное чудо»

- Есть большой массив текстов на естественном языке
- Массив пополняется новыми текстами
- Новый текст часто бывает «повтором», но с незначительными изменениями, вроде:
  - Перестановка абзацев
  - Ошибки
  - Дополнительные слова или фразы, изъятие некоторых слов/фраз
- Нам надо быстро определять такие случаи при поступлении нового текста

Первые соображения

- Полнотекстовый поиск?

Полнотекстовый поиск?

В полнотекстовом поиске	В нашей задаче
Запрос краткий, но точный	Запрос представлен полным текстом искомого материала с искажениями
Обычно в запросе нет ошибок	Ошибки в запросе наверняка есть
Осуществляется поиск вхождения конкретных лексем в искомом тексте	Осуществляется поиск текстов целиком

Первые соображения

- Полнотекстовый поиск? - скорее всего, нет
- pg_trgm + индексы?

pg_trgm + индексы?

Разбивает текст на триграммы и позволяет искать близкие по наборам триграмм. Поддерживается индексами.

Ура, сработало, но:

- Большой индекс
- Производительность поиска приемлема до 1 млн.
- На 4 млн. один поиск занимал примерно 10 мин.

Что не так?

- индексирование триграмм и их вхождений

Первые соображения

- Полнотекстовый поиск? - скорее всего, нет
- pg_trgm + индексы? - хорошо, но не для этого

Однако, стало понятно что *n*-граммный подход работает.

Развитие идеи

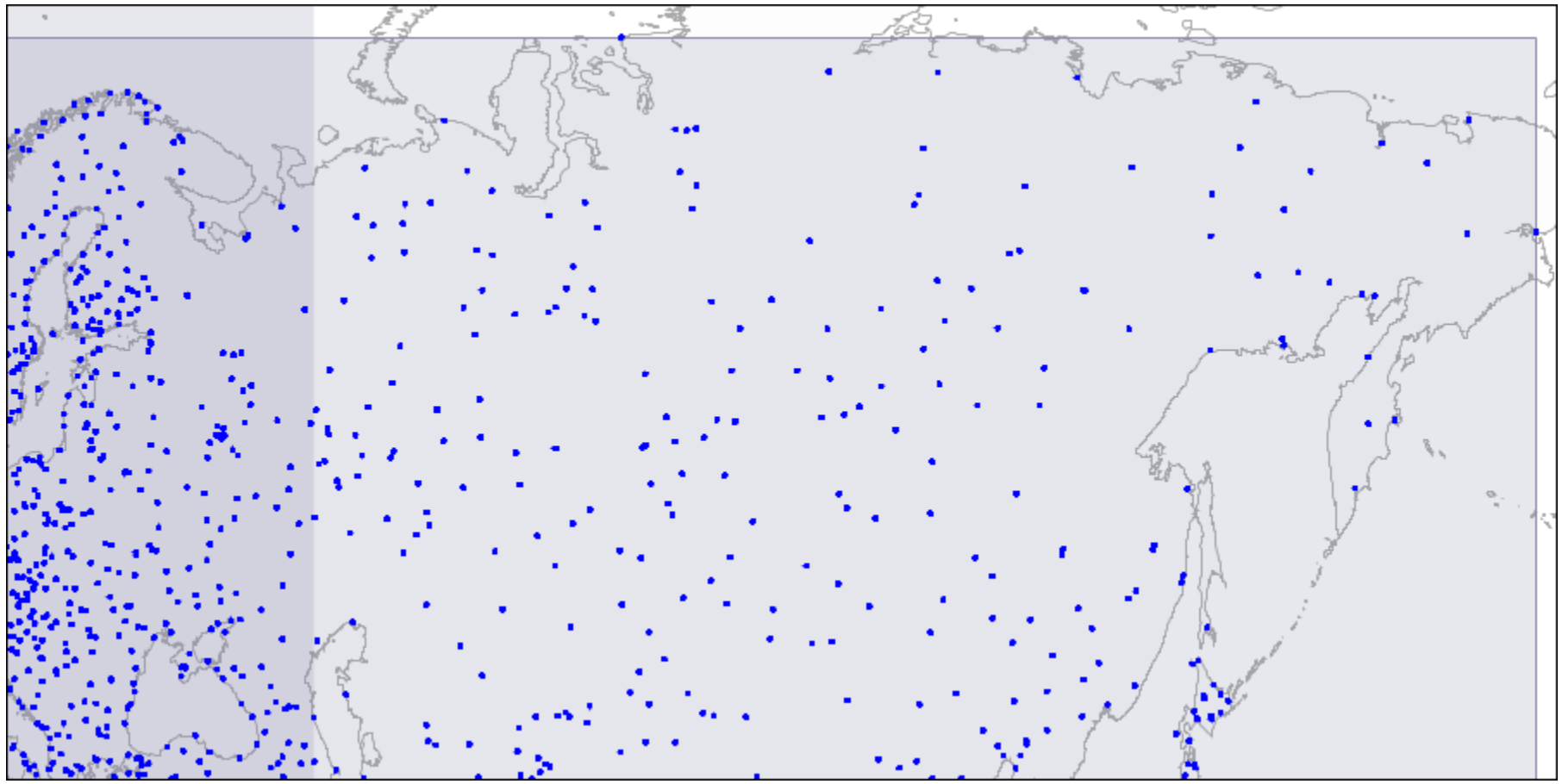
- Единица поиска (и индексирования) — целый текст
- Поиск «похожих» текстов наводит на мысль о наличии меры близости
- Посмотрим на GiST

GiST – generalized search tree

- Сбалансированная по высоте древовидная структура
- Каждый элемент листового узла содержит предикат, которому соответствует индексированное значение
- Каждый узел дерева также содержит предикат, которому удовлетворяют значения всех дочерних узлов/листьев
- Для поиска используется *функция согласованности*, которая определяет согласуется ли предикат узла индекса с условием поиска.

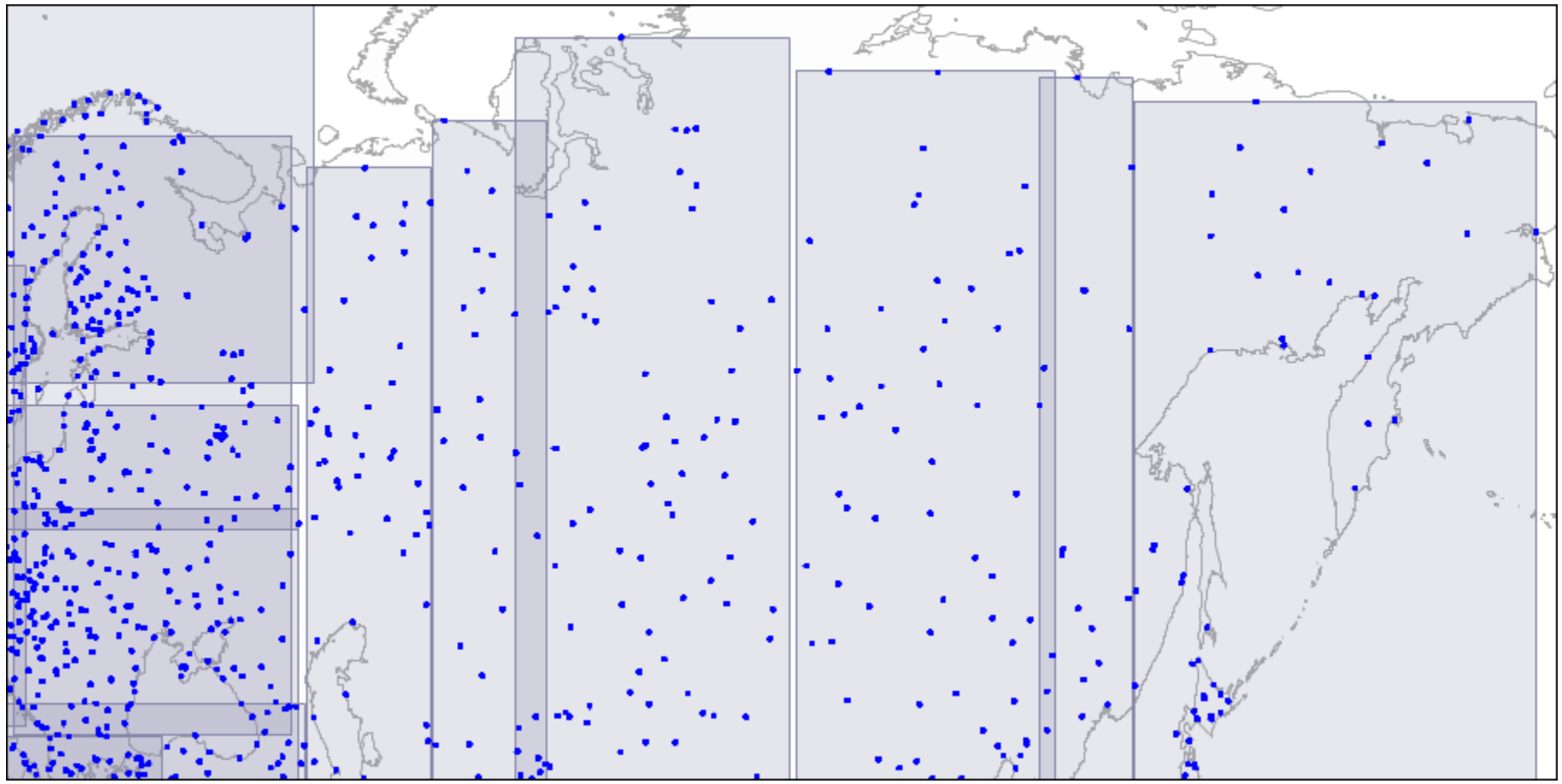
https://habr.com/ru/company/postgrespro/blog/333878/

GiST – generalized search tree



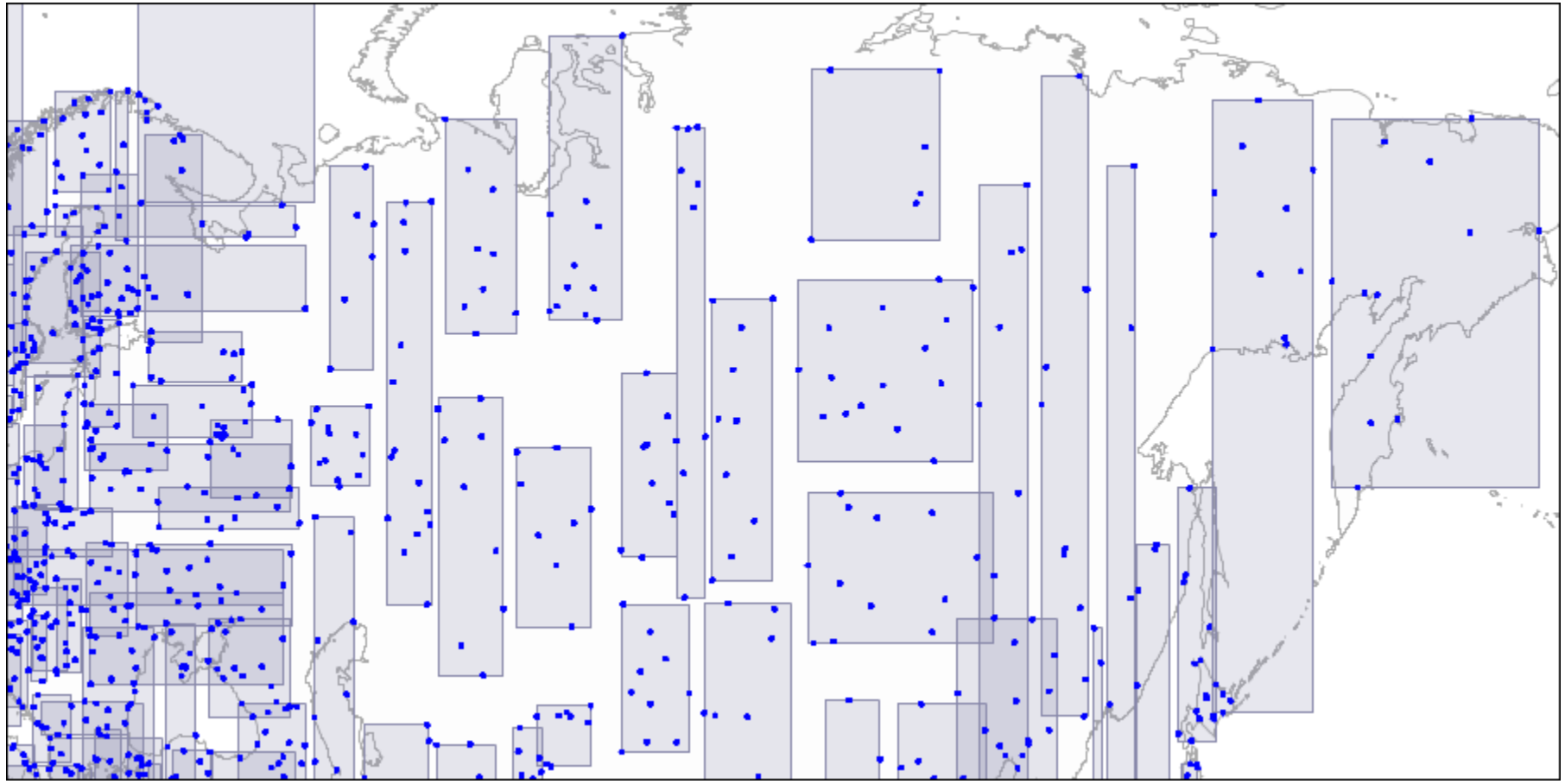
https://habr.com/ru/company/postgrespro/blog/333878/

GiST – generalized search tree



https://habr.com/ru/company/postgrespro/blog/333878/

GiST – generalized search tree



https://habr.com/ru/company/postgrespro/blog/333878/

Развитие идеи

- Единица поиска (и индексирования) — целый текст
- Поиск «похожих» текстов наводит на мысль о наличии меры близости
- Посмотрим на GiST

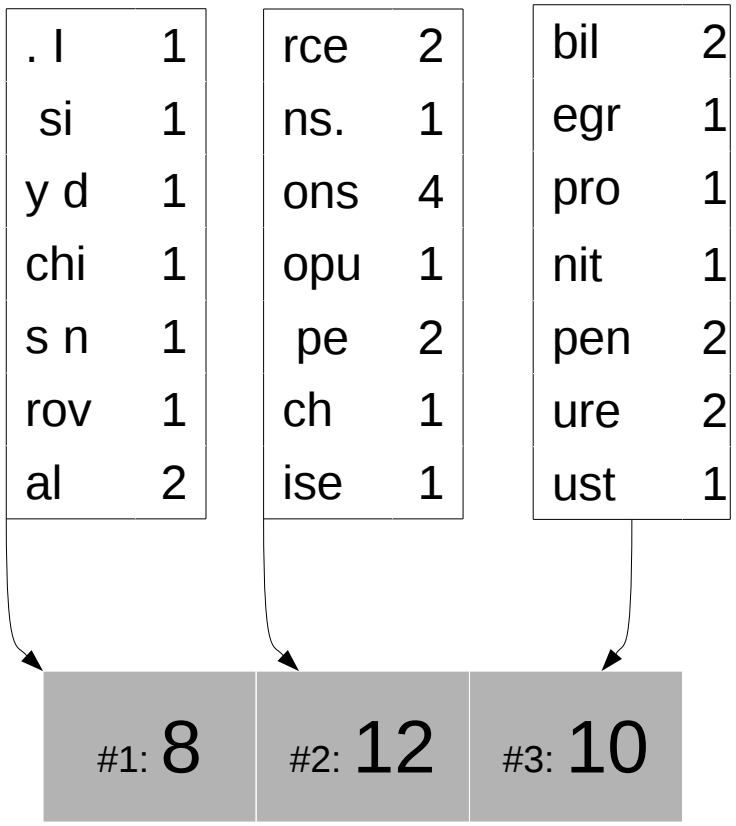
Надо свести задачу к пространственному поиску.

Осталось как-то разложить тексты в пространстве.

Пространственный поиск текстов

Посчитаем частоты триграмм в тексте и снизим размерность, объединив триграммы хэшами:

«PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organisations.»

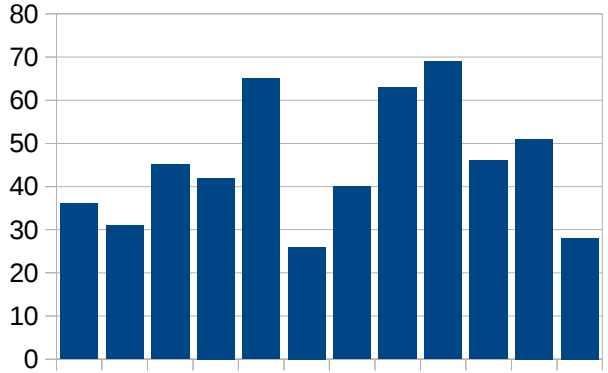


Пространственный поиск текстов

80 строк кода на Си, и у нас есть профили текстов:

```
search_test=# select get_vector(12,:'TEXT');

                get_vector
-----------------------------------
 {36,31,45,42,65,26,40,63,69,46,51,28}
```



Расширение *cube* добавит поддержку многомерных пространств:

```
search_test=# CREATE INDEX ix_textvector ON texsts_table USING gist
(cube(get_vector(12, content)));

CREATE INDEX

Time: 139408.755 ms (02:19.409)
```

Пространственный поиск текстов

Ошибку первого рода устраним перепроверкой:

```
CREATE OR REPLACE FUNCTION search(IN msg text, IN lim integer DEFAULT 60)
RETURNS TABLE (content text, distance float)
IMMUTABLE AS
$$
SELECT content, distance
FROM
    (SELECT content, cube(get_vector(12, content))<->cube(get_vector(12,
msg)) distance
    FROM texsts_table
    ORDER BY cube(get_vector(12, content))<->cube(get_vector(12, msg))
    LIMIT lim) tb
    WHERE similarity(msg, content) >= 0.6;
$$ LANGUAGE SQL;
```

Пространственный поиск текстов

Наконец, выполняем поиск:

```
search_test=# select search('<длинный текст>');
Time: 110.150 ms
```

Что получилось

- Относительно компактный индекс
- Отсутствие зависимости от размеров текстов и их содержания
- Быстрое построение индекса и быстрый поиск

Благодарности

- Егору Рогову за его труд вообще, и из статьи по индексам в частности
- Александре Кузнецовой за помощь с функцией построения профилей

Спасибо за внимание!

Вопросы…

Андрей Зубков

Администратор баз данных

zubkov@moonset.ru