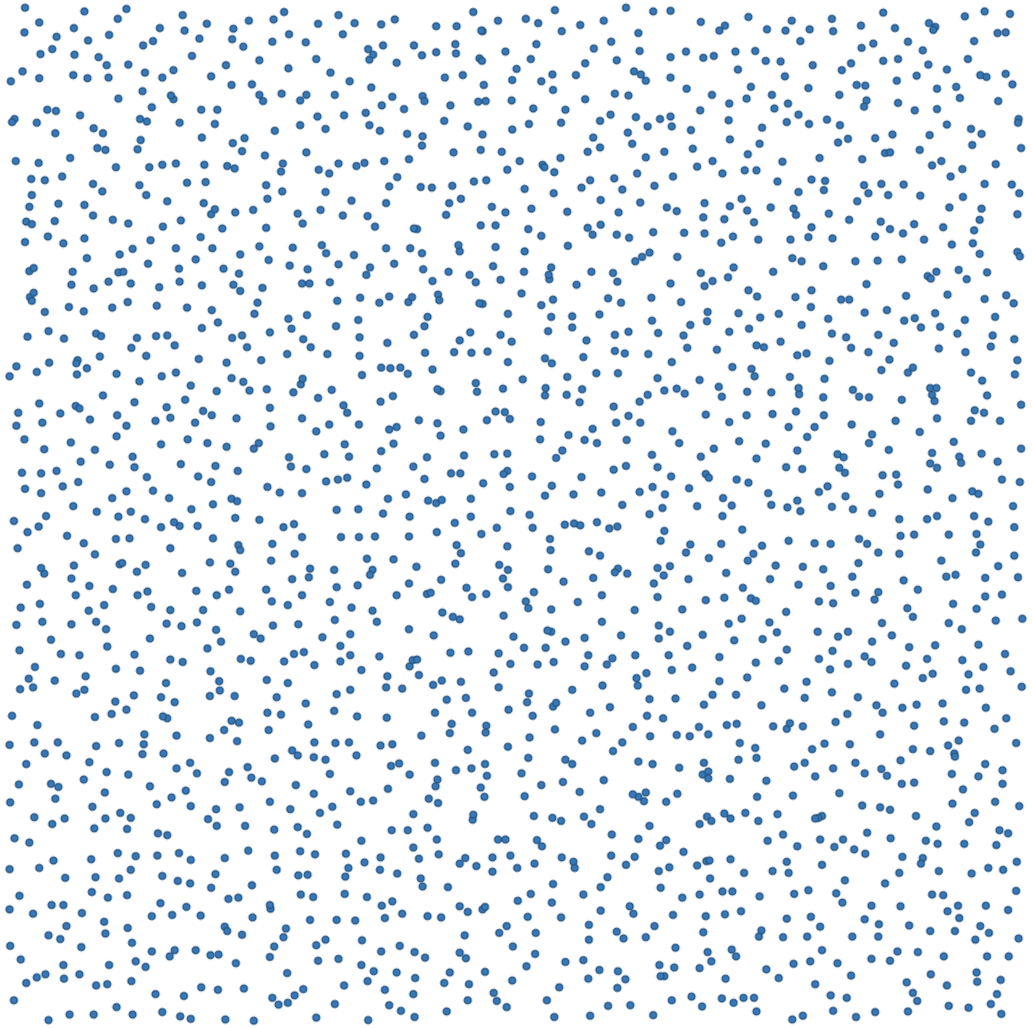
Быстрый метод создания GiST индекса в PostgreSQL

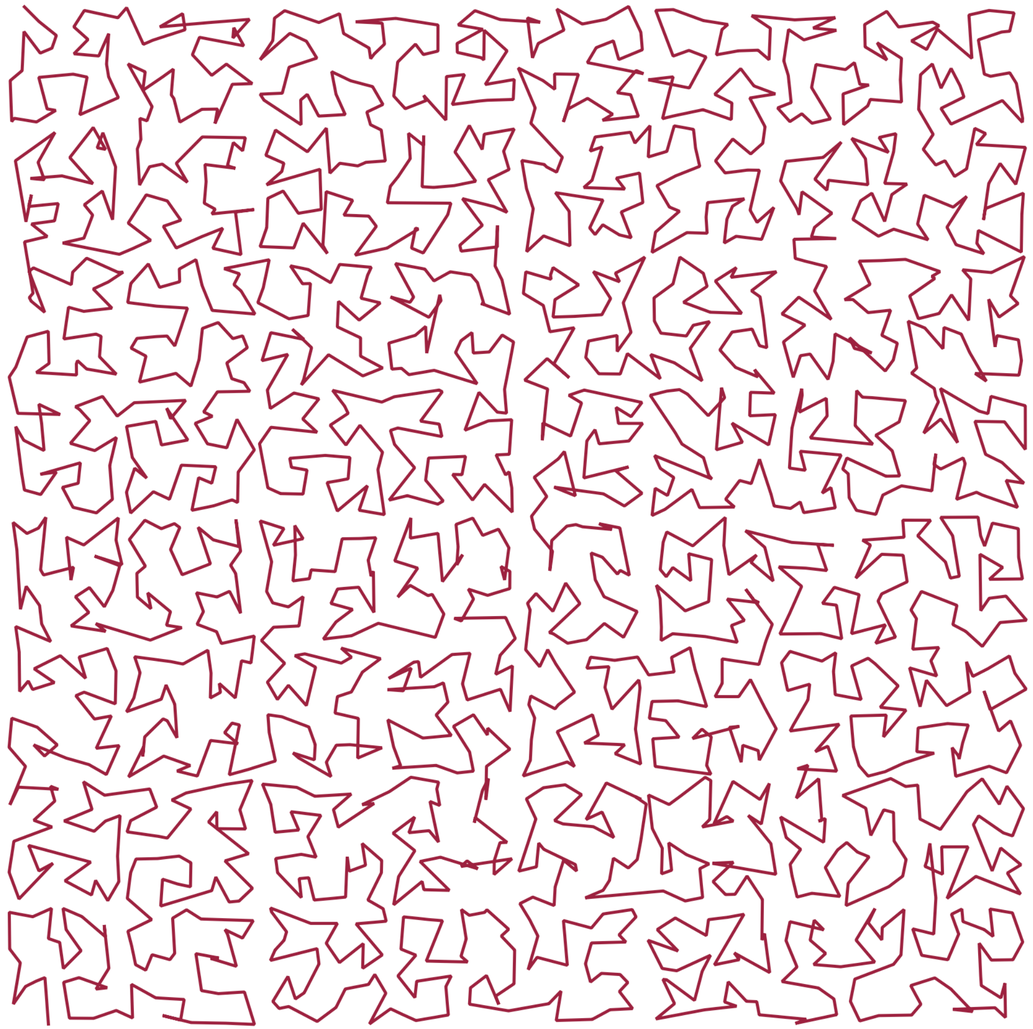
Методы создания GiST индекса в PostgreSQL

- До Postgres 14 GiST индекс можно было создавать только методом вставки элементов по одному и был режим buffering, который ускорял этот метод за счет сокращения количества операций случайного чтения/записи с диска.
- В Postgres 14 появился сортировочный метод создания GiST индекса
- В PostGIS 3.2 была добавлена поддержка сортировочного метода для 2D геометрии

Как работает построение GiST индекса методом сортировки в PG14

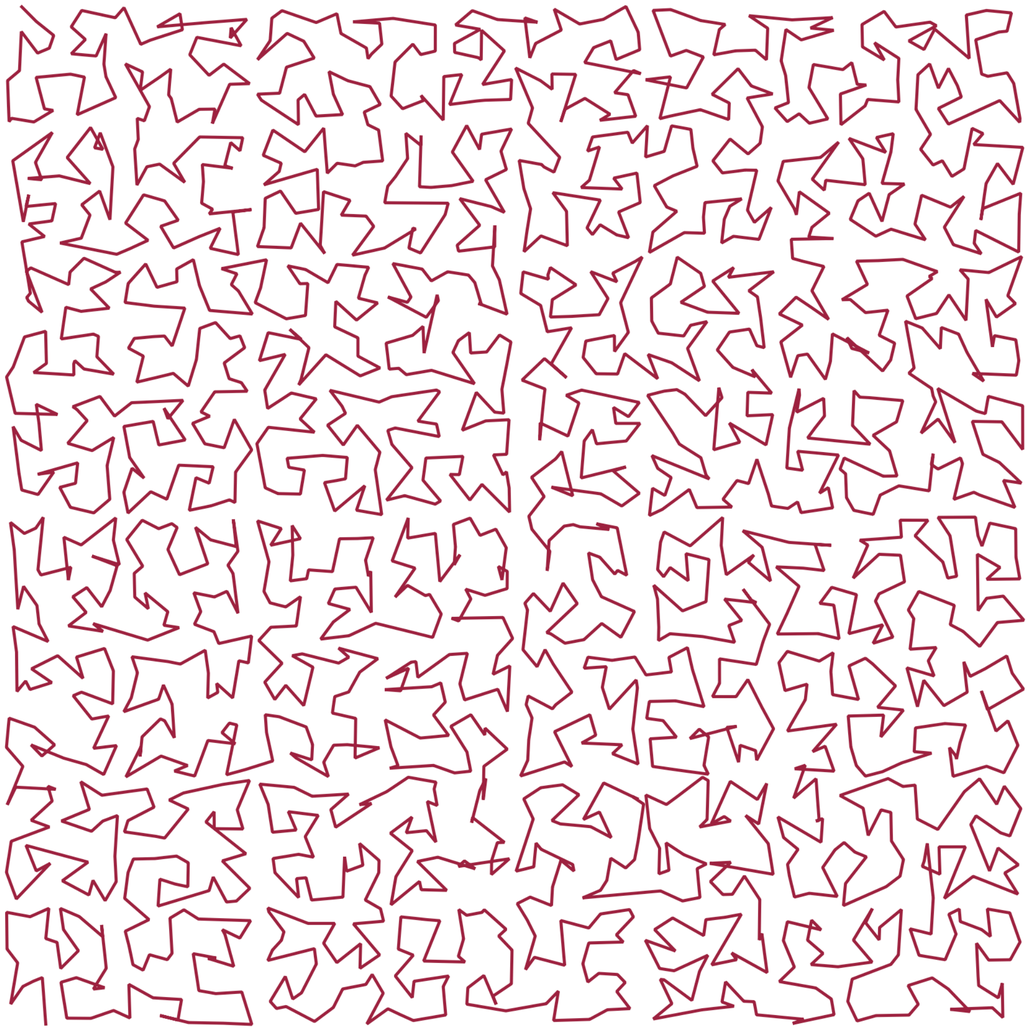
1. Сортировка элементов. Для геометрии используется кривая Гильберта.

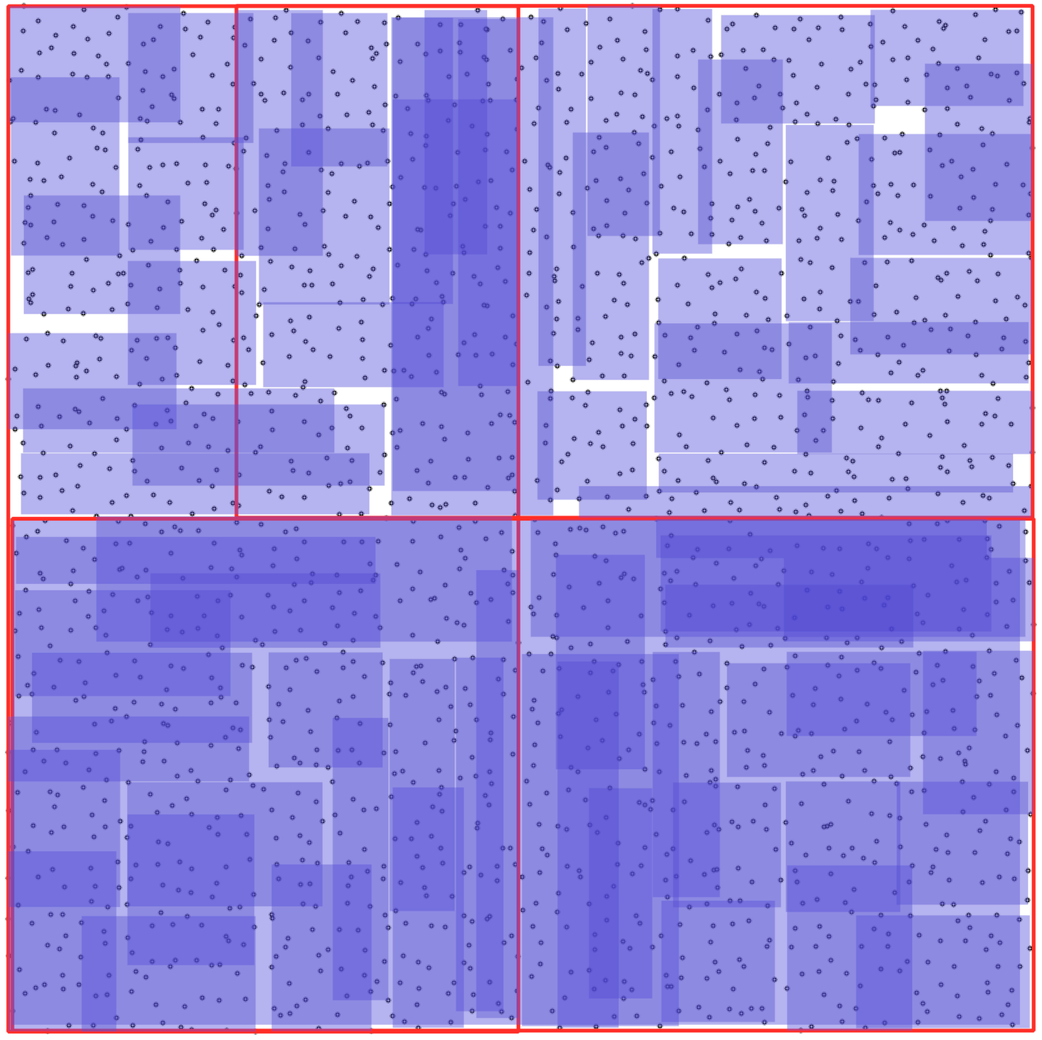




Как работает построение GiST индекса методом сортировки в PG14

2. Заполнение индексных страниц элементами в отсортированном порядке





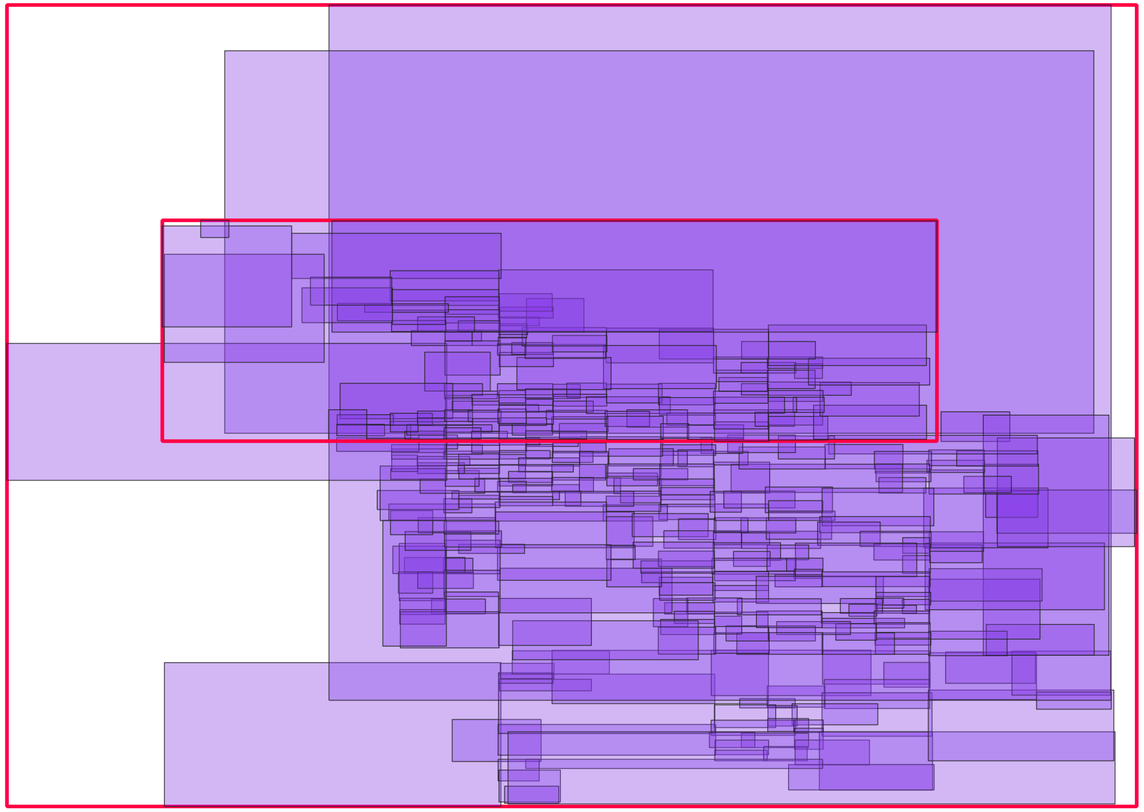
Везде выигрываем, нигде не проигрываем?

- create index roads_rdr_idx on roads_rdr using gist(geom);
  PG14 / WITH SORTSUPPORT / CREATE 200 ms
  PG14 / NO SORTSUPPORT / CREATE 1705 ms

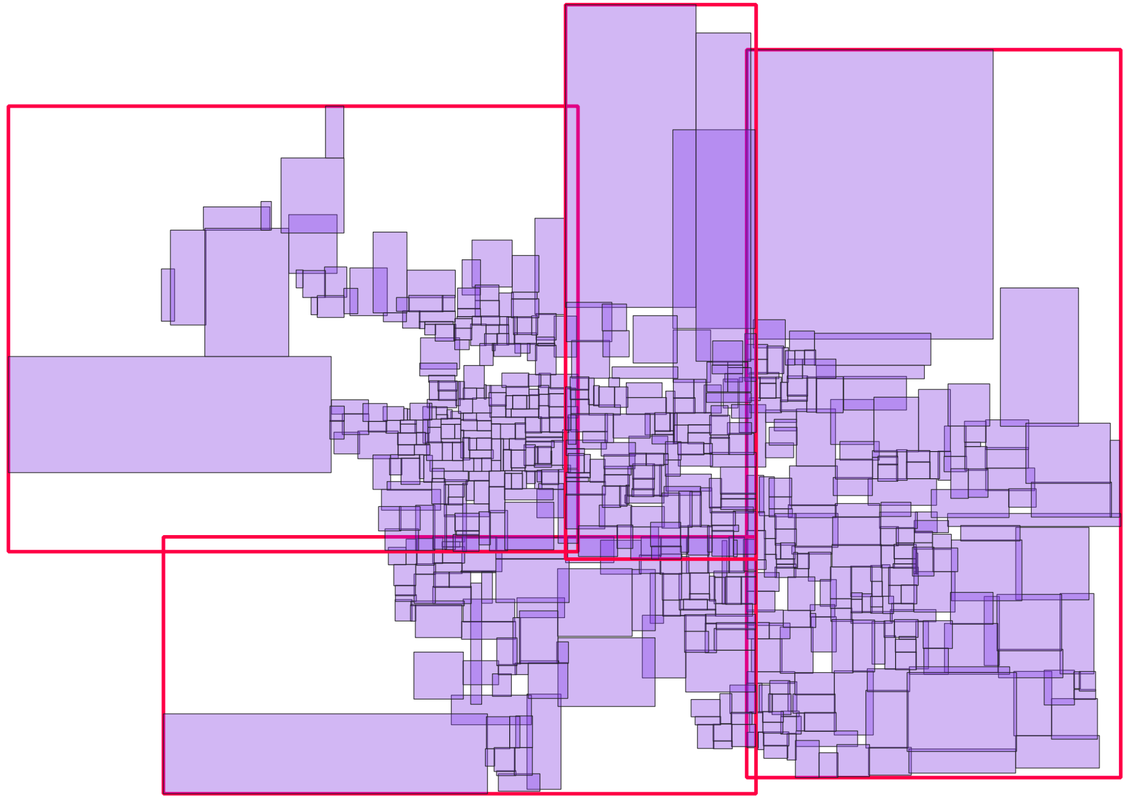
- select pg_relation_size('roads_rdr_idx');
  PG14 / WITH SORTSUPPORT / IDXSIZE 2940928 kb
  PG14 / NO SORTSUPPORT / IDXSIZE 5439488 kb

- select count(*) from roads_rdr a, roads_rdr b where a.geom && b.geom;
  PG14 / WITH SORTSUPPORT / QUERY 11200 ms
  PG14 / NO SORTSUPPORT / QUERY 4700 ms

В чем проблема?

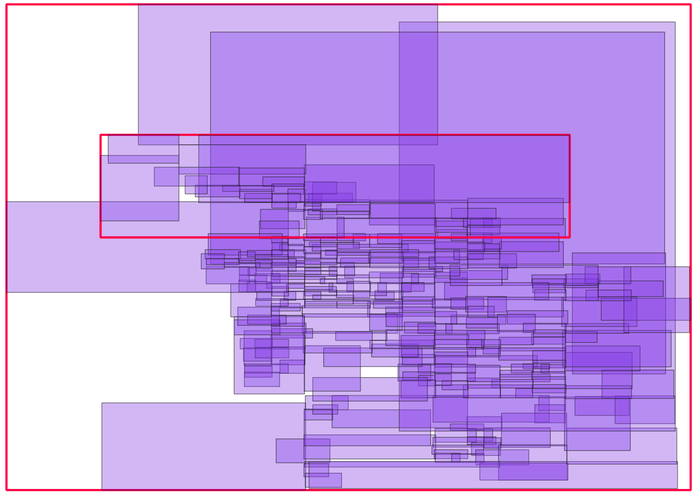


Индекс созданный методом сортировки

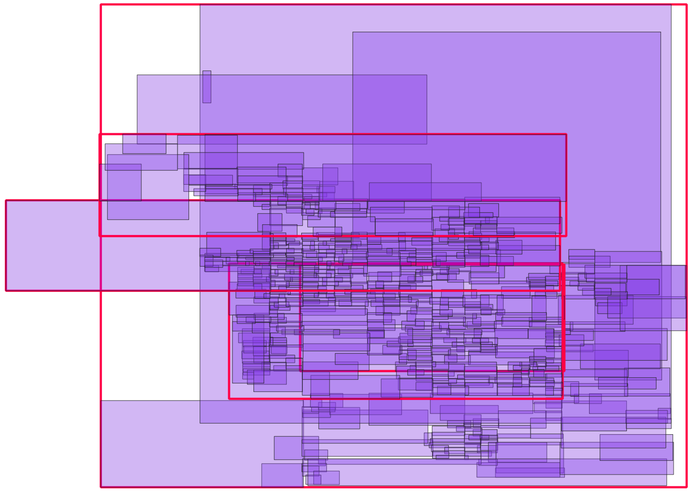


Индекс созданный обычным методом

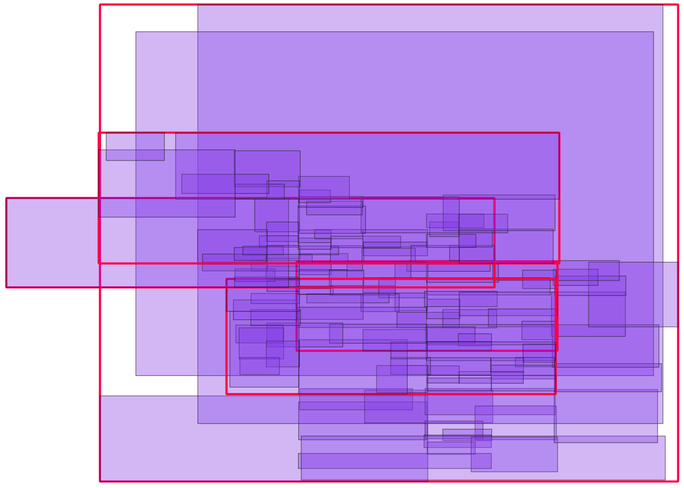
Зависимость между скоростью выполнения запросов и степенью заполнения (fillfactor) страниц индекса



fillfactor = 100
self-join = 3120.143 ms
index size = 2.6 mb



fillfactor = 50
self-join = 2346.177 ms
index size = 5.3 mb



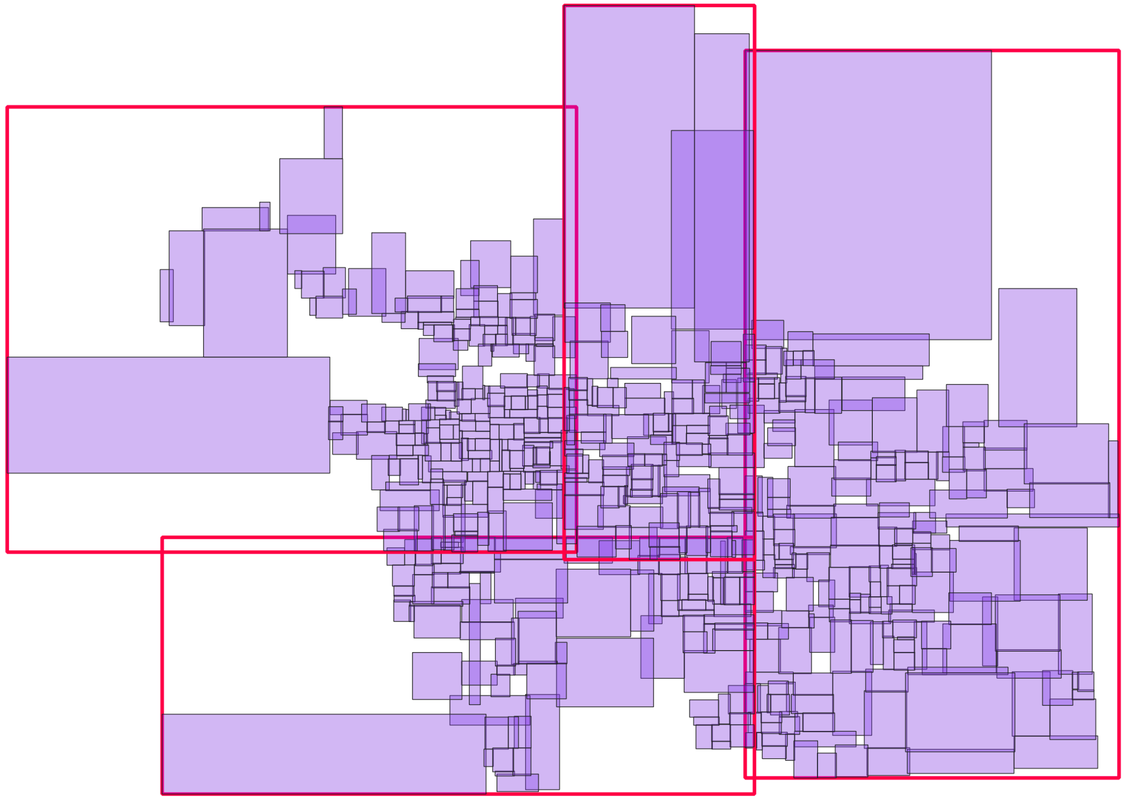
fillfactor = 10
self-join = 1019.454 ms
index size = 29.2 mb

Рассмотренные варианты решения

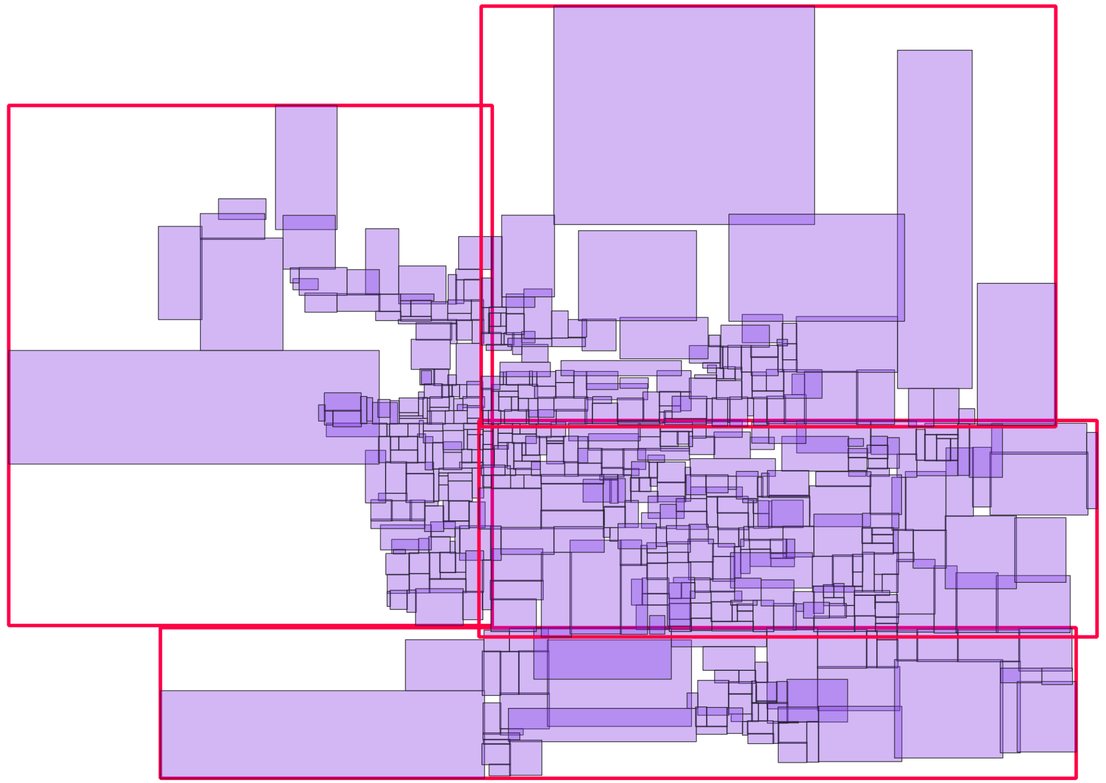
- Искать удачное место для разделения страницы индекса заполняя до fillfactor / 2 и дальше определить начинать ли новую страницу проверяя пенальти.

- Формировать сразу N страниц индекса и из них выбирать самую удачную для вставки следующего элемента с помощью penalty. Если страница переполняется, то она разделяется на несколько страниц с помощью вызова picksplit. Если количество страниц после разделения > N, то записать на диск те, вставка в которые происходила позже всего.

- Накапливать элементы в буфер размером в несколько страниц и далее разделять его на страницы индекса с помощью picksplit.



Индекс созданный обычным методом



Индекс созданный методом сортировки после патча

- https://commitfest.postgresql.org/36/3488/

Бенчмарки

	original	patched	nosortsupport	original, fillfactor=50	patched, fillfactor=50,	nosortsupport, fillfactor=50
Create index	51.2429 / 49.085	165.6421 / 164.4735	273.2494 / 270.9475	55.5905 / 54.0845	136.4715 / 135.3335	233.9812 / 232.115
Self-join	4100.5645 / 4093.227	1752.5885 / 1752.0995	1709.3702 / 1708.841	2951.8764 / 2943.608	1805.9543 / 1799.056	1183.0967 / 1181.63
Tiling	458.5242 / 455.017	418.7391 / 416.4425	424.466 / 424.548	441.8033 / 436.993	426.9725 / 422.82	417.6006 / 416.346
kNN k=1	401.8068 / 401.1595	242.7706 / 241.64	248.6685 / 247.928	306.6242 / 304.366	240.7218 / 239.898	210.1498 / 209.614
kNN k=100	2300.7032 / 2277.1295	2035.3501 / 2035.1155	2072.0323 / 2070.766	2144.8499 / 2143.695	2053.8628 / 2052.981	2033.3401 / 2030.6085

	original	patched	nosortsupport	original, fillfactor=50	patched, fillfactor=50,	nosortsupport, fillfactor=50
Number of levels	3	3	3	3	3	3
Number of pages	359	605	671	650	615	1208
Number of leaf pages	356	600	666	644	609	1194
Number of tuples	93034	93280	93346	93325	93290	93883
Number of invalid tuples	0	0	0	0	0	0
Number of leaf tuples	92676	92676	92676	92676	92676	92676
Total size of tuples	2609260 bytes	2619100 bytes	2621740 bytes	2620900 bytes	2619500 bytes	2643220 bytes
Total size of leaf tuples	2599200 bytes	2602128 bytes	2602920 bytes	2602656 bytes	2602236 bytes	2609256 bytes
Total size of index	2940928 bytes	4956160 bytes	5496832 bytes	5324800 bytes	5038080 bytes	9895936 bytes
Effective fill	0.8872	0.5285	0.477	0.4922	0.5199	0.2671

Выводы

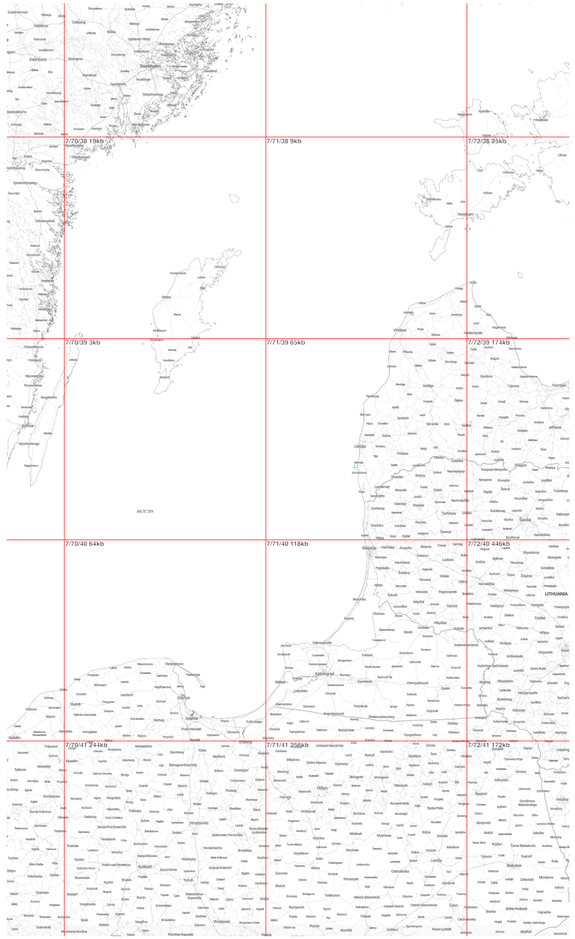
- В Postgres 14 с PostGIS 3.2 можно использовать построение GiST индекса методом сортировки, но стоит уменьшать fillfactor, чтобы получать приемлемую скорость выполнения запросов в ущерб размеру индекса.
- Патч с усовершенствованным алгоритмом построения индекса методом сортировки принят сообществом и будет доступен (?) в Postgres 15.
- Построение индекса методом сортировки будет поведением по-умолчанию в PostGIS 3.3.

What’s new in PostGIS 3.2

Performance improvements for MVT generation

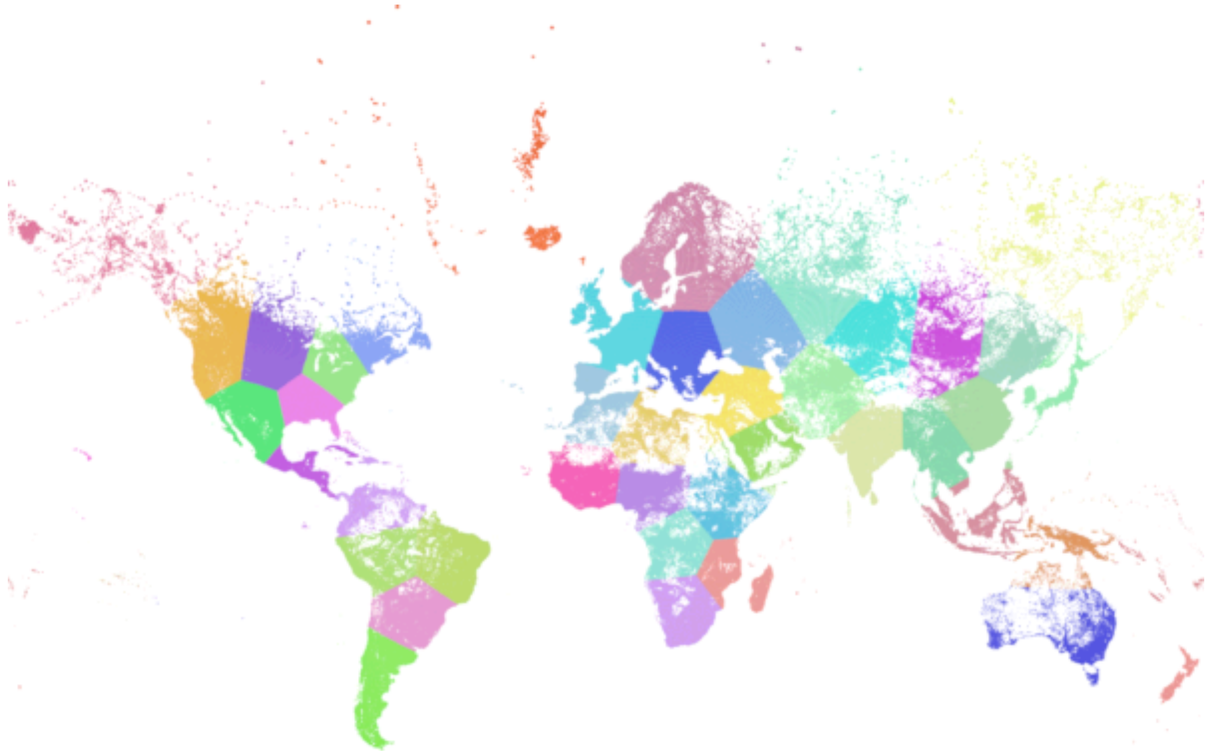
ST_AsMVT, ST_AsMVTGeom

- wagyu’s quick_lr_clip is replaced with lwgeom_clip_to_ordinate_range. 10-20% faster ST_AsMVTGeom
- ST_RemoveRepeatedPoints is O(NlogN) instead of O(N^2) for MULTIPOINT



ST_ClusterKMeans now supports max_radius argument

- Use it when you're not sure what is the number of clusters but you know what the size of clusters should be.



ST_MakeValid improvements