Эффективная работа с пространственными данными в PostgreSQL

Февраль, 2015, Москва, Россия

Александр Коротков, Олег Бартунов, Фёдор Сигаев

Postgres Professional

Олег Бартунов::Teodor Sigaev

- Locale support
- Extendability (indexing)
  - GiST, GIN, SP-GiST
- Extensions:
  - intarray
  - pg_trgm
  - ltree
  - hstore, hstore v2.0 → jsonb
  - plantuner
  - jsquery
- Full Text Search (FTS)

https://www.facebook.com/oleg.bartunov
obartunov@gmail.com

Alexander Korotkov

- Indexed regexp search
- GIN compression & fast scan
- Fast GiST build
- Range types indexing
- Split for GiST
- Indexing for jsonb
- jsquery
- Generic WAL + create am (WIP)

aekorotkov@gmail.com

Где бывают нужны пространственные данные (ПД)?

- ГИС
- Астрономия
- САПР
- Рекомендационные системы
- …

Почему нужно хранить ПД в СУБД?

- Доступность
- Транзакционность
- Простота обслуживания

Почему нужна специальная поддержка ПД в СУБД?

- Нужны специальные типы данных
- Нужны специальные операции (функции, операторы)
- Нужны специальные индексы

Способы работы с ПД в PostgreSQL

- Встроенные типы данных
- PostGIS
- pg_sphere
- cube
- Q3C

Встроенные типы

- Отдельные типы для разных геометрических объектов: point, box, polygon, circle и т.д.
- Геометрия на плоскости
- GiST индексы, поддержка KNN для точек и прямоугольников

http://www.postgresql.org/docs/9.4/static/datatype-geometric.html

PostGIS

- Типы-контейнеры
  - geometry – геометрия в 2D-4D евклидовом пространстве
  - geography – геометрия на сфере
- Соответствие спецификаци Simple Features для SQL от OGS
- GiST индексы, поддержка KNN
  - Оператор "<->" – расстояние до центра MBR
  - Оператор "<#>" – расстояние до края MBR

http://postgis.net/

pgSphere

- Геометрия на сфере
- Отдельные типы для разных геометрических объектов: spoint, sbox, sline, spath, spolygon и т. д.
- GiST индексы
- Форк: https://github.com/akorotkov/pgsphere
  - Bug fixes
  - KNN
  - Experimental spatial join

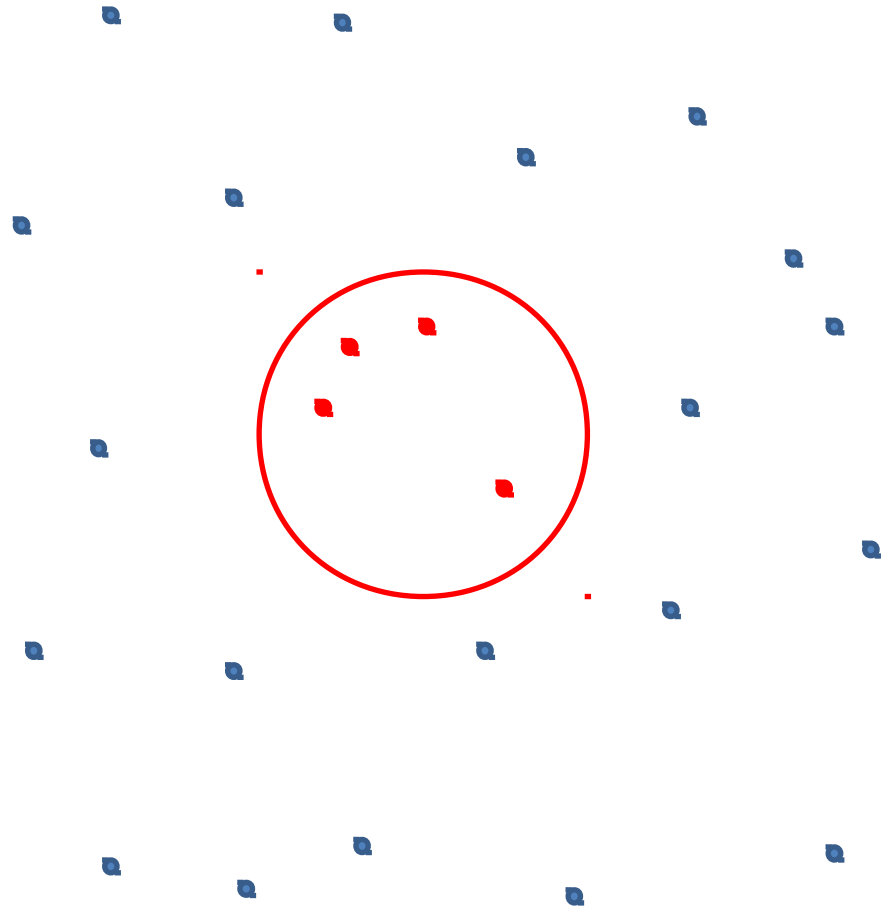
Q3C

- Не вводятся новые типы данных, работа только с точками на сфере
- Функциональный btree индекс
- Пространственный поиск по btree, spatial join

https://code.google.com/p/q3c/

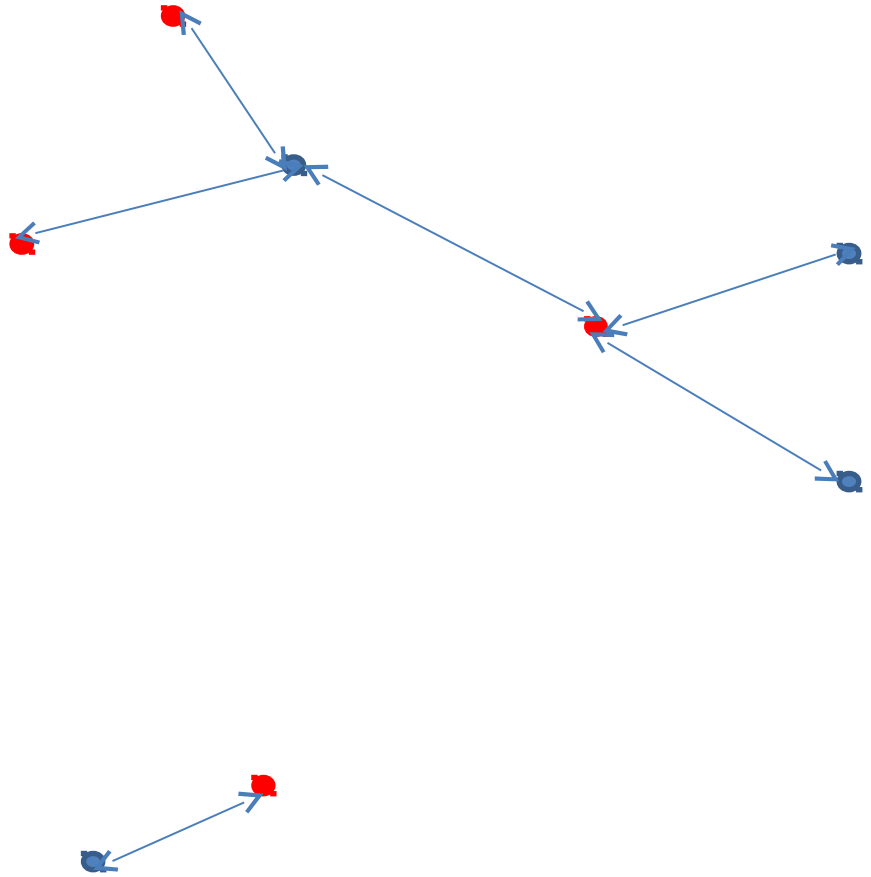
Радиальный запрос

Найти все точки, расположенные не дальше чем X от заданной точки (расположенные внутри заданной окружности).



Кросс-отождествление (spatial join)

Из двух множеств точек найти все пары, расположенные не дальше чем X друг от друга.

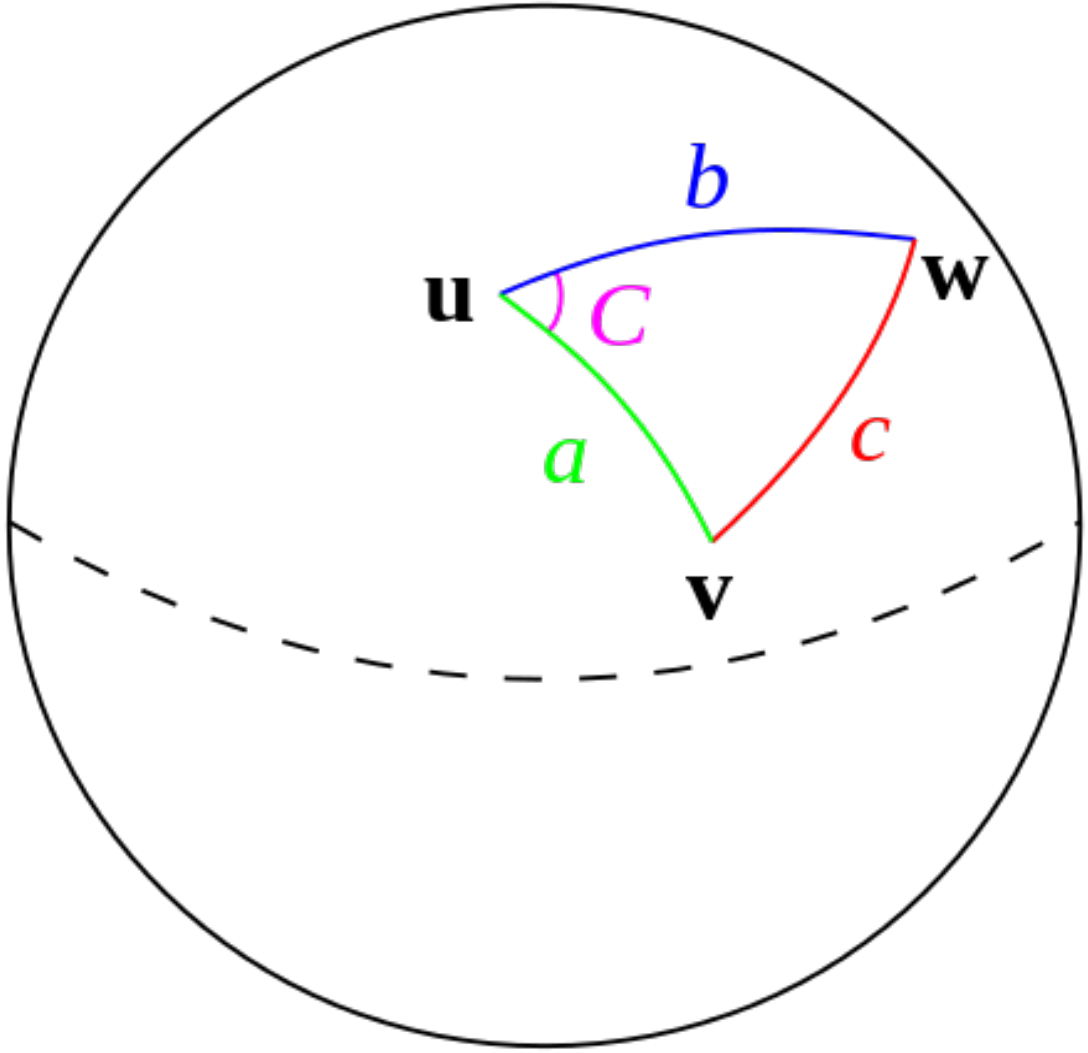


Сравнение методов поиска на сфере

- haversine
- Q3C
- pgSphere
- PostGIS

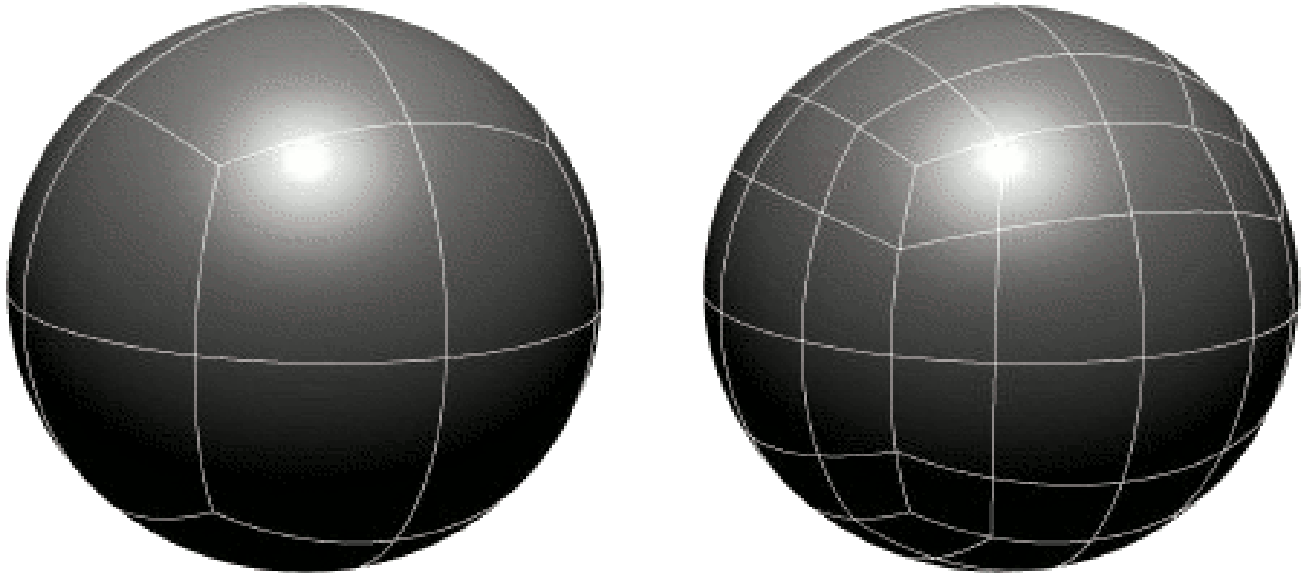
PROFESSIONAL Postgres

Haversine



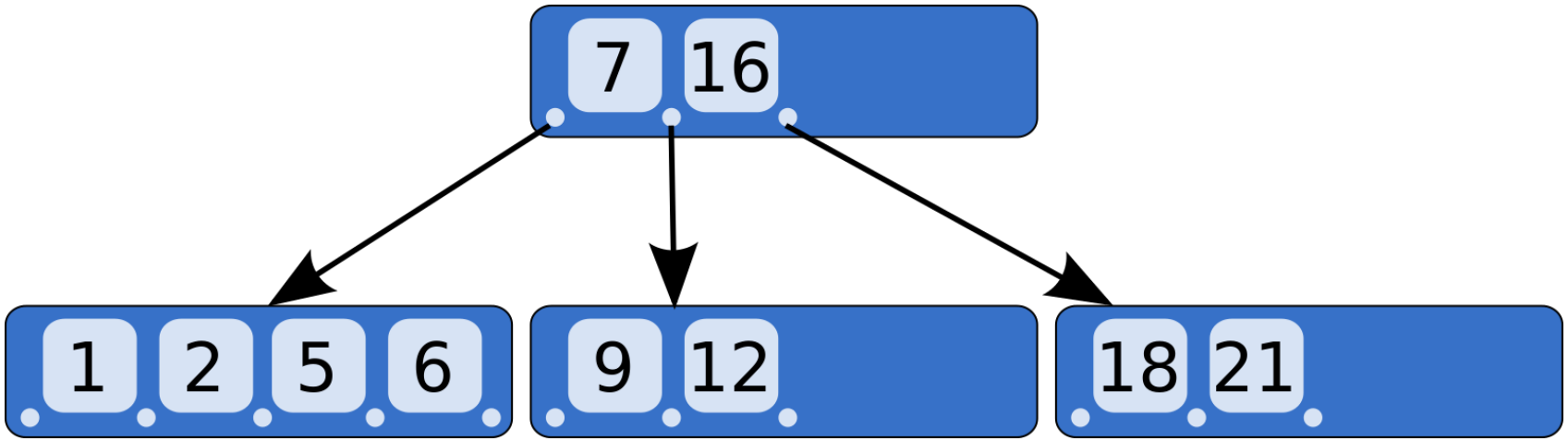
Структура q3c

- Куб вписан в сферу
- Центральная проекция куба на сферу
- Quad-tree на гранях куба связывает координаты на сфере с целым числом (IPIX)



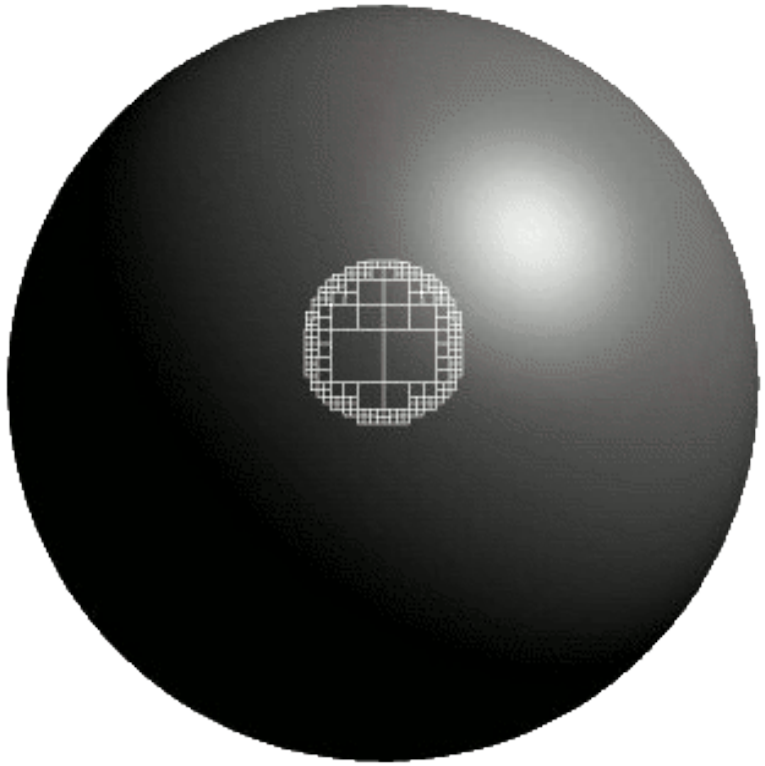
Индекс q3c

По номерам квадрантов в q3c строится обычный btree индекс



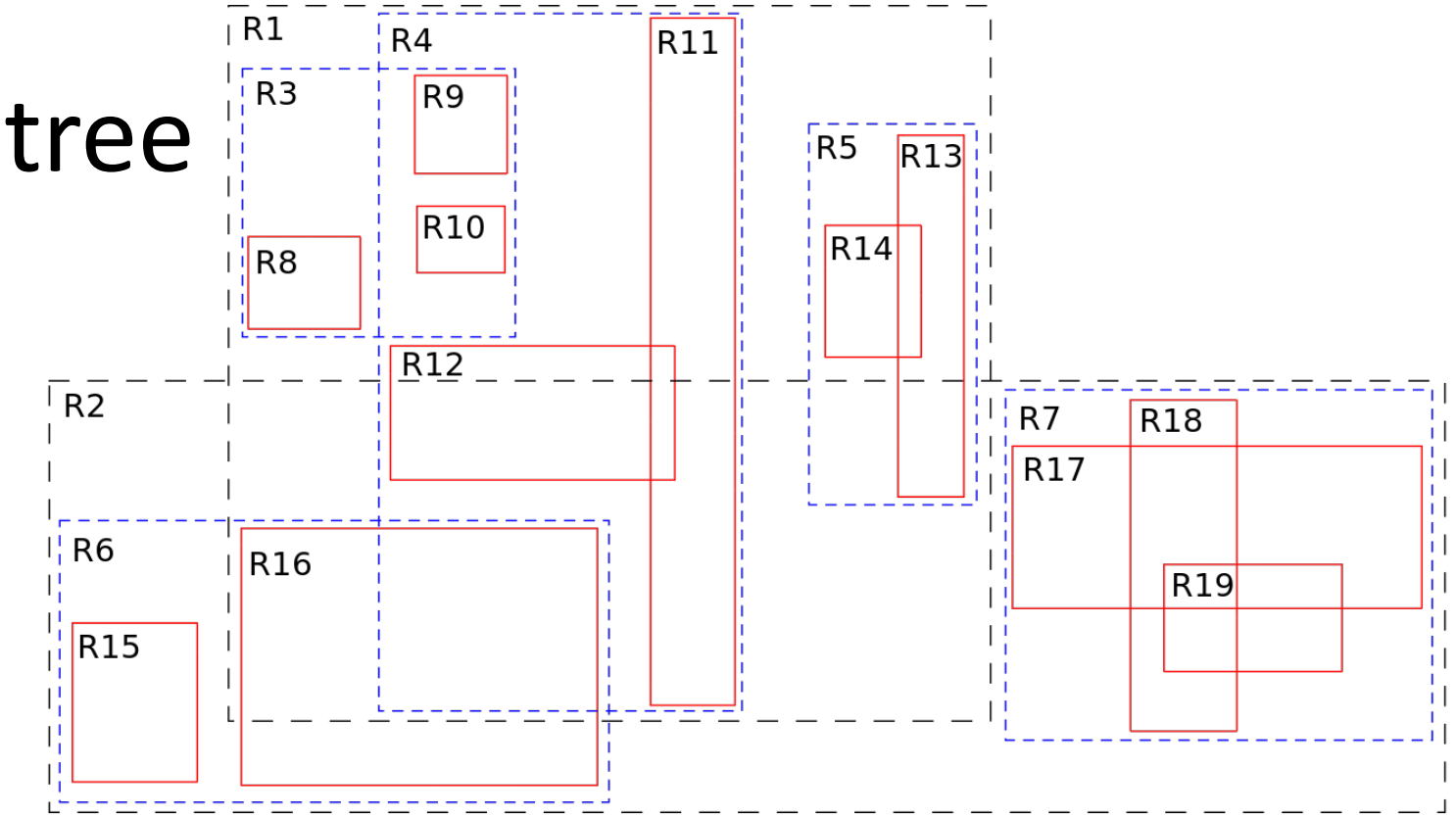
Обработка запросов q3c

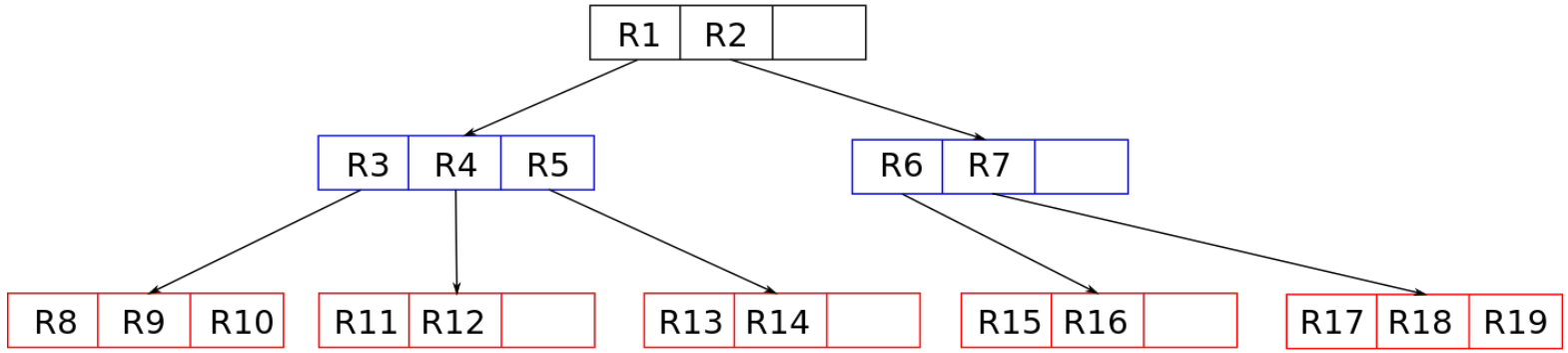
Пространственный запрос преобразуется в объединение диапазонных запросов к номеру квадранта.



PROFESSIONAL Postgres

R-tree





PROFESSIONAL Postgres

R-tree: пример

Generalized search tree (GiST)

- Обобщение над R-tree и его вариациями
- Единственная полноценная реализация – в PostgreSQL
- Множество реализация (operator class) для решения различных задач

Версии pgSphere

- pgSphere 1.1.1 – последнее версия на начало исследования
- pgSphere 1.1.2 – исправлены ошибки, изменен алгоритм разделения узла
- pgSphere 1.1.5 – добавлен opclass spoint2, позволяющий хранить точки в листовых узлах (не нужен recheck)

cube contrib

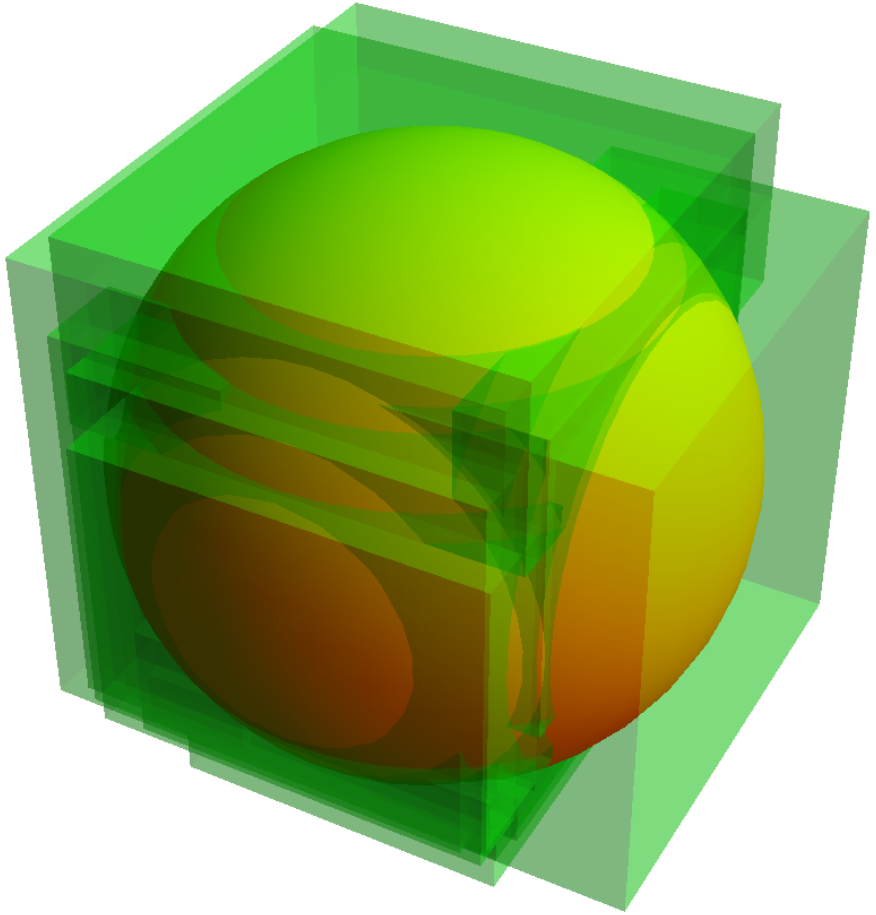
- Тип cube – «куб» (на самом деле n-мерный прямоугольник)
- GiST индексы
- Стас Кельвич GSoC 2013 – компактное хранение точек(committed), KNN, типы, улучшение GiST.

http://www.postgresql.org/docs/9.4/static/cube.html

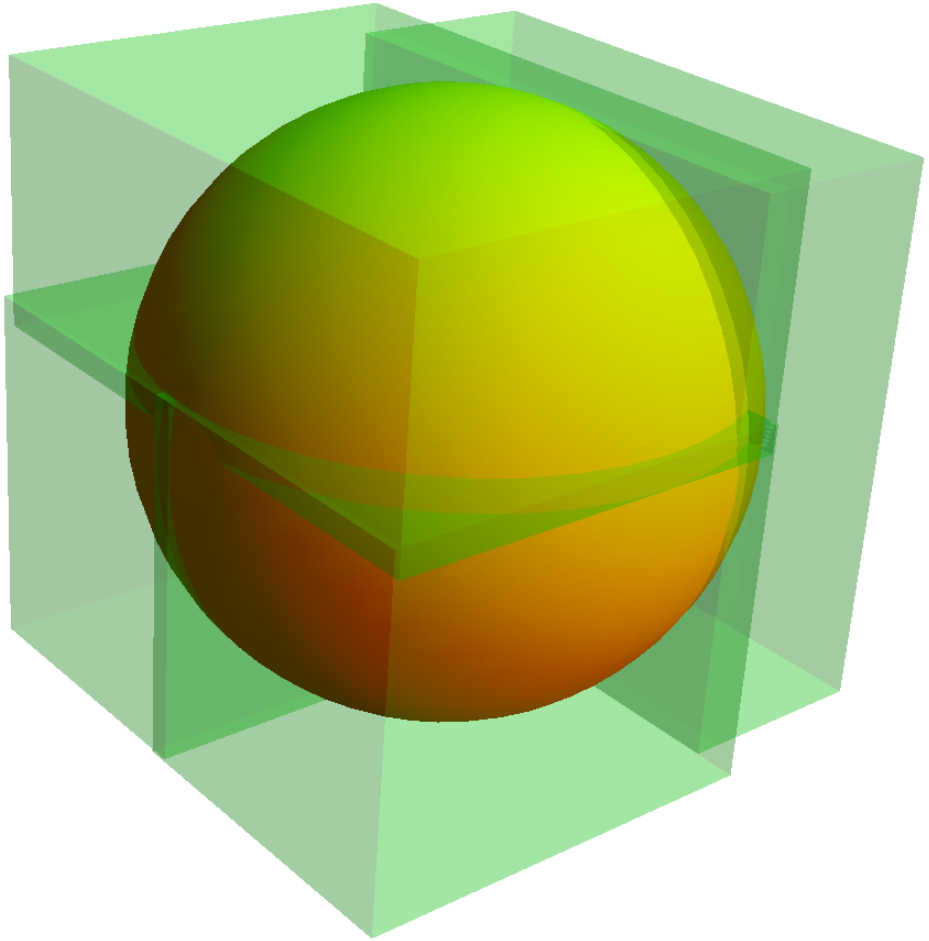
PROFESSIONAL Postgres

pgSphere

Старый split

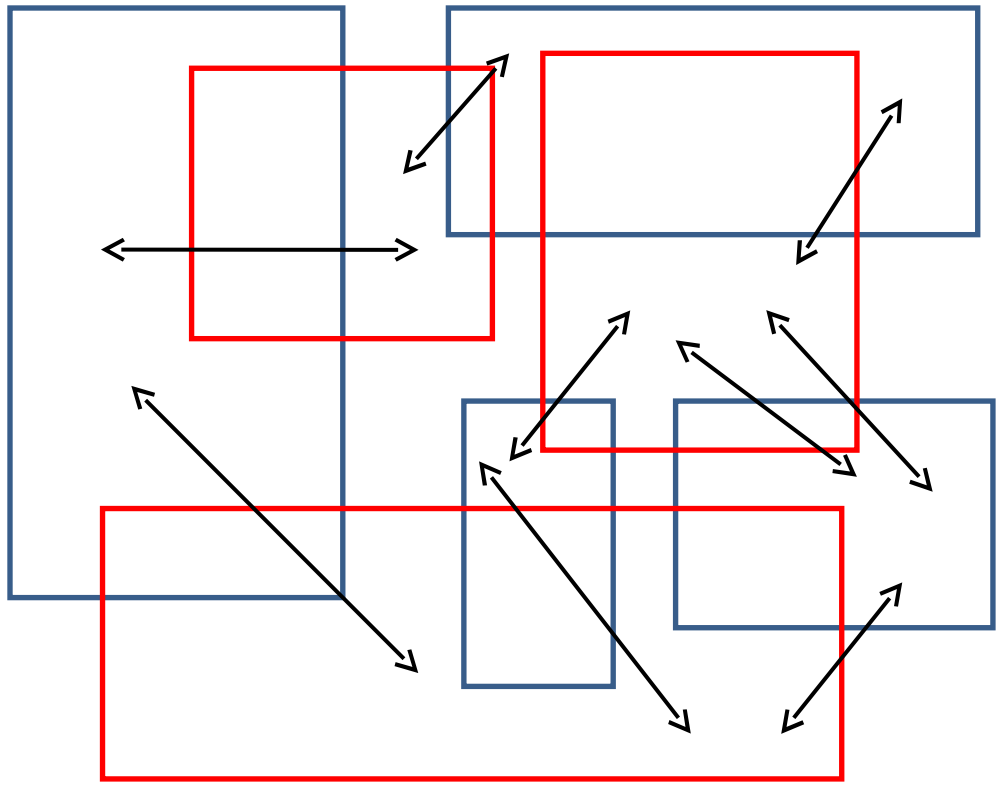


Новый split



Кросс-отождествление по двум индексам

- Выбор пар пересекающихся MBR в корневой странице R-tree
- Рекурсивное повторение операции для страниц, соответствующих выбранным парам MBR

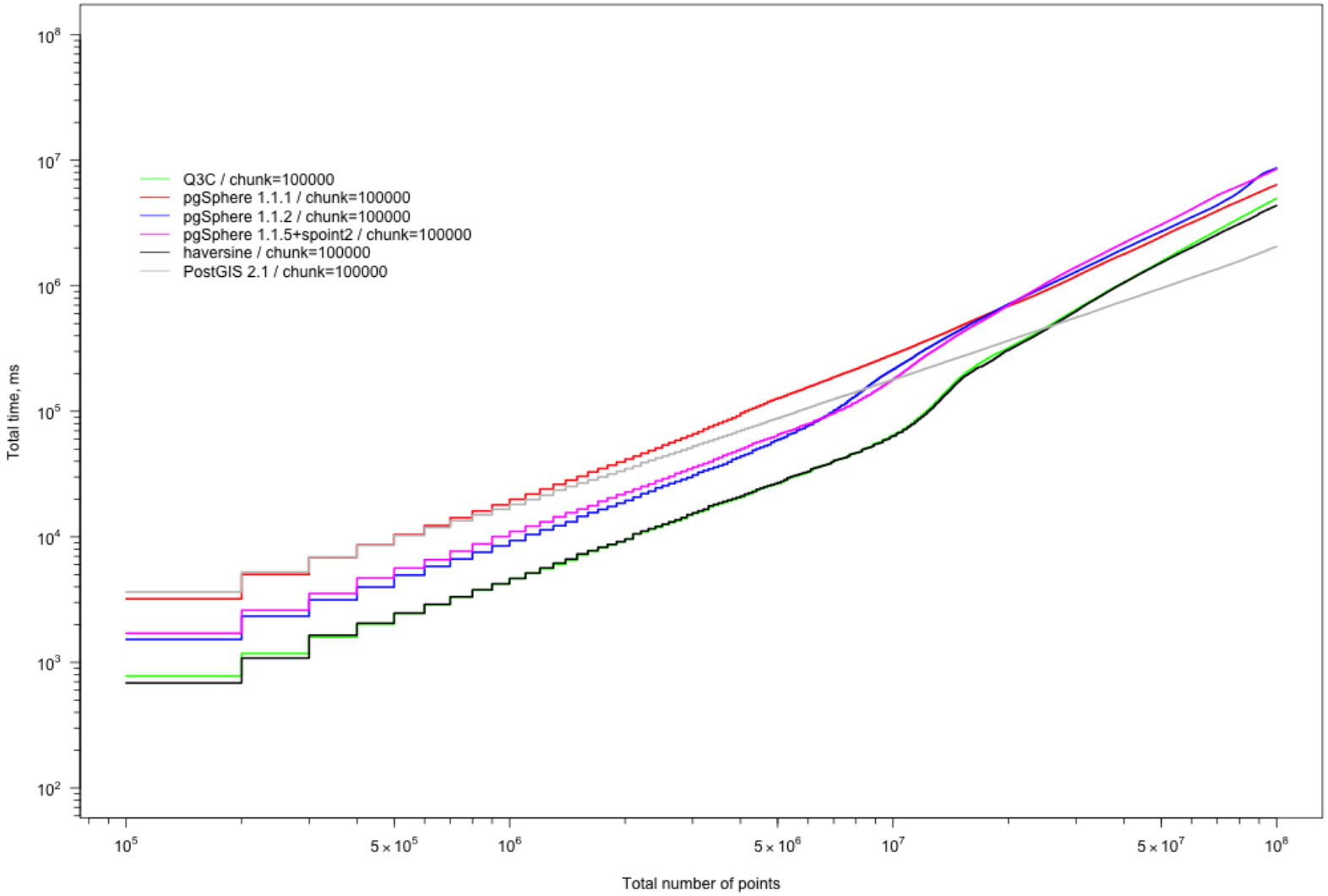


PostGIS

- Расширение для работы с ГИС данными, реализующее стандарт OpenGIS
- Очень популярно, используется в таких ГИС платформах, как ArcGIS и QGIS
- Распространяется по лицензии GNU GPL 2
- Содержит в себе реализацию n–мерного R–дерева на основе GiST.

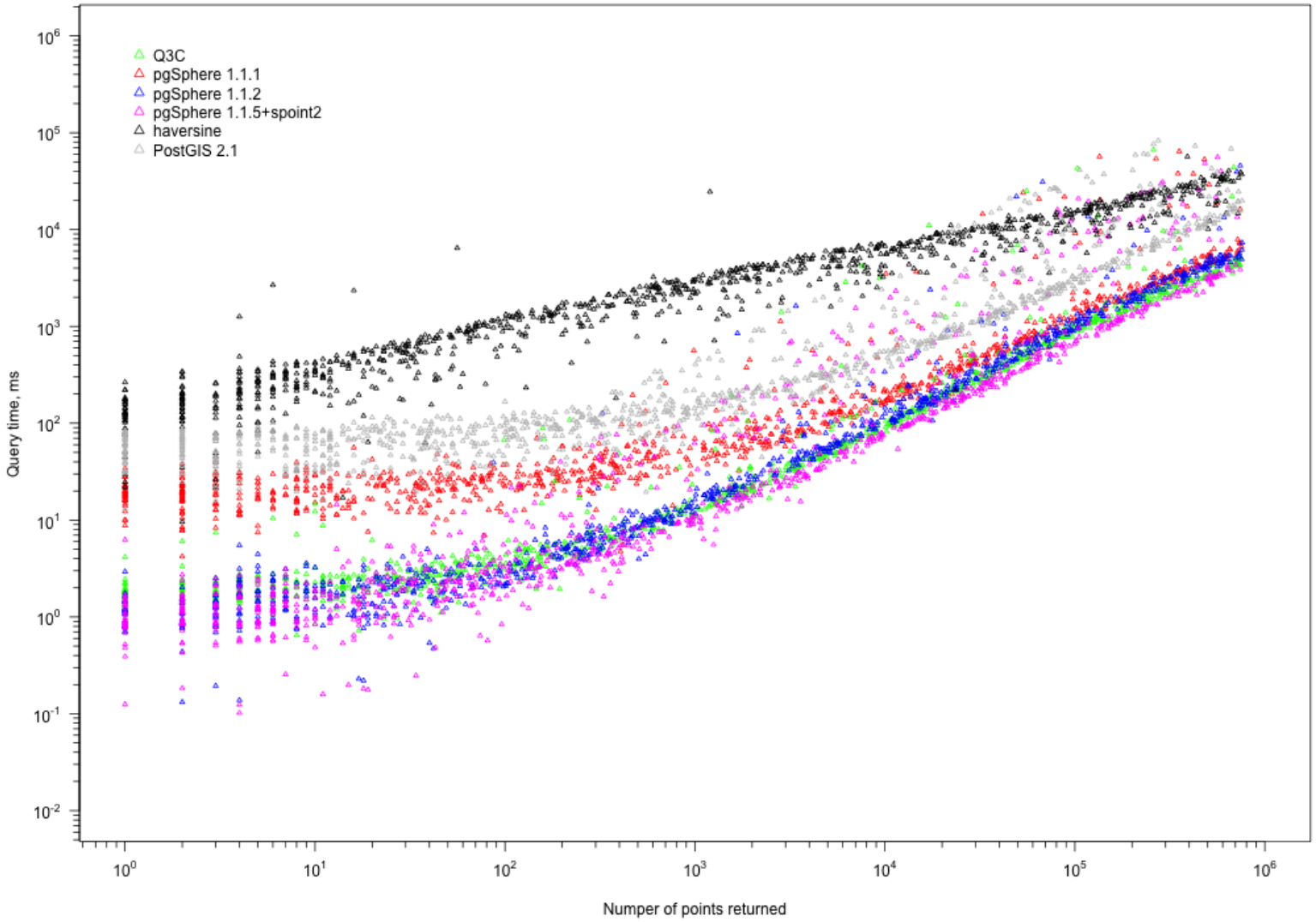
Время, затрачиваемое на вставку

- Q3C
- pgSphere
  - 1.1.1
  - 1.1.2
  - 1.1.5 spoint2
- haversine
- PostGIS 2.1



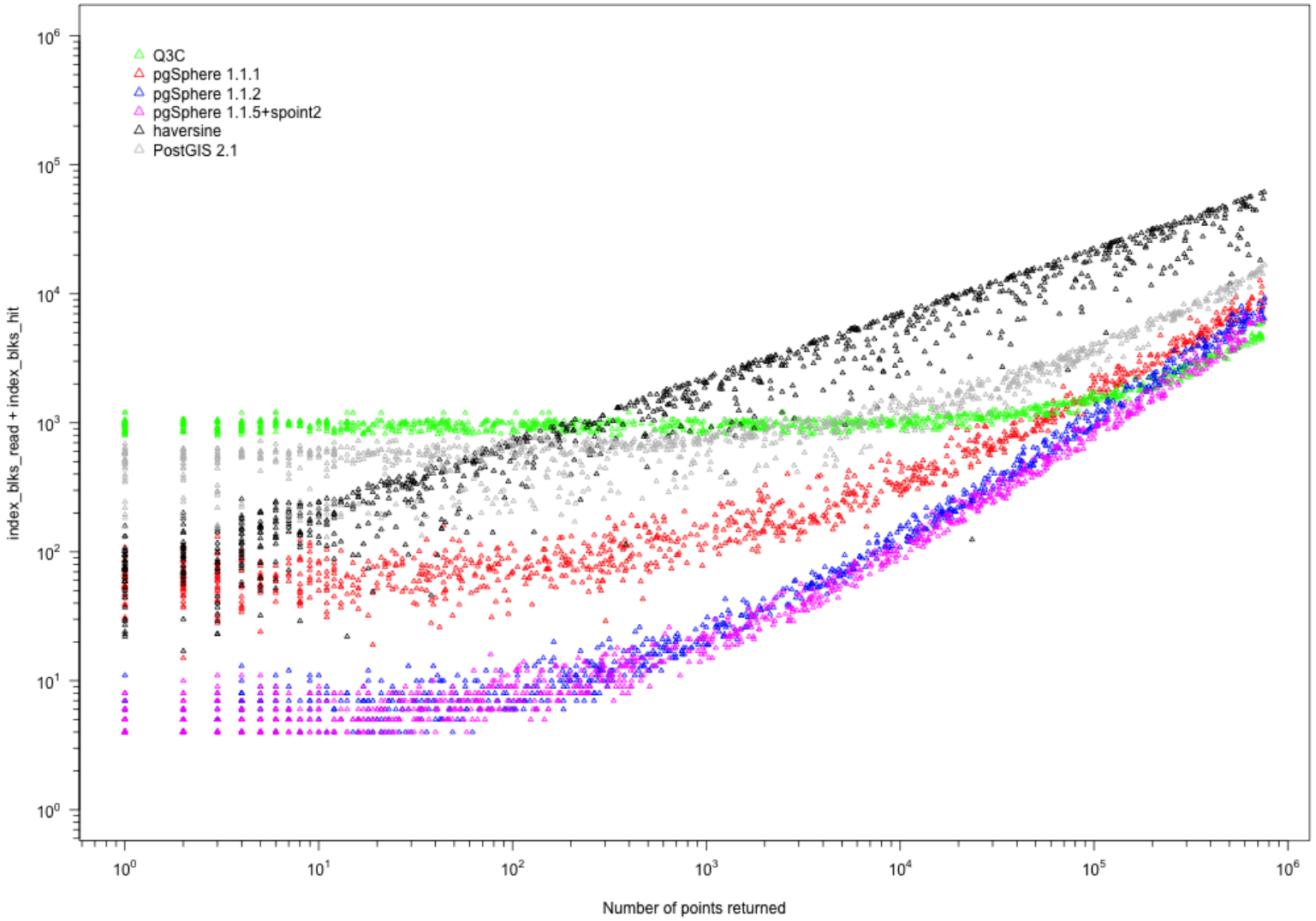
Время выполнения радиальных запросов

- Q3C
- pgSphere
  - 1.1.1
  - 1.1.2
  - 1.1.5 spoint2
- haversine
- PostGIS 2.1



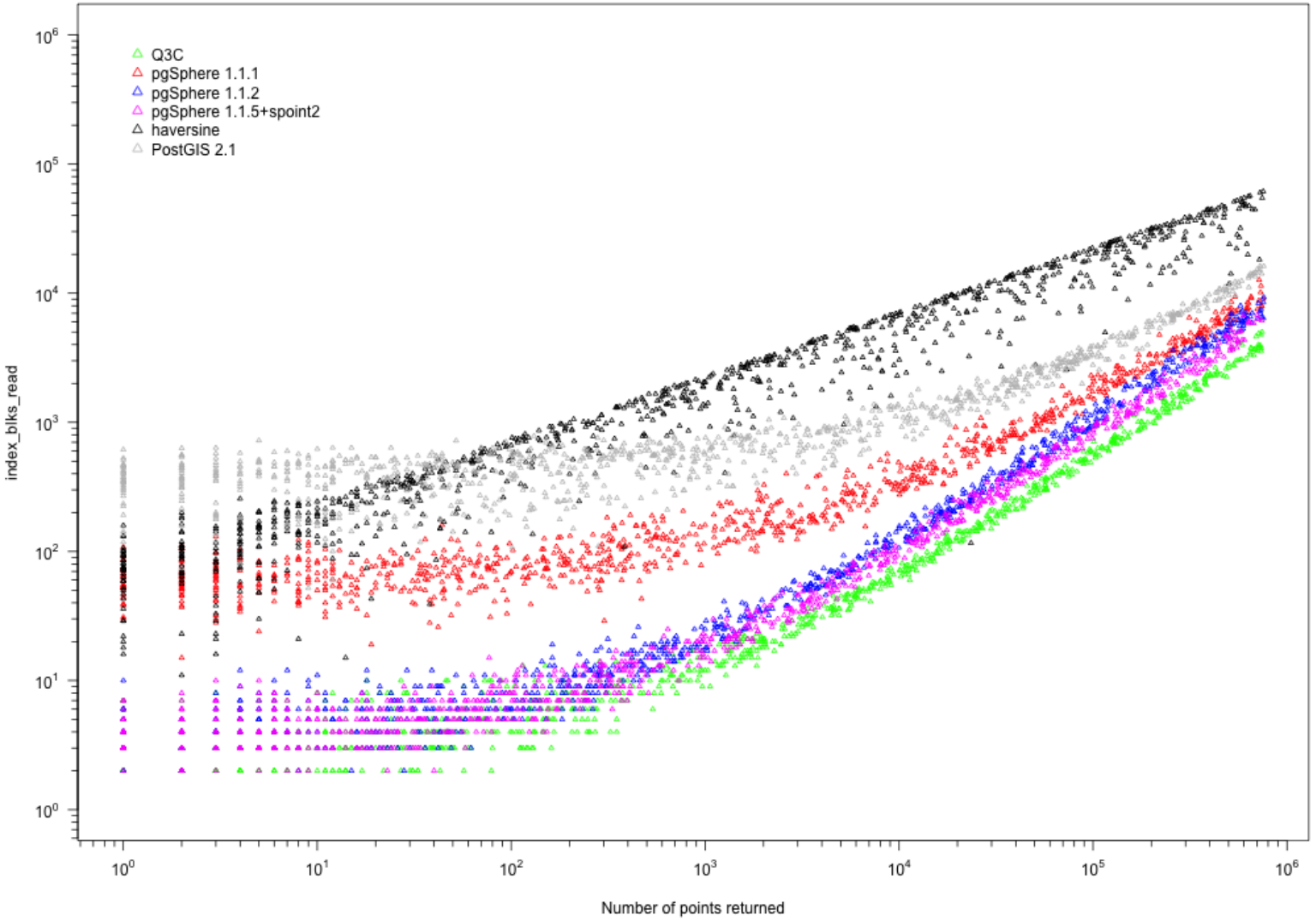
Число использованных блоков индекса

- Q3C
- pgSphere
  - 1.1.1
  - 1.1.2
  - 1.1.5 spoint2
- haversine
- PostGIS 2.1



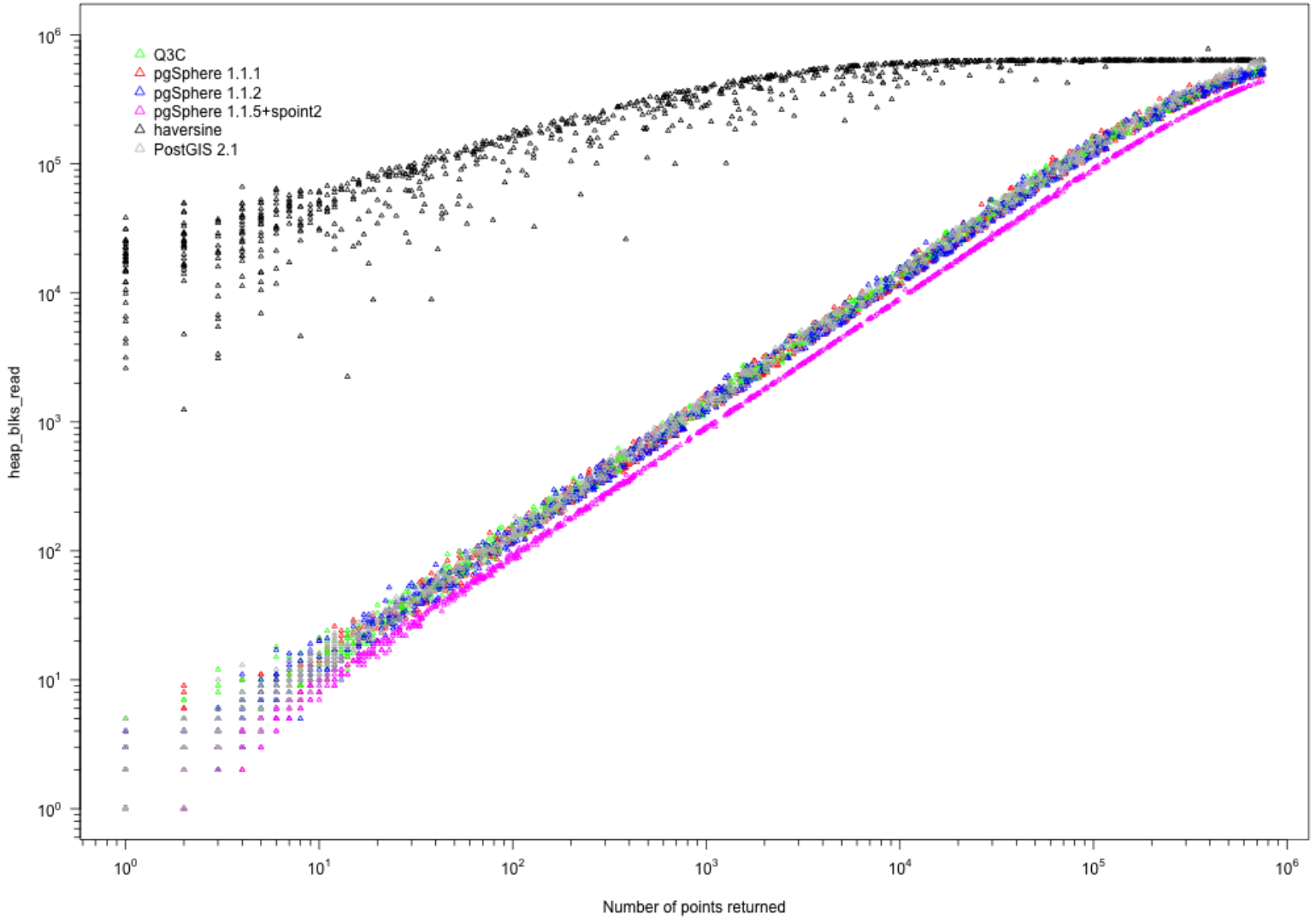
Число считанных блоков индекса при shared_buffers = 512 МБ

- Q3C
- pgSphere
  - 1.1.1
  - 1.1.2
  - 1.1.5 spoint2
- haversine
- PostGIS 2.1



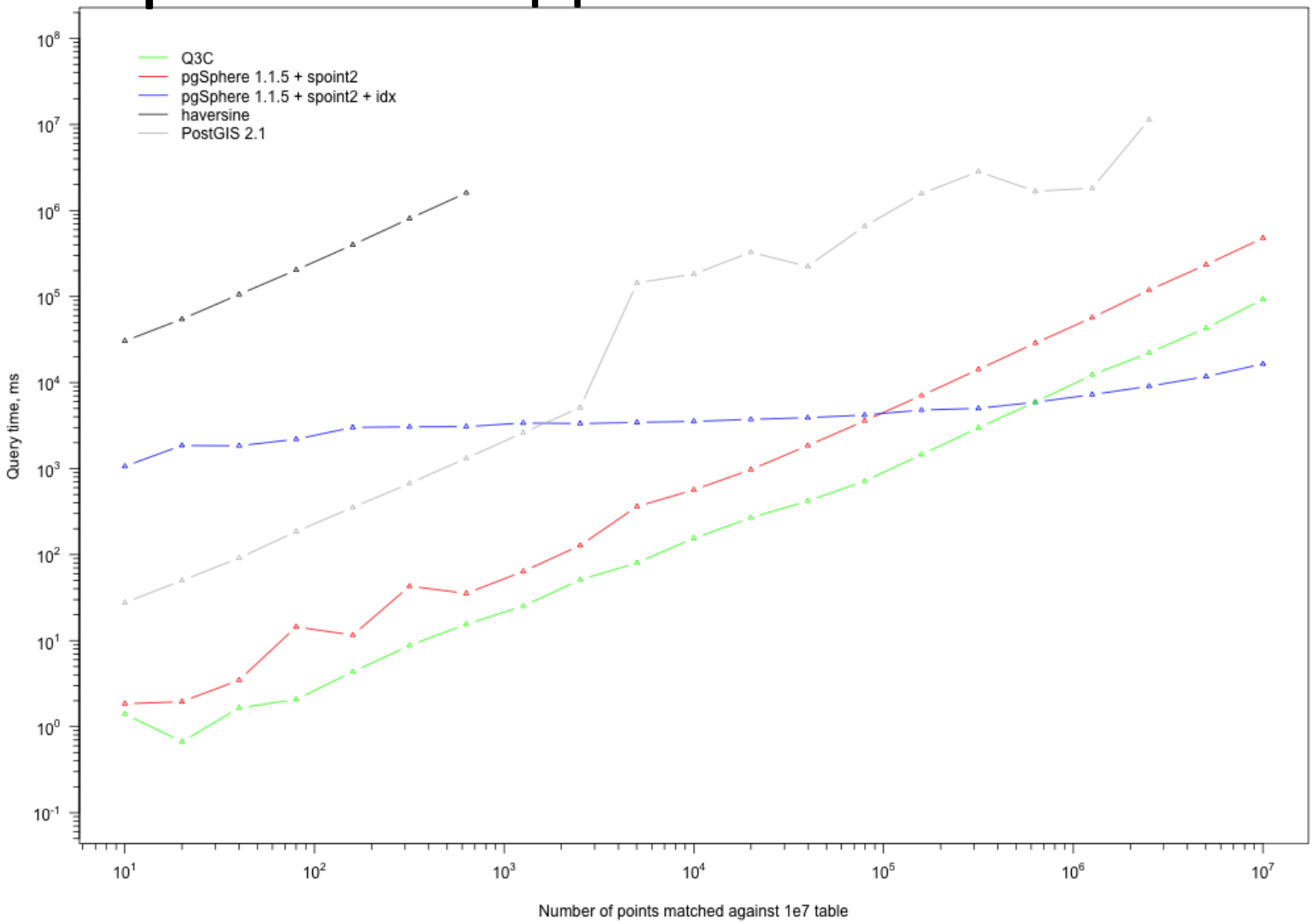
Число считанных блоков таблицы

- Q3C
- pgSphere
  - 1.1.1
  - 1.1.2
  - 1.1.5 spoint2
- haversine
- PostGIS 2.1



Время выполнения кросс-отождествления

- Q3C
- pgSphere
  - 1.1.5 spoint2
  - 1.1.5 spoint2 idx
- haversine
- PostGIS 2.1



Что не так с PostGIS?

- Алгоритм разделения узла
- Ошибки при реализации класса операторов (?)

KNN-GiST

- KNN — найти K ближайших соседей (K- nearest neighbourhood)
- Очень трудная для традиционных алгоритмов поиска
  - Индекс не помогает, так как нет предиката
  - Требуется full table scan, сортировка
  - Используются разные «костыли» - range search, как задать нужный радиус ?
- Идея KNN-GiST
  - Использовать новую стратегию обхода поискового дерева (GiST) - Priority Queue вместо DFS (BFS)
  - Сразу получаем необходимые K-записей в нужном порядке
  - Чем больше таблица, тем больше выигрыш KNN-GiST

Knn: History of development

- Начало проекта - **Sep 8, 2007 at 7:54 PM**

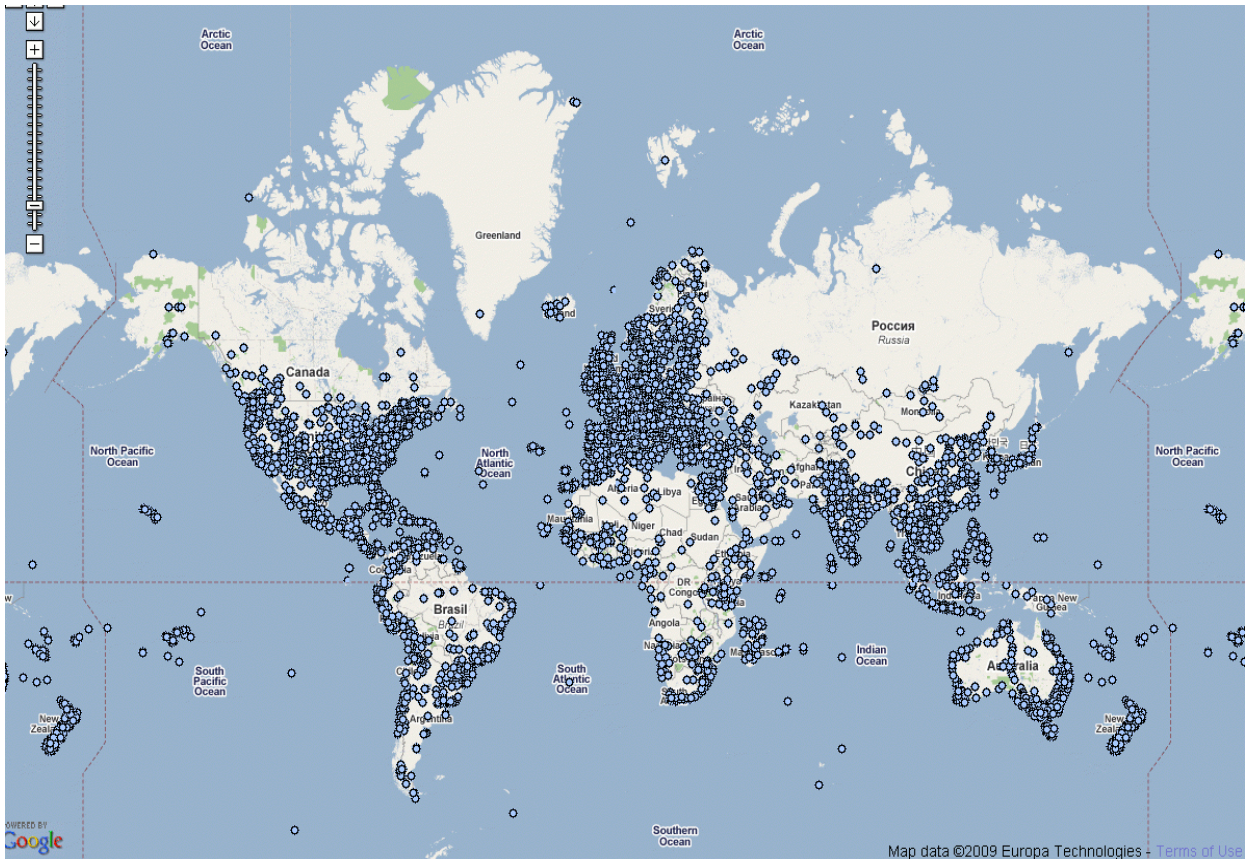
```
date  Sat, Sep 8, 2007 at 7:54 PM
subject     Chat with Sergey V. Karpov

7:36 PM me: я тут knn-search занимаюсь, масса интересного. Все думаю,
как в постгресе это поиметь
 Sergey: а что это такое?
7:37 PM me: k-nearest соседей - супер важная задача
  найти 5 ближайших точек
7:38 PM Sergey: ближайших к чему?
 me: к заданной точке
7:39 PM Sergey: в какой системе координат?
 me: в любой, в n-мерном пространстве. В простом варианте - хотя бы
на земле/небе
7:40 PM это нужно для поиска похожих картинок, например.
  навиный вариант повторять запросы - не катит
```

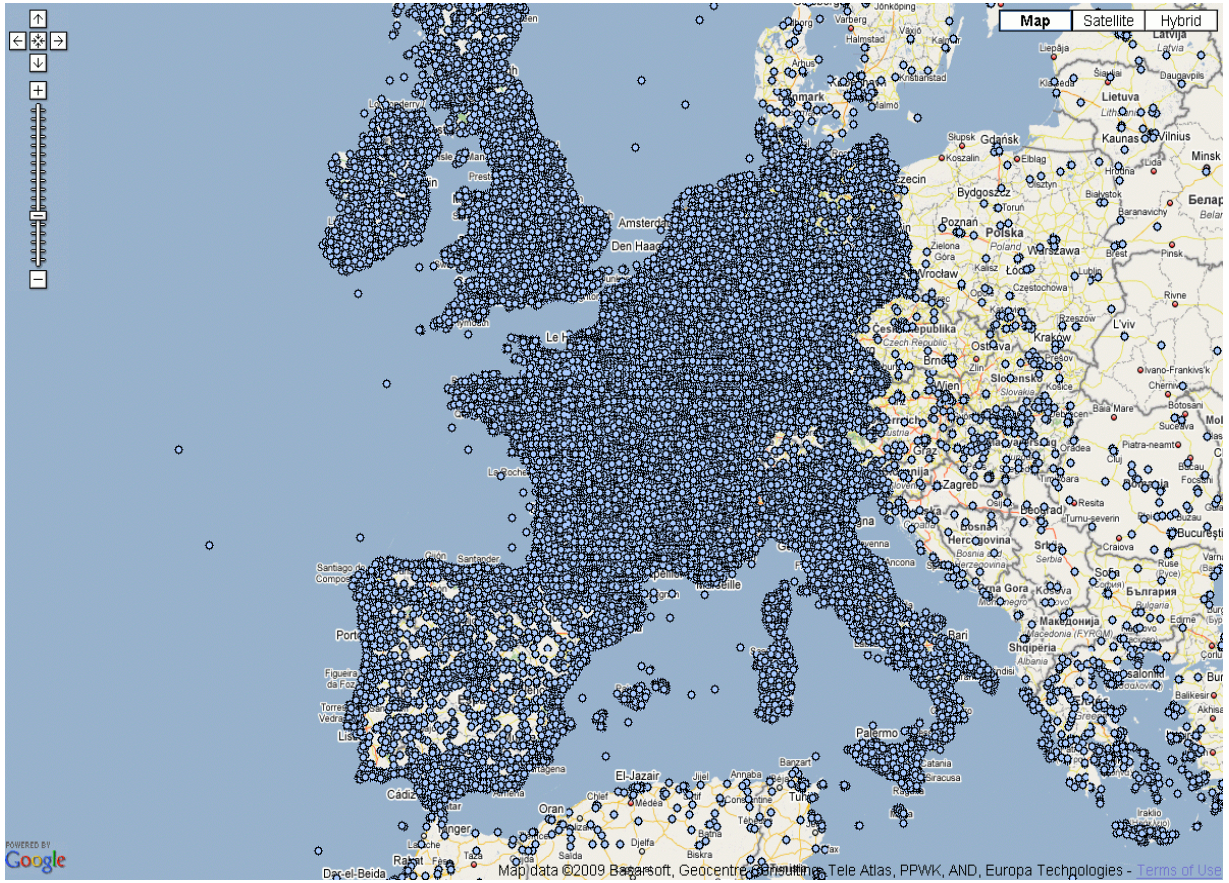
Knn: History of development

- TODO (http://www.sai.msu.su/~megera/wiki/TODO)
  начало 2008 года, уже есть понимание что делать
- 25 июля 2009 года – письмо Paul Ramsey (POSTGIS)
- 10 июля 2009 года – контракт Open Planning Project Inc.
- 20 ноября 2009 года – патч KNNGiST v.0.1 (модуль расш)
- Commitfest nightmare
  - 22 июля 2010 – KNNGiST (v.0.8), commitfest
  - 13 сентября 2010 – KNNGiST (v.0.9)
  - 03 декабря 2010 – Tom Lane committed for 9.1 !
  - 21 января 2011 – contrib/btree_gist committed !

KNN-GiST: Как найти нужный радиус ?

Рестораны





KNN-search: Examples

- Synthetic data – 1,000,000 randomly distributed points

```
create table qq ( id serial, p point, s int4);

insert into qq (p,s)  select point( p.lat, p.long), (random()*1000)::int
from ( select  (0.5-random())*180 as lat, random()*360 as long
       from ( select generate_series(1,1000000) ) as t
) as p;
create index qq_p_s_idx on qq using gist(p);
analyze qq;
```

- Query – find k-closest points to (0,0)

```
set enable_indexscan=on|off;
explain (analyze on, buffers on)

  SELECT * FROM qq ORDER BY (p <-> '(0,0)') ASC LIMIT 10;
```

- <-> - distance operator, provided by data type

KNN-search: Examples

- Выигрыш 1700 раз для K=10 !

```
 Limit (actual time=327.674..327.675 rows=10 loops=1)
   Buffers: shared hit=7353
   ->  Sort (actual time=327.673..327.674 rows=10 loops=1)
         Sort Key: ((p <-> '(0,0)'::point))
         Sort Method: top-N heapsort  Memory: 25kB
         Buffers: shared hit=7353
         ->  Seq Scan on qq (actual time=0.009..180.513 rows=1000000 loops=1)
               Buffers: shared hit=7353
 Execution time: 327.698 ms
--------------------------
 Limit (actual time=0.116..0.160 rows=10 loops=1)
   Buffers: shared hit=14
   ->  Index Scan using qq_p_s_idx on qq (actual time=0.116..0.156 rows=10 loops=1)
         Order By: (p <-> '(0,0)'::point)
         Buffers: shared hit=14
 Execution time: 0.183 ms
```

KNN-search: Examples

- Выигрыш 220 раз для K=1000 !

```
 Limit (actual time=314.262..314.471 rows=1000 loops=1)
   Buffers: shared hit=7353
   ->  Sort (actual time=314.261..314.365 rows=1000 loops=1)
         Sort Key: ((p <-> '(0,0)'::point))
         Sort Method: top-N heapsort  Memory: 127kB
         Buffers: shared hit=7353
         ->  Seq Scan on qq (actual time=0.008..172.222 rows=1000000 loops=1)
               Buffers: shared hit=7353
 Execution time: 314.599 ms
--------------------------
 Limit (actual time=0.073..1.334 rows=1000 loops=1)
   Buffers: shared hit=1016
   ->  Index Scan using qq_p_s_idx on qq (actual time=0.072..1.225 rows=1000 loops=1)
         Order By: (p <-> '(0,0)'::point)
         Buffers: shared hit=1016
 Execution time: 1.429 ms1
```

KNN-search: Examples

- Выигрыш 15600 раз для K=10 и N=10,000,000 !

```
 Limit (actual time=3212.470..3212.473 rows=10 loops=1)
   Buffers: shared hit=96 read=73434
   ->  Sort (actual time=3212.469..3212.470 rows=10 loops=1)
         Sort Key: ((p <-> '(0,0)'::point))
         Sort Method: top-N heapsort  Memory: 25kB
         Buffers: shared hit=96 read=73434
         ->  Seq Scan on qq (actual time=0.009..1827.449 rows=10000000 loops=1)
               Buffers: shared hit=96 read=73434
 Execution time: 3212.492 ms
--------------------------
 Limit (actual time=0.143..0.184 rows=10 loops=1)
   Buffers: shared read=14
   ->  Index Scan using qq_p_s_idx on qq (actual time=0.143..0.182 rows=10 loops=1)
         Order By: (p <-> '(0,0)'::point)
         Buffers: shared read=14
 Execution time: 0.205 ms
```

KNN-search: Очепятки

- Триграммы (лопата: '__л', '_ло', 'лоп', 'опа', 'пат', 'ата')
- Метрика (похожесть) S (W1, W2)
  - $N_{union}/max(N_1, N_2)$
  - $2 * N_{union}/(N_1 + N_2)$
  - $N_{union}/(N_1 + N_2 - N_{union})$
- Расстояние D (W1,W2):
  - 1 – S
  - 1/S

```
CREATE EXTENSION pg_trgm;

=# \d words
   Table "public.words"
 Column | Type | Modifiers
--------+------+-----------
 w      | text |
Indexes:
    "words_w_idx" gist (w gist_trgm_ops)
```

KNN-search: Очепятки

```
select w from words order by w <-> 'risource' limit 5;
    w
----------
 resource
 rice
 rivets
 ridden
 rivalled
(5 rows)
Limit (actual time=0.333..0.345 rows=5 loops=1)
   ->  Index Scan using words_w_idx on words (actual time=0.333..0.344 rows=5 loops=1)
         Order By: (w <-> 'risource'::text)
 Execution time: 0.366 ms
```

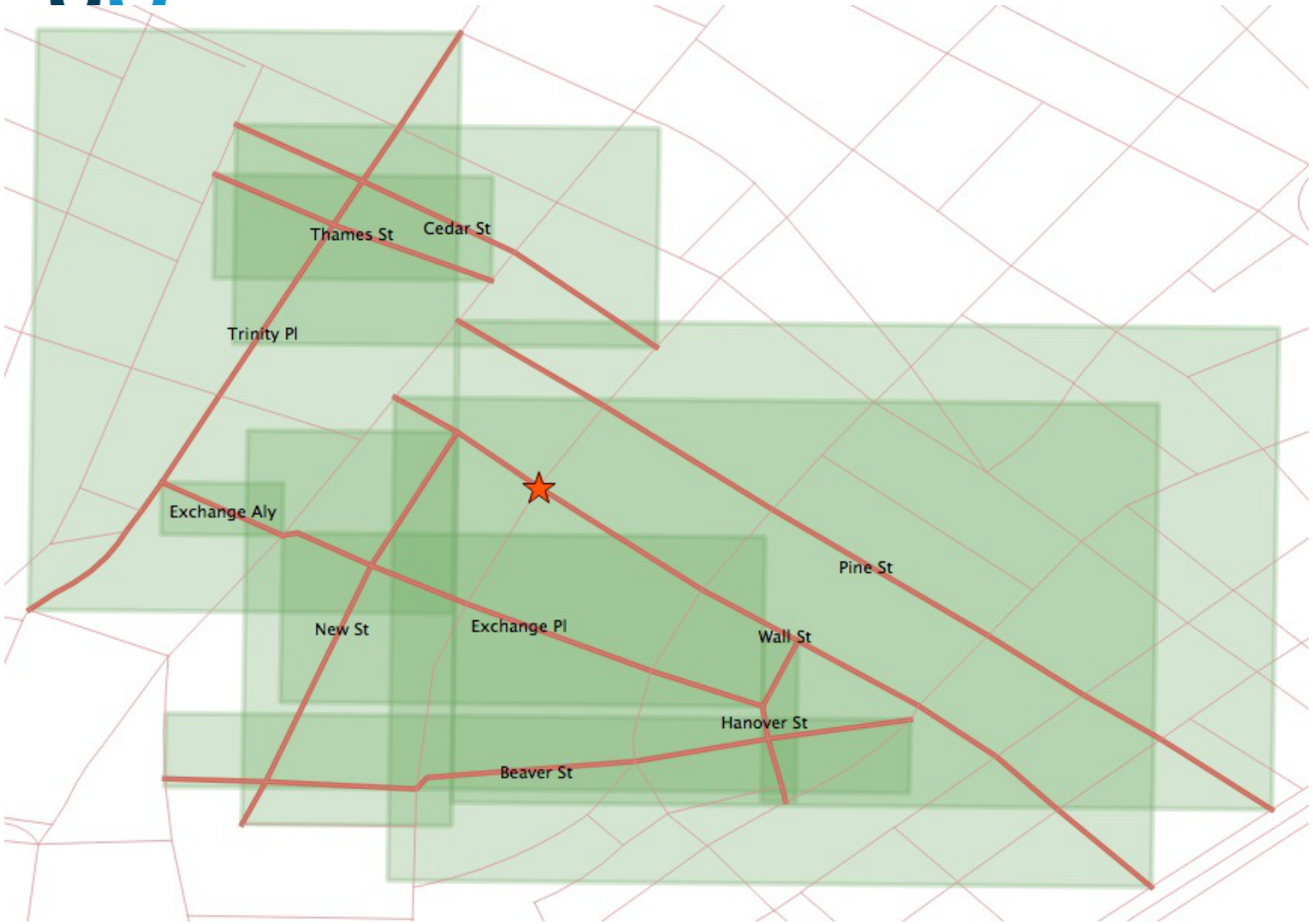
KNN-search: Очепятки (Lateral Query)

```
SELECT w.w, s.w as similar FROM words w CROSS JOIN LATERAL (SELECT w FROM words
ORDER BY w.w <-> w limit 2) s WHERE w.w LIKE 'ar%' AND w.w <> s.w;
      w       |   similar
--------------+-------------
 archiver     | anniversary
 armadillos   | armchairs
 artillery    | distillery
 arithmetized | authorized
 armchairs    | airspeed
(5 rows)
 Nested Loop (actual time=0.727..2.592 rows=5 loops=1)
   ->  Index Scan using words_w_idx on words w (actual time=0.127..0.165 rows=5 loops=1)
         Index Cond: (w ~~ 'ar%'::text)
   ->  Subquery Scan on s (actual time=0.483..0.483 rows=1 loops=5)
         Filter: (w.w <> s.w)
         Rows Removed by Filter: 1
         ->  Limit (actual time=0.321..0.481 rows=2 loops=5)
               ->  Index Scan using words_w_idx on words (actual time=0.318..0.477 rows=2 loops=5)
                     Order By: (w <-> w.w)
 Execution time: 2.633 ms
```

KNN-search: Очепятки

```
=# select w from words order by w <-> '%isourc%' limit 5;
     w
----------
 resource
 iron
 into
 idles
 injure
(5 rows)
 Limit (actual time=0.251..0.261 rows=5 loops=1)
   ->  Index Scan using words_w_idx on words (actual time=0.251..0.260 rows=5 loops=1)
         Order By: (w <-> '%isourc%'::text)
 Execution time: 0.282 ms
```

KNN-Search: Ограничения



Как найти ближайшие улицы, зная только их MBR?

Расстояние до границы MBR или до центра MBR даёт лишь

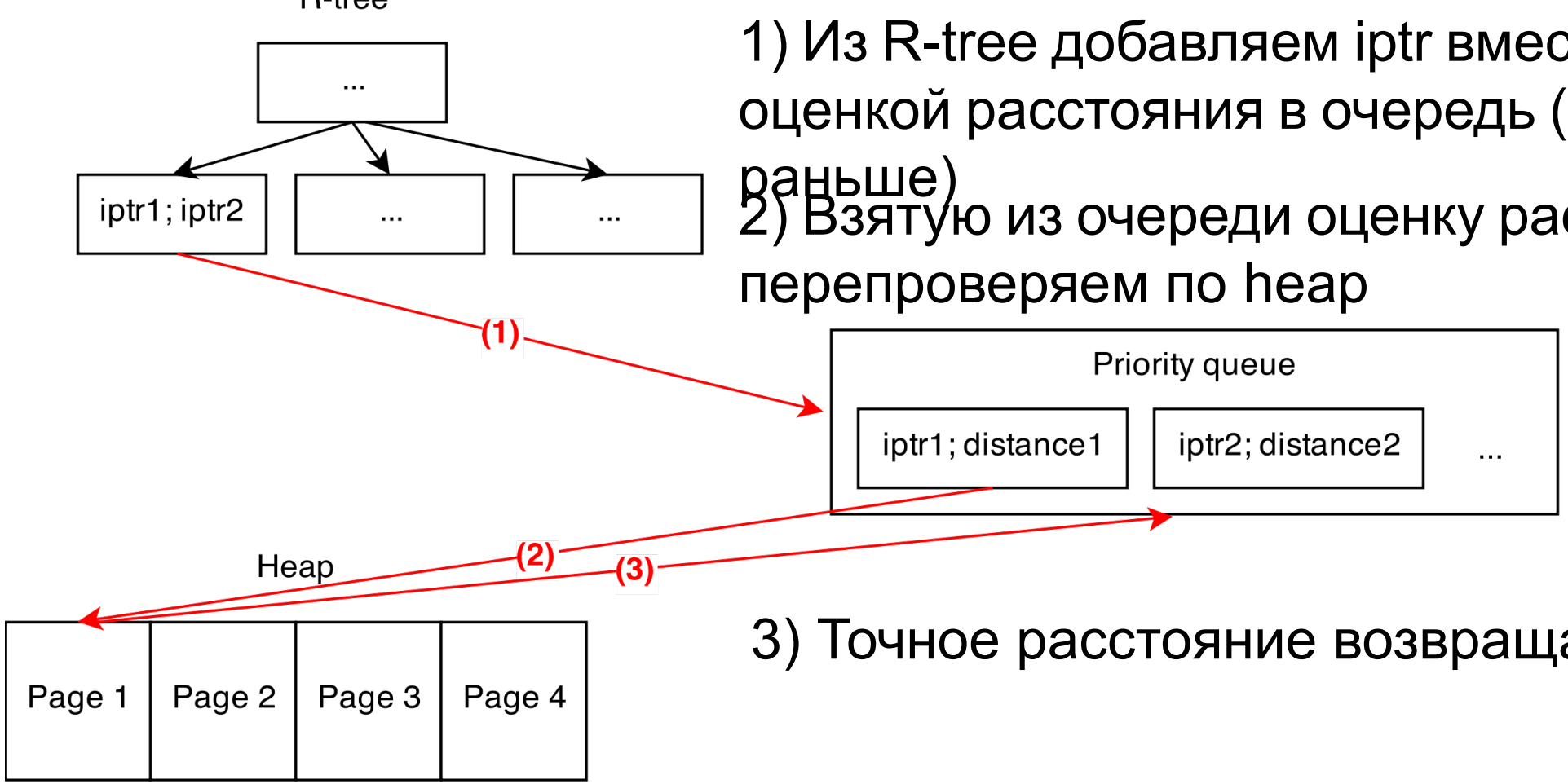
KNN-Search: Ограничения

```
WITH closest_candidates AS (
  SELECT streets.gid, streets.name, streets.geom
  FROM nyc_streets streets
  ORDER BY streets.geom <->
    'SRID=26918;POINT(583571.905921312 4506714.34119218)'::geometry
  LIMIT 100
)
SELECT gid, name
FROM closest_candidates
ORDER BY ST_Distance(geom,
    'SRID=26918;POINT(583571.905921312 4506714.34119218)'::geometry)
LIMIT 1;
```

User-level hack.

Откуда известно, что 100 оптимальное значение?

Решение: Exact KNN



1) Из R-tree добавляем iptr вместе с оценкой расстояния в очередь (как и было раньше)

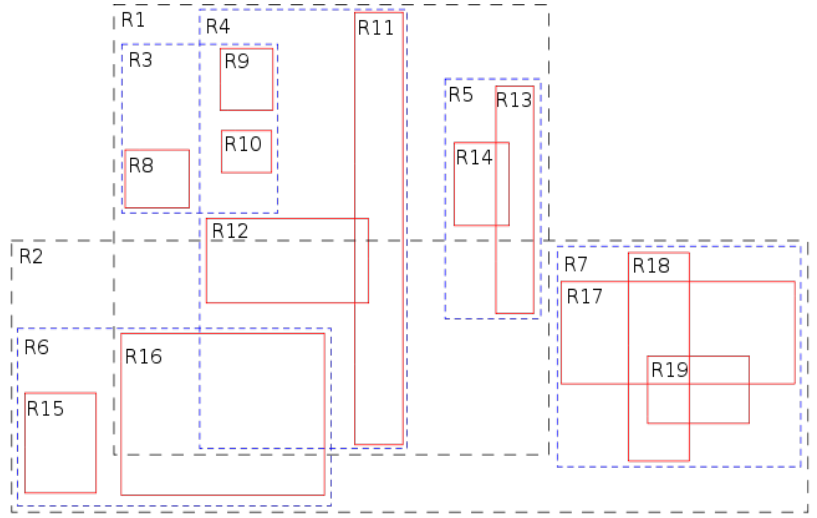
2) Взятую из очереди оценку расстояния перепроверяем по heap

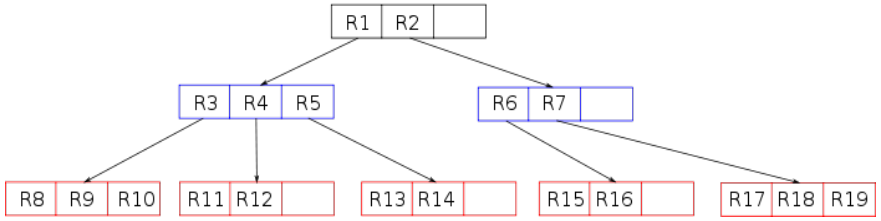
3) Точное расстояние возвращаем в heap

Exact KNN: статус

- Даёт правильный результат за минимально возможное время.
- Выложен патч на commitfest.
- Есть много архитектурных вопросов (доступ к heap из index scan).

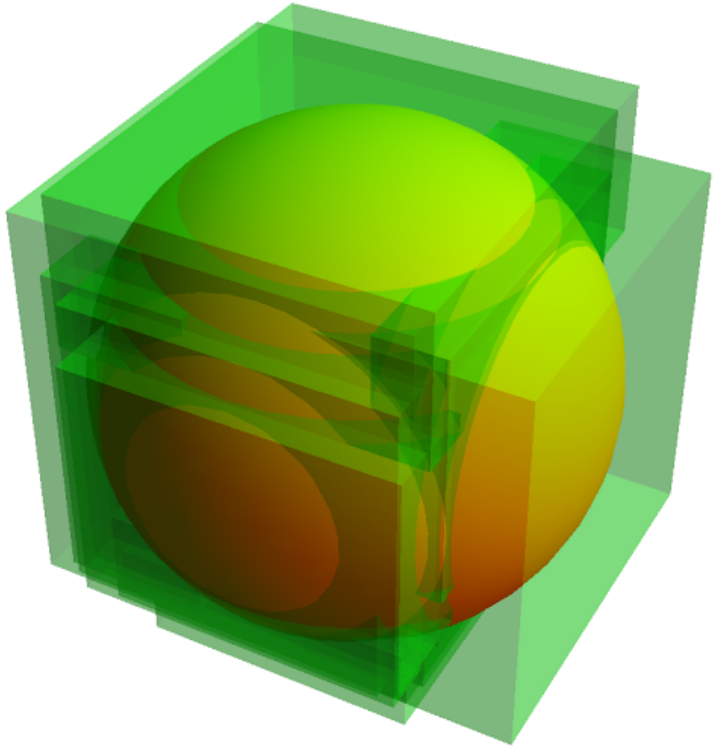
Spaghetti indexing …





R-tree fails here — bounding box of each separate spaghetti is the same

Spaghetti indexing …



R-tree fails here — bounding box of each separate spaghetti is the same

Idea: Use multiple boxes

Вывод

- СУБД PostgreSQL предоставляет богатые возможности для работы с ПД, в т.ч. в сферической системе координат, и позволяет эффективно выполнять различные виды поиска по ним.
- Перспективные направления работы:
  - Exact KNN
  - Spatial join
  - Multiple index tuples per heap tuple (VODKA index)

Postgres PROFESSIONAL

Спасибо за внимание!