Быстрый метод создания GiST индекса в PostgreSQL

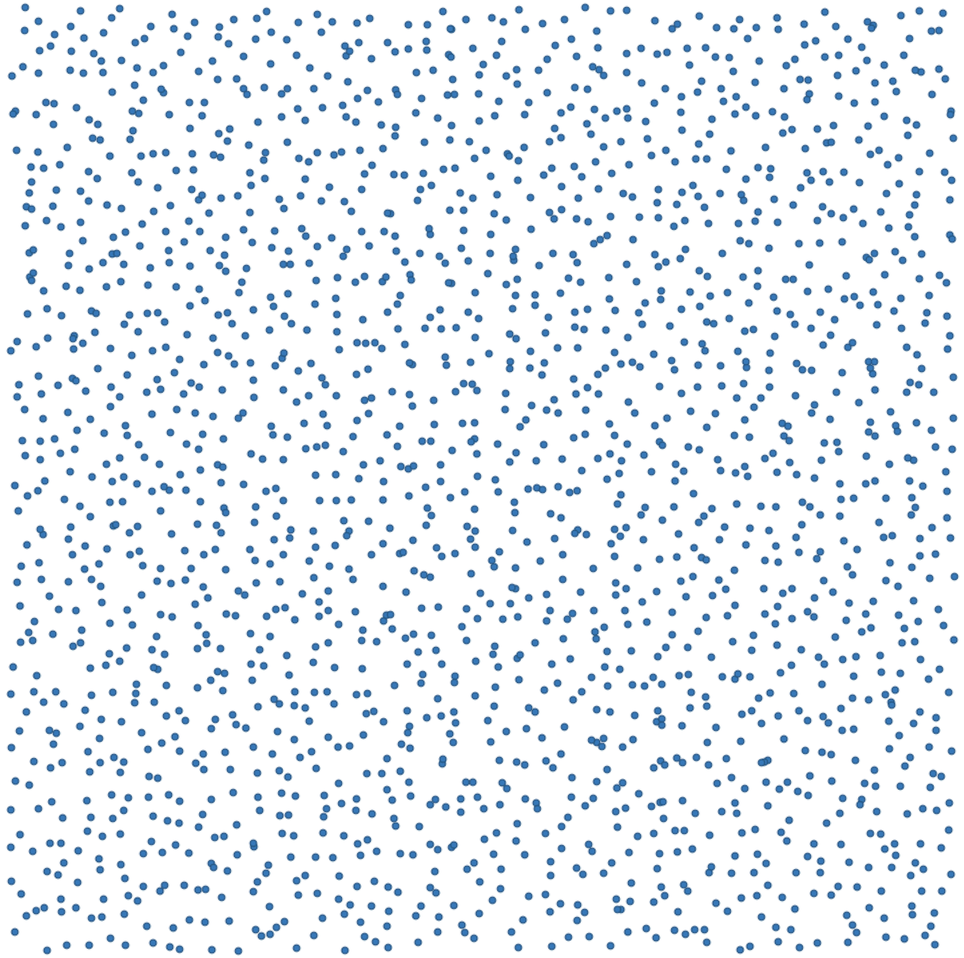
KONTUR

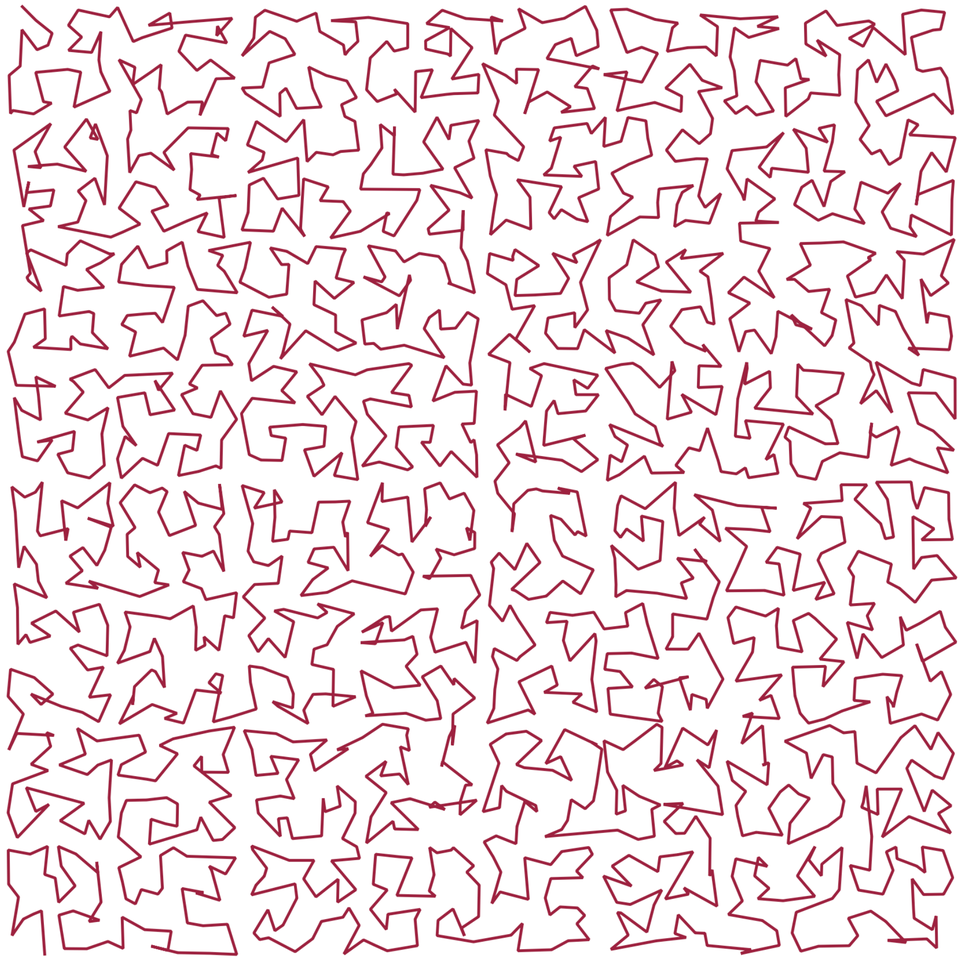
Методы создания GiST индекса в PostgreSQL

- 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.

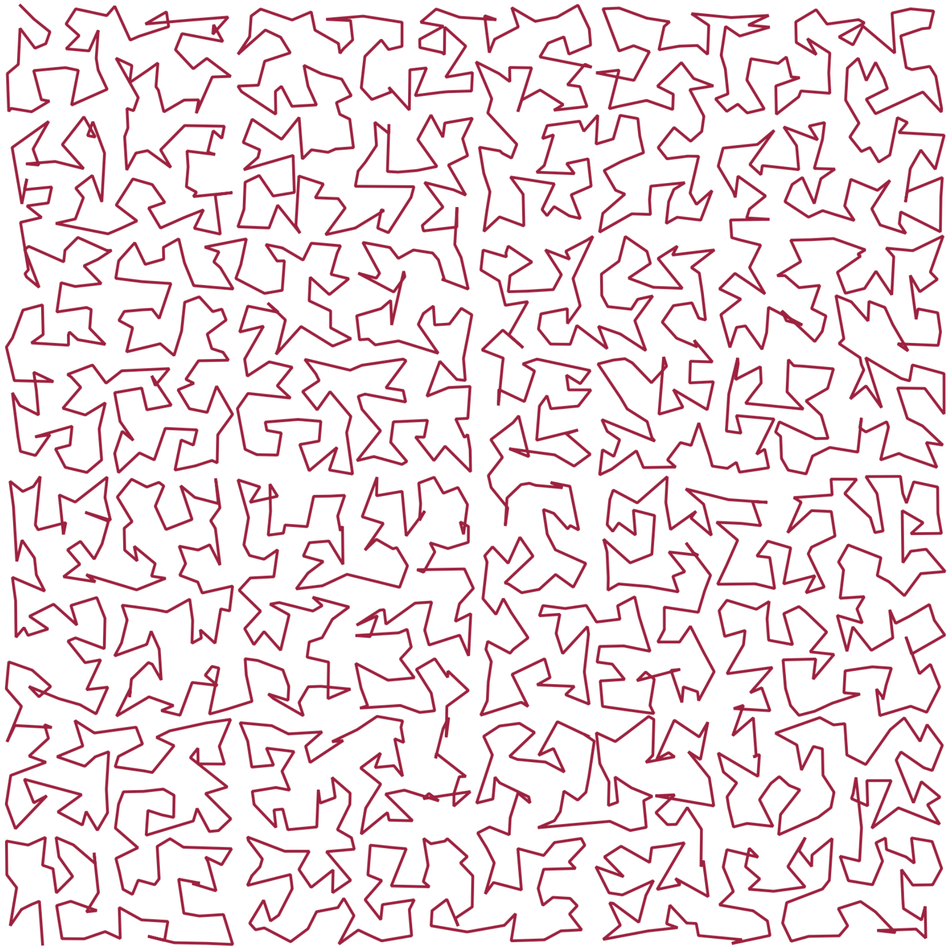


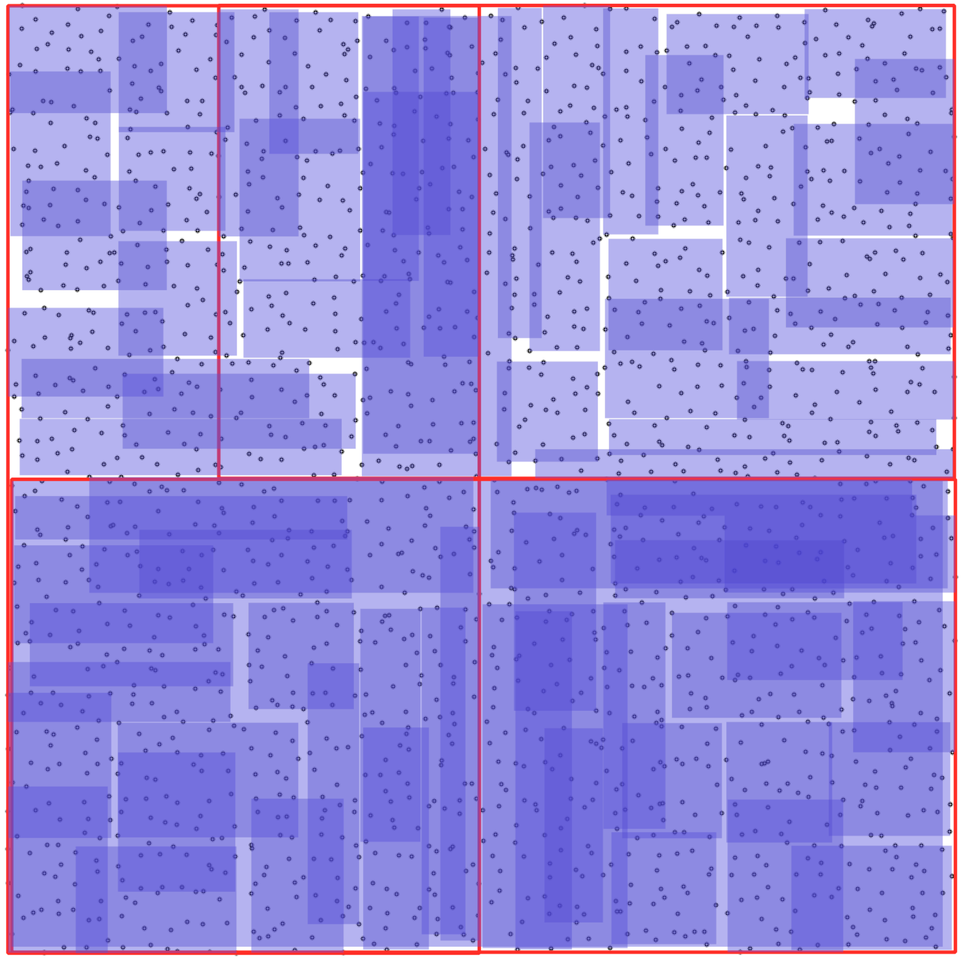


KONTUR

Как работает построение GiST индекса методом сортировки в PG14

2. Заполнение индексных страниц элементами в отсортированном порядке





KONTUR

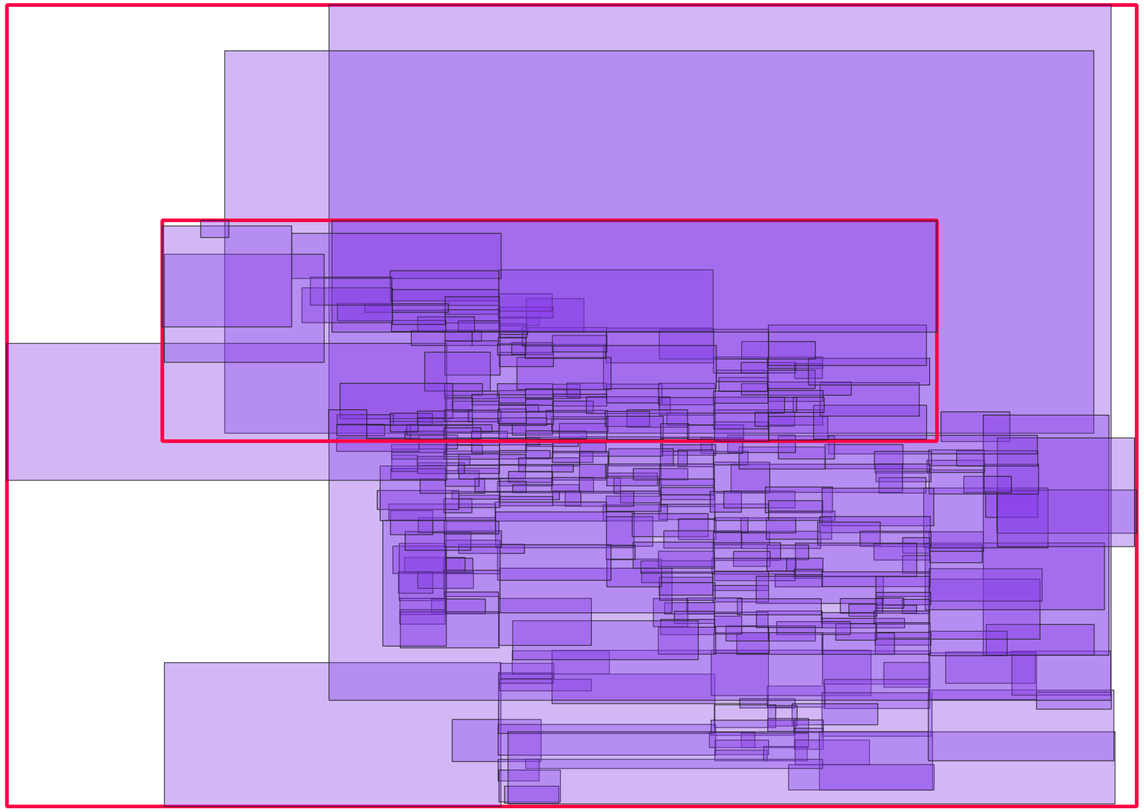
Везде выигрываем, нигде не проигрываем?

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

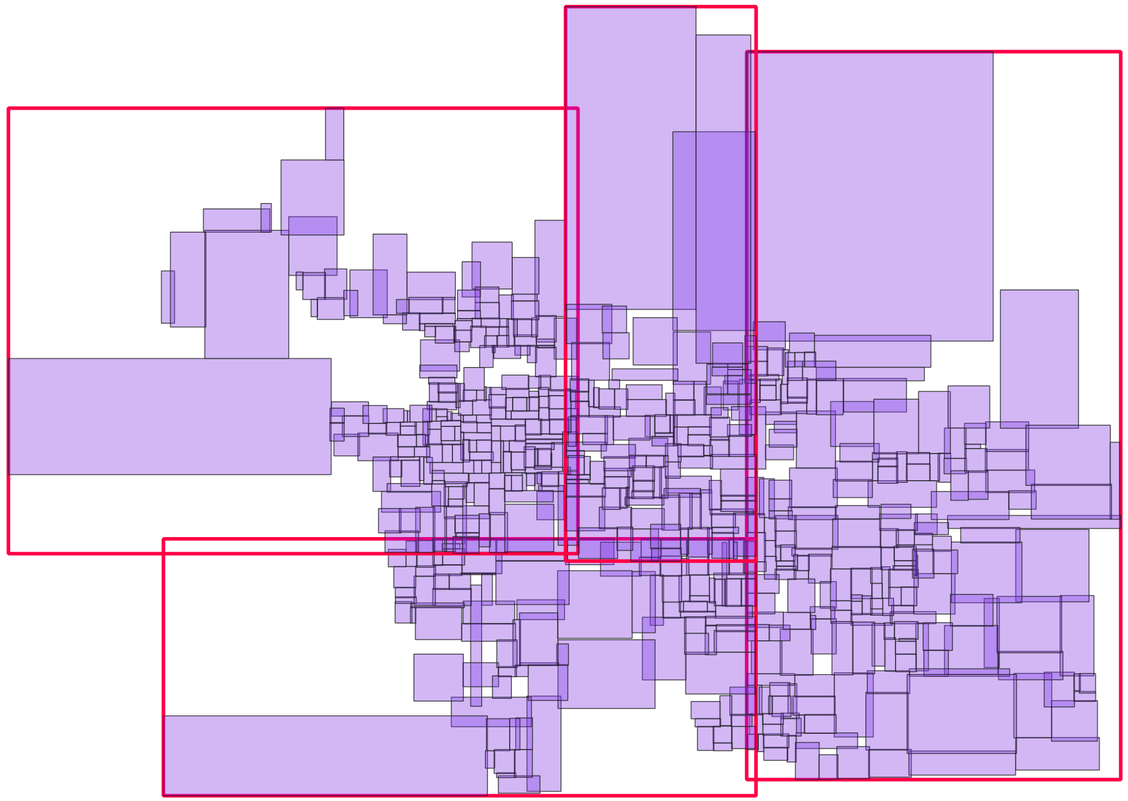
- select pg_relation_size('roads_rdr_idx');
  PG14 / WITH SORTSUPPORT / IDXSIZE 2940928 kb
  PG14 / NO SORTSUPPORT / IDXSIZE 5439488 kb

- 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

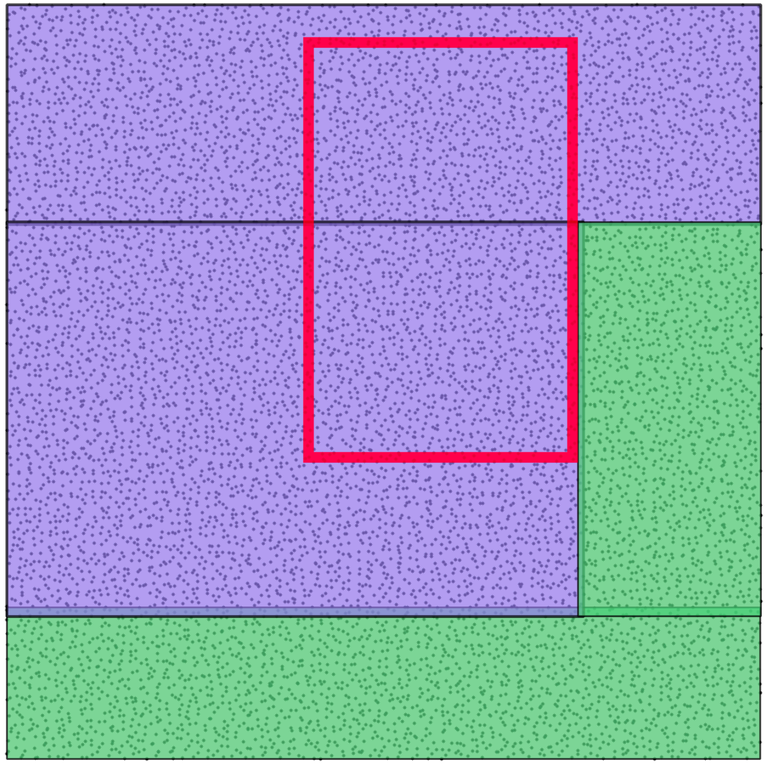


Индекс созданный обычным методом

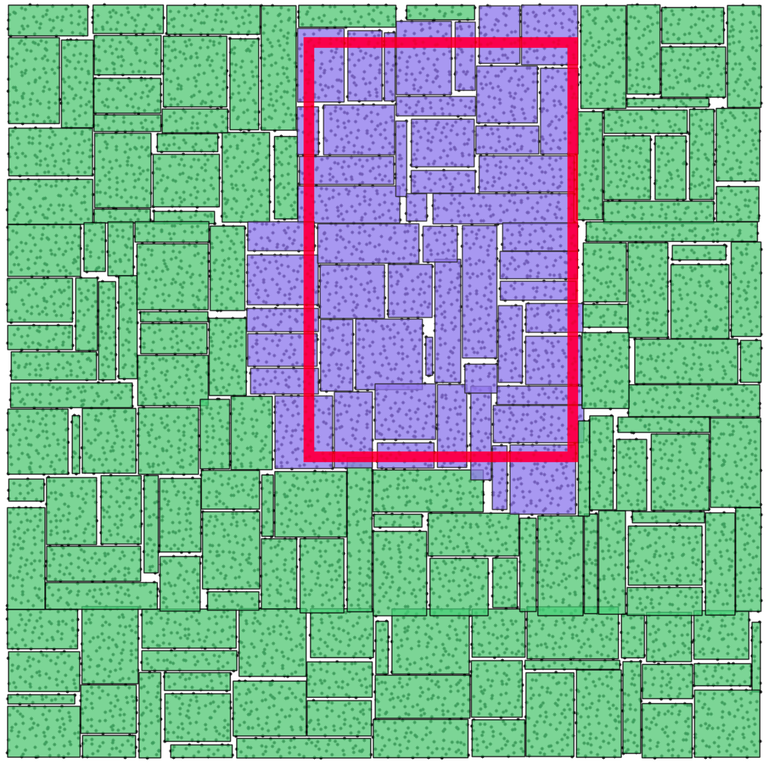
KONTUR

Как происходит сканирование GiST индекса

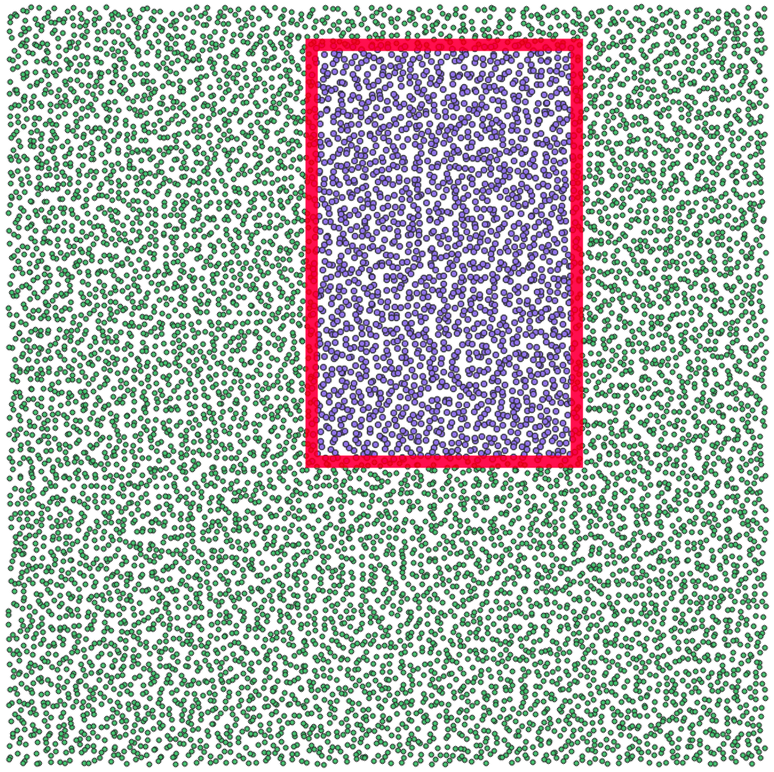
Стратегия: overlap (&&)



Level 1



Level 2

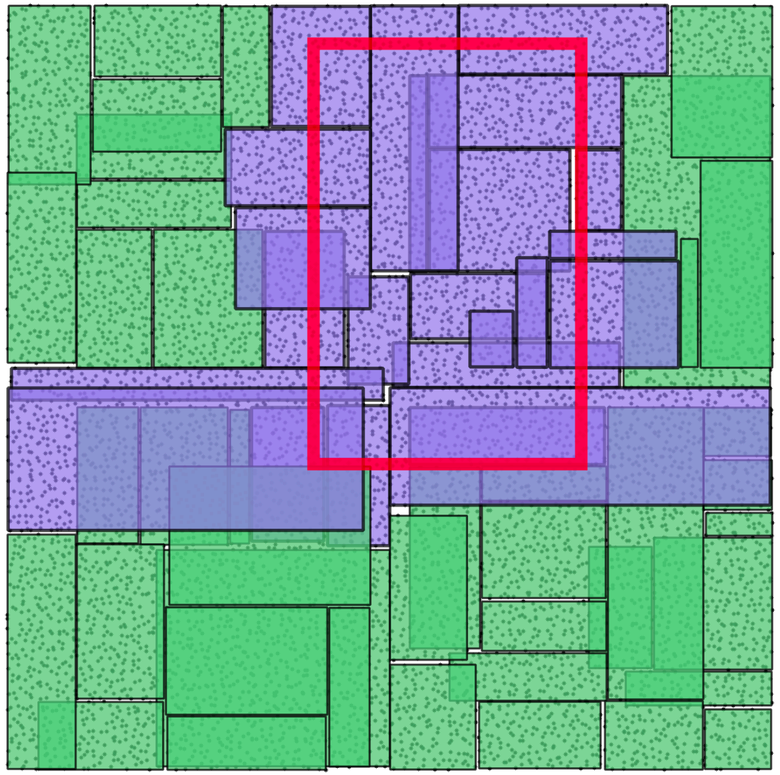


Level 3

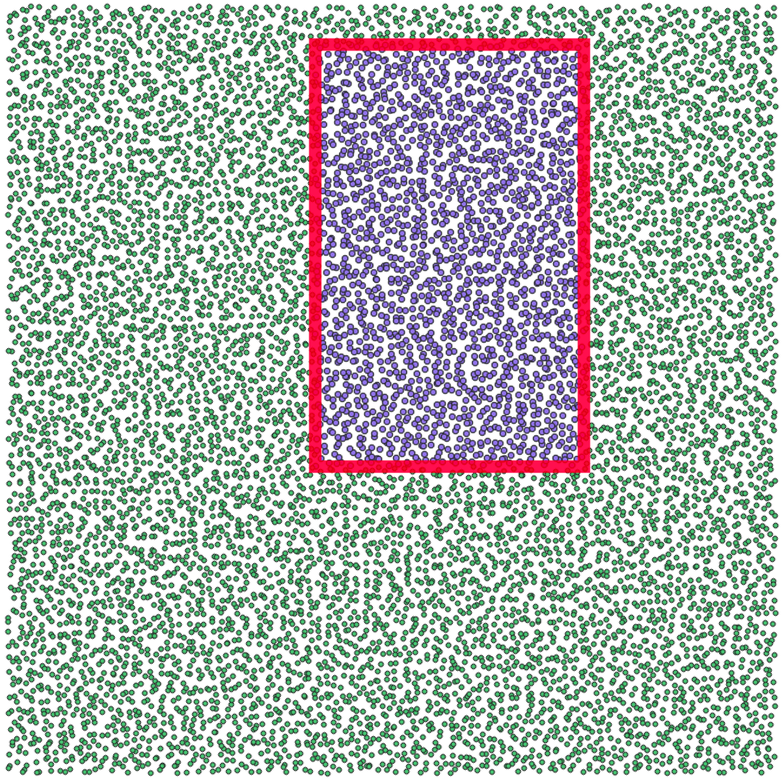
KONTUR

Как происходит сканирование GiST индекса

Стратегия: overlap (&&)

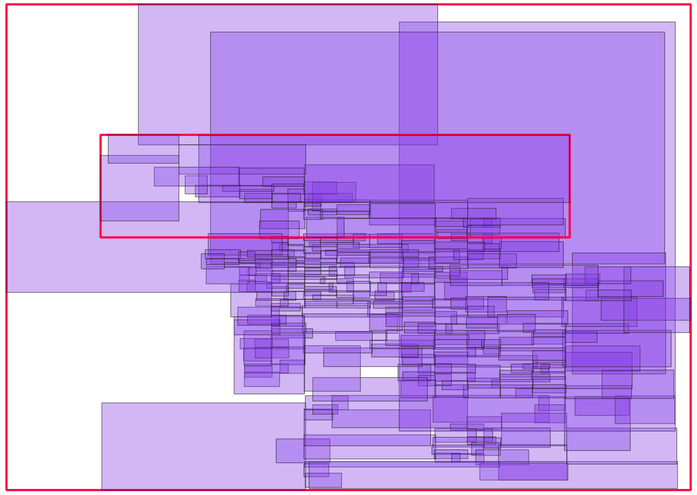


Level 1

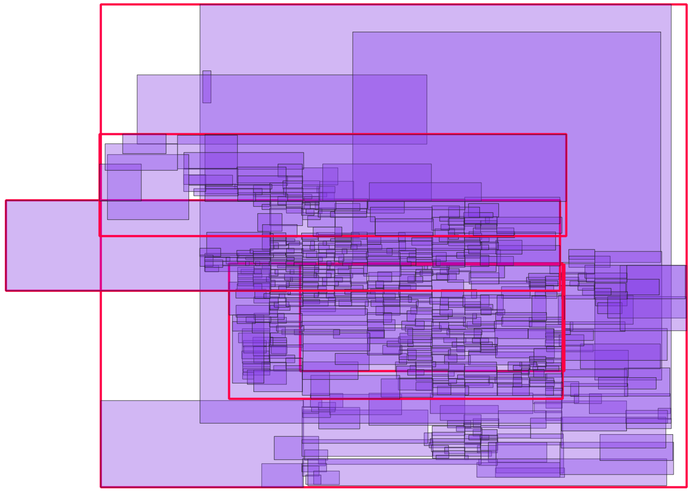


Level 2

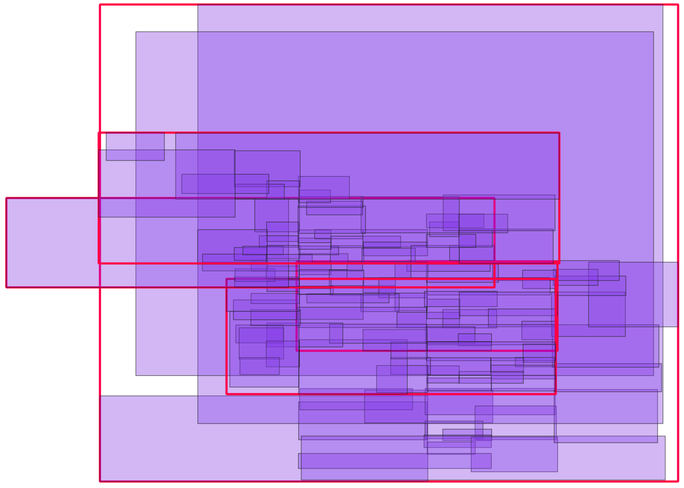
Зависимость между скоростью выполнения запросов и степенью заполнения (fillfactor) страниц индекса



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

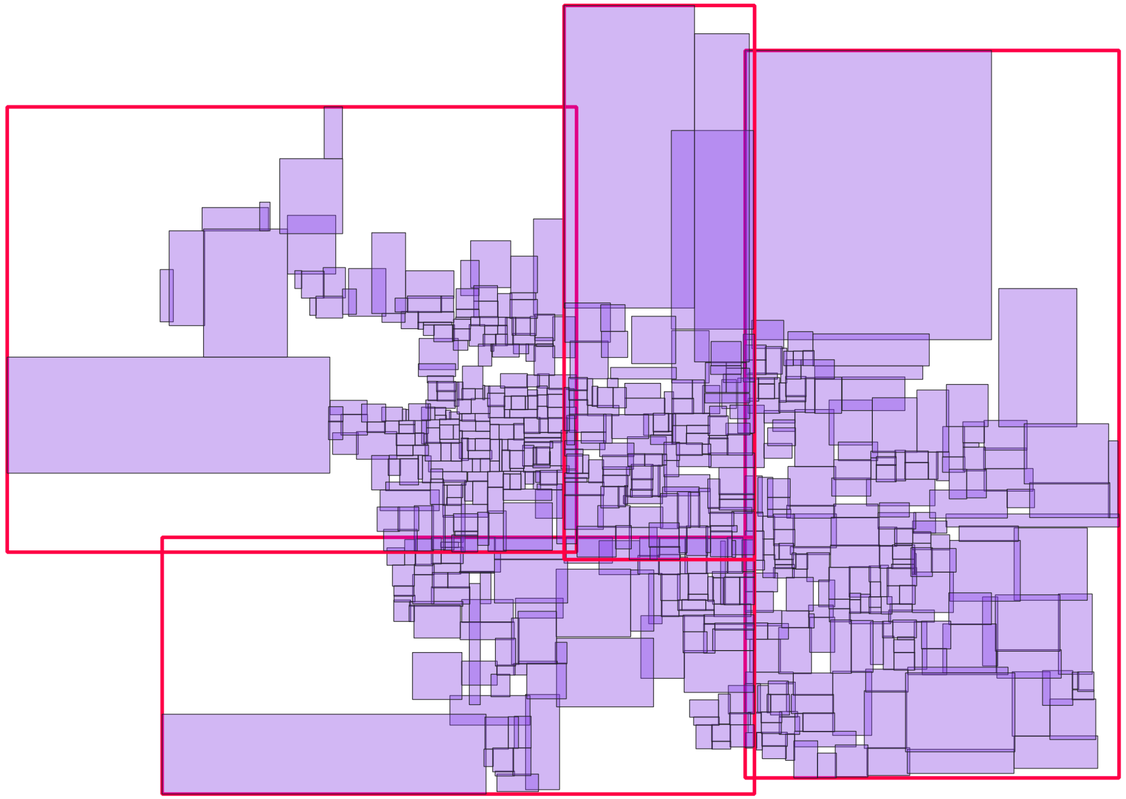
KONTUR

Рассмотренные варианты решения

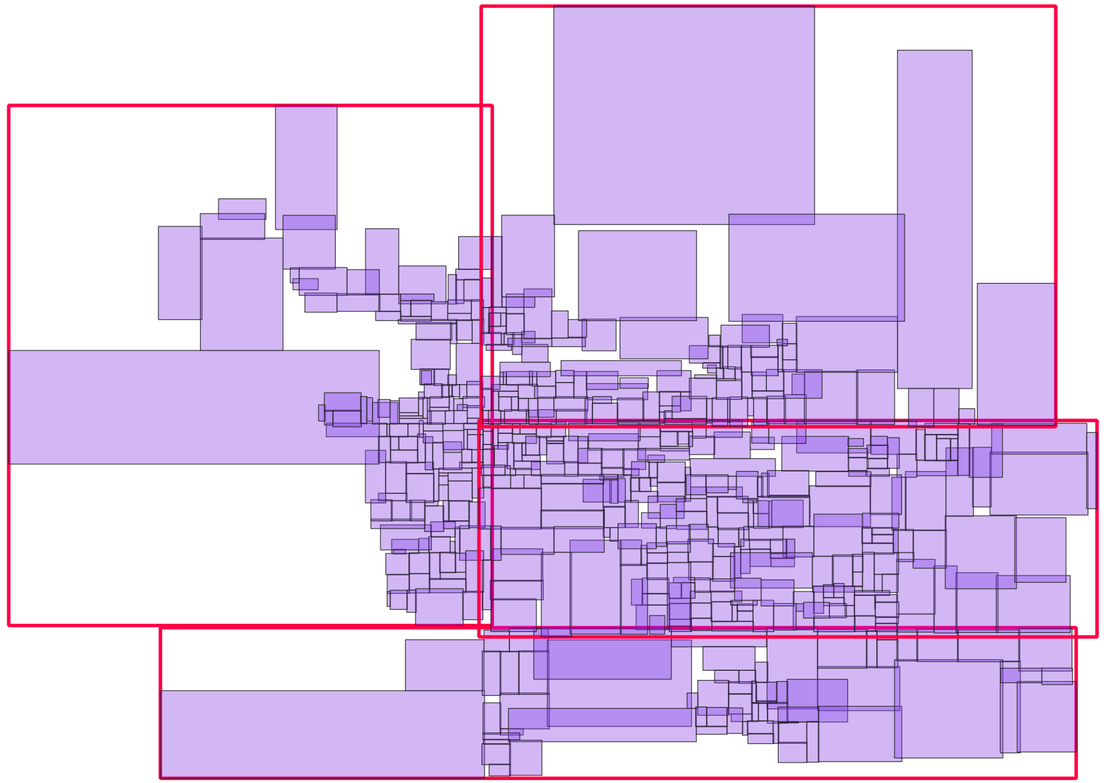
- Искать удачное место для разделения страницы индекса заполняя до fillfactor / 2 и дальше определить начинать ли новую страницу проверяя пенальти.

- 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/

KONTUR

Бенчмарки

	SORT SUPPORT (PG14)	SORT SUPPORT (PG15)	NO SORT SUPPORT
CREATE INDEX	51.2429	165.6421	273.2494
SELECT, SELF-JOIN	4100.5645	1752.5885	1709.3702
SELECT, TILING	458.5242	418.7391	424.466
SELECT, KNN (K = 1)	401.8068	242.7706	248.6685
SIZE OF INDEX (MB)	2.940928	4.95616	5.496832

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.

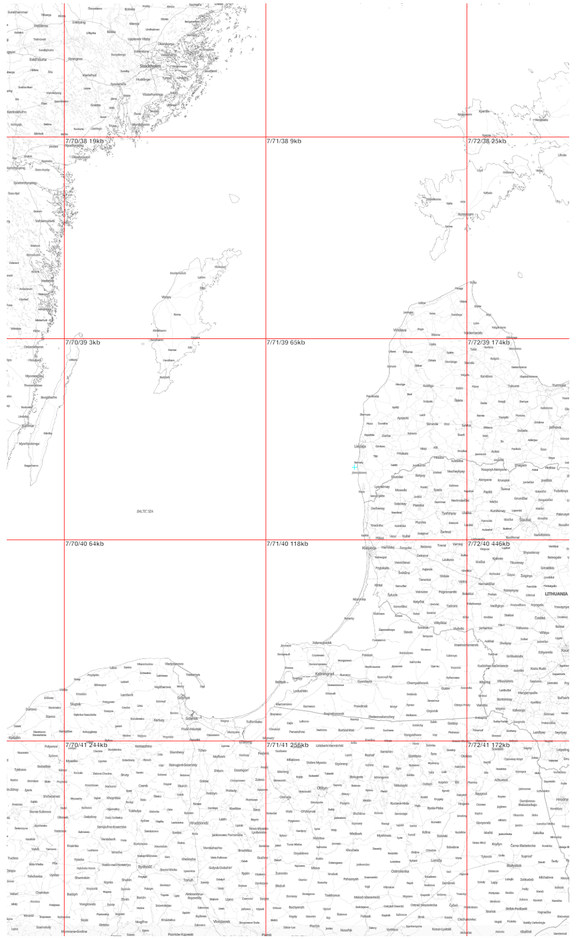
What’s new in PostGIS 3.2

KONTUR

Performance improvements for MVT generation

ST_AsMVT, ST_AsMVTGeom

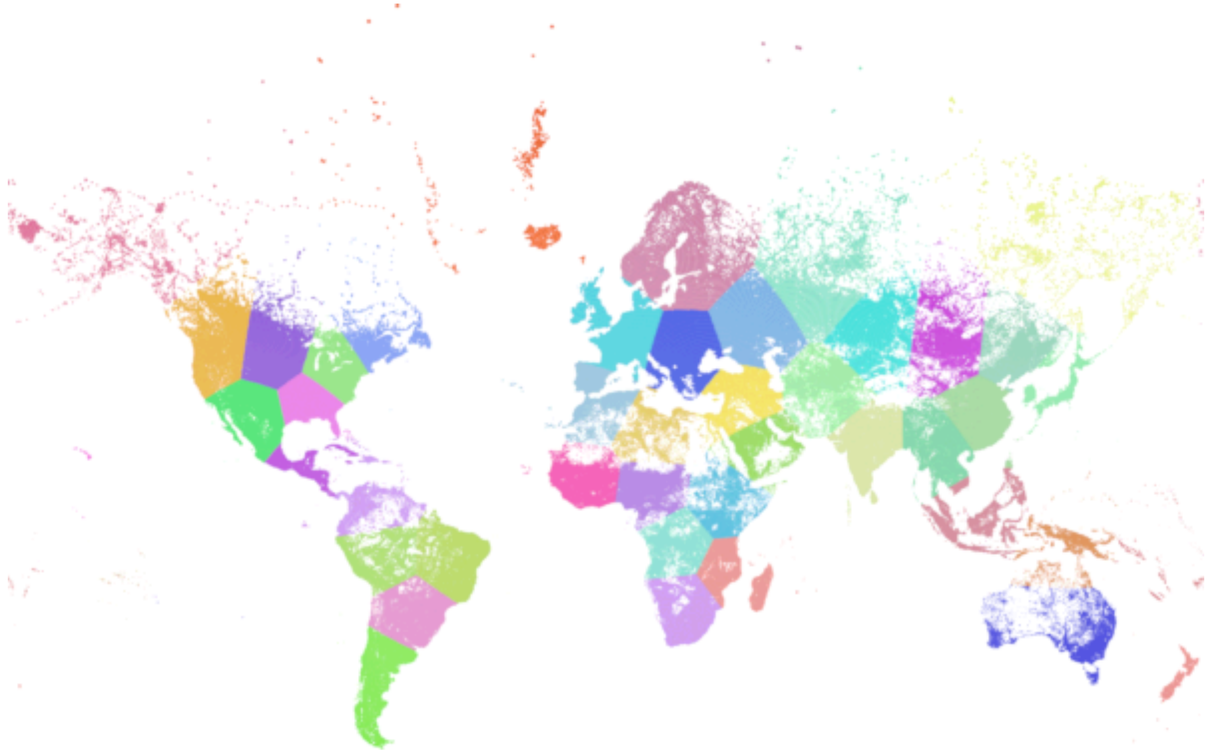
- wagyu’s quick_lr_clip is replaced with lwgeom_clip_to_ordinate_range. 10-20% faster ST_AsMVTGeom
- ST_RemoveRepeatedPoints is O(NlogN) instead of O(N^2) for MULTIPOINT



KONTUR

ST_ClusterKMeans now supports max_radius argument

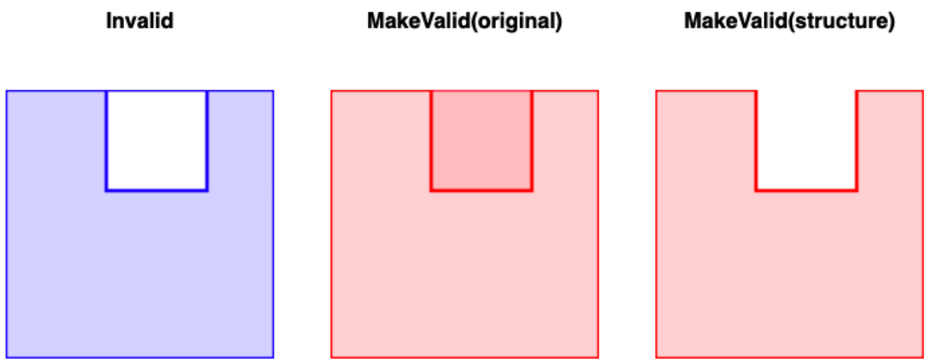
- Use it when you’re not sure what is the number of clusters but you know what the size of clusters should be.

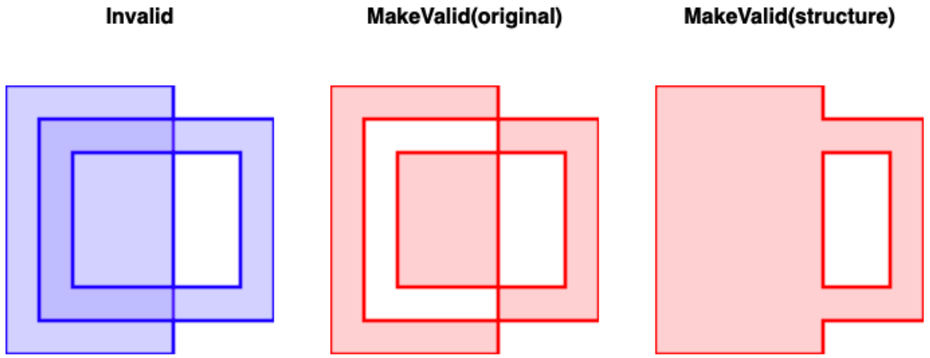


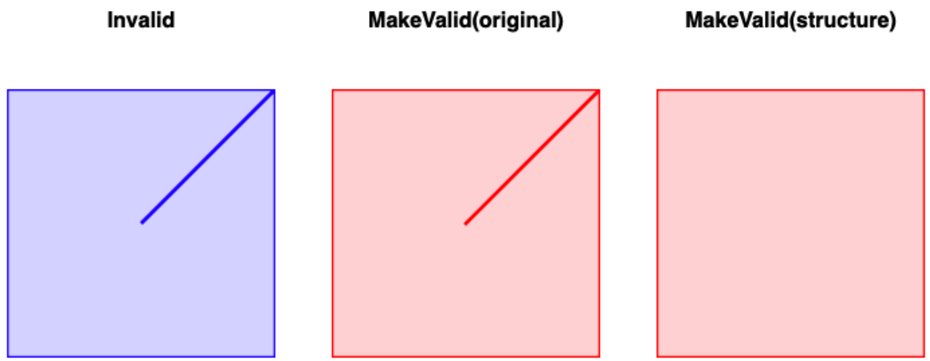
- https://data.humdata.org/dataset/kontur-population-dataset

KONTUR

GEOS 3.10







KONTUR

Raster module

- ST_InterpolateRaster() fills in raster cells between sample points using one of a number of algorithms (inverse weighted distance, average, etc) using algorithms from GDAL
- ST_Contour() generates contour lines from raster values using algorithms from GDAL
- ST_PixelAsCentroids, ST_PixelAsCentroid reimplemented in C