

PostgreSQL для администраторов баз данных
и разработчиков

Использование расширения PostGIS при оптимизации запросов с геопространственными данными



**Меня хорошо видно
& слышно?**



Защита проекта



Снытко Михаил

Бэкенд разработчик
ПСБ, отдел Инвестиций

https://github.com/fjod/pg_dba_24

План защиты

Цель и задачи проекта

Какие технологии использовались

Что получилось

Выводы

Вопросы и рекомендации

Цель и задачи проекта

Цель проекта: работа с геоданными в PostgreSQL

1. Создать структуру БД, использующую геоданные
2. Сделать два инстанса БД, один с оптимизацией
3. Наполнить базы данных сгенерированными значениями
4. Убедиться что оптимизация работает

Какие технологии использовались

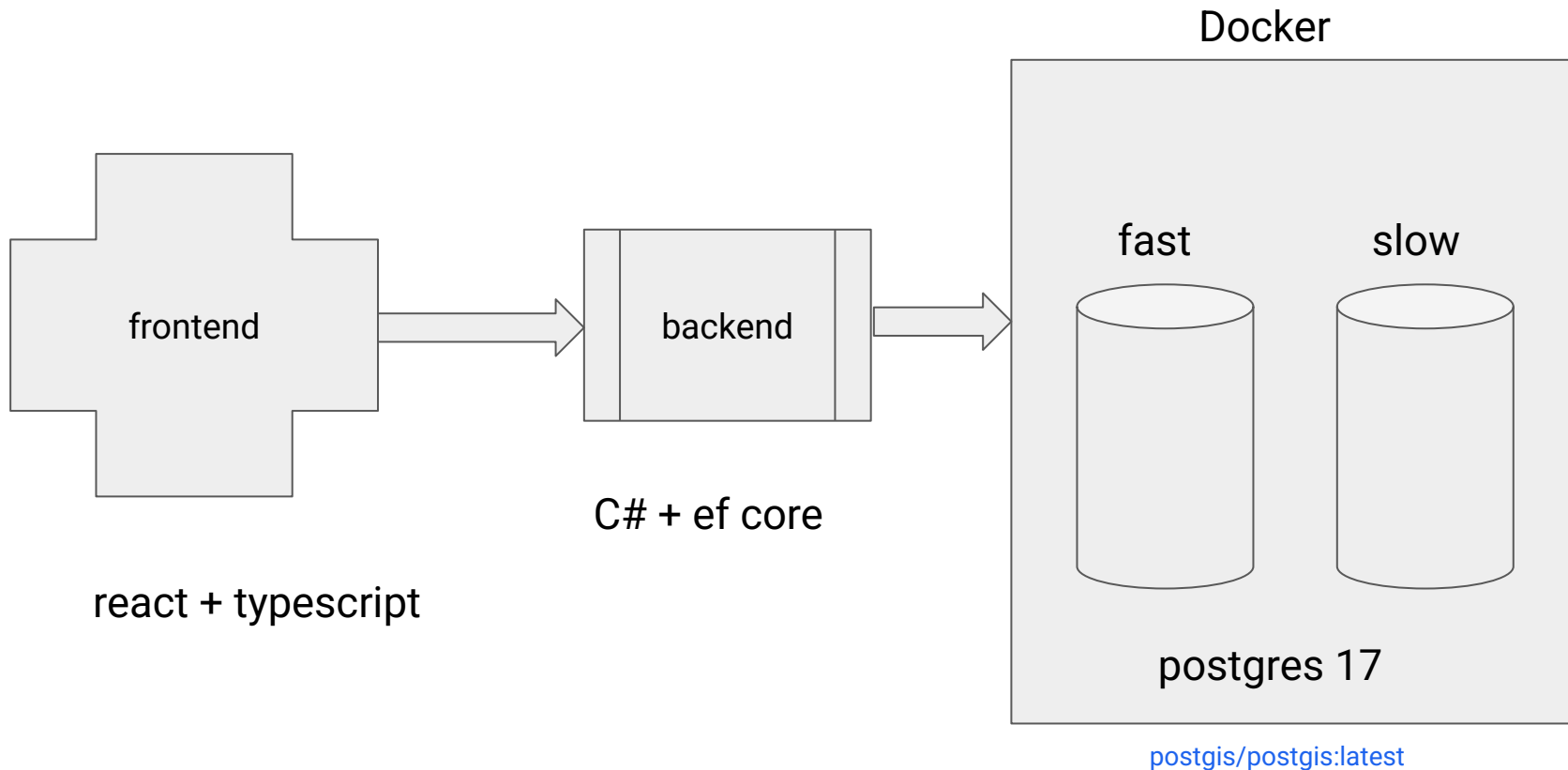
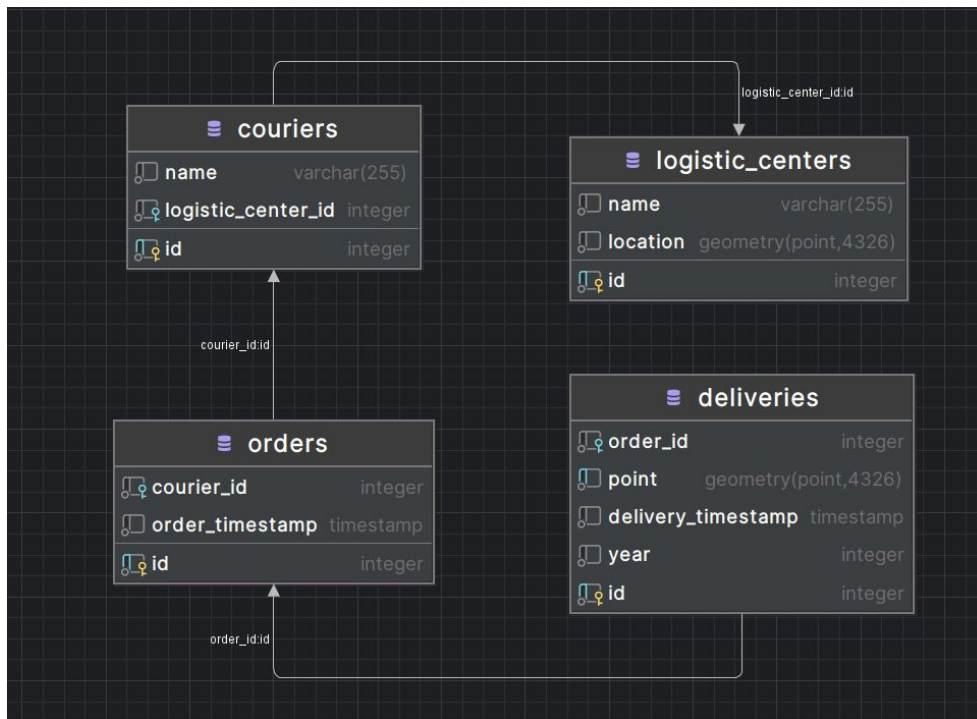


Схема БД



Запрос точек в полигоне

```
▼ SELECT ST_AsText(point), order_id, delivery_timestamp
FROM deliveries
WHERE ST_Intersects(
    ..... geog1 point,
    ..... geog2 ST_MakeEnvelope(32.085065712094298, 54.87186750531864, 32.238874305844298, 54.895072980740217, 4326)
    ..... )
ORDER BY delivery_timestamp
LIMIT 100;
```



План запроса точек в прямоугольнике (с индексом)

Session Result 95 Tx

QUERY PLAN

```
1 Limit (cost=20647022.44..20647086.19 rows=100 width=44)
2   → Result (cost=20647022.44..21672876.60 rows=1609183 width=44)
3     → Sort (cost=20647022.44..20651045.39 rows=1609183 width=44)
4       Sort Key: delivery_timestamp
5         → Index Scan using idx_deliveries_geom on deliveries (cost=0.42..2
6           Index Cond: (point && '0103000020E610000001000000050000005DBAE9
7           Filter: st_intersects(point, '0103000020E61000000100000005000000
```

План запроса точек в прямоугольнике (без индекса)

The screenshot displays a database query plan for a session. The plan is titled "QUERY PLAN" and shows the following steps:

- 1 Limit (cost=51975155.40 .. 51975229.56 rows=100 width=44)
- 2 → Gather Merge (cost=51975155.40 .. 53282318.69 rows=1762448 width=44)
- 3 Workers Planned: 2
- 4 → Result (cost=51974155.37 .. 52535935.67 rows=881224 width=44)
- 5 → Sort (cost=51974155.37 .. 51976358.43 rows=881224 width=44)
- 6 Sort Key: delivery_timestamp
- 7 → Parallel Seq Scan on deliveries (cost=0.00 .. 51940475.62 rows=)
- 8 Filter: st_intersects(point, '0103000020E61000000100000000500

A large white arrow points to the "Parallel Seq Scan on deliveries" step (line 7), indicating the primary operation in the plan.

Запрос точек рядом с логистическим центром для каждого заказа

```
SELECT b.point, b.order_id
FROM (
    ....WITH ranked_deliveries AS (
    ....SELECT
    ....d.id,
    ....d.order_id,
    ....d.point,
    ....d.delivery_timestamp,
    ....ROW_NUMBER() OVER (PARTITION BY d.order_id ORDER BY d.delivery_timestamp) AS rn
    ....FROM deliveries d
    ....JOIN orders o 1..n<->1: ON o.id = d.order_id
    ....JOIN couriers c 1..n<->1: ON c.id = o.courier_id
    ....JOIN logistic_centers lc 1..n<->1: ON lc.id = c.logistic_center_id
    ....WHERE lc.id = 1
    ....AND ST_DWithin(geog1 lc.location, geog2 d.point, tolerance 0.1)
    ....AND d.delivery_timestamp ≥ '2024-09-16 21:00:00.000000'
    ....AND d.delivery_timestamp < '2024-09-17 21:00:00.000000'
    ....)
    ....SELECT
    ....ST_AsText(point) AS point,
    ....order_id
    ....FROM ranked_deliveries
    ....WHERE rn ≤ 10
    ....ORDER BY order_id, delivery_timestamp
    ....) AS b
```



Планы запроса точек рядом с логистическим центром (с индексом)

	QUERY PLAN
1	Subquery Scan on b (cost=16155.06..16155.73 rows=1 width=36)
2	→ Subquery Scan on ranked_deliveries (cost=16155.06..16155.72 rows=1 width=44)
3	→ WindowAgg (cost=16155.06..16155.09 rows=1 width=56)
4	Run Condition: (row_number() OVER (?) ≤ 10)
5	→ Sort (cost=16155.06..16155.07 rows=1 width=44)
6	Sort Key: d.order_id, d.delivery_timestamp
7	→ Nested Loop (cost=32.71..16155.05 rows=1 width=44)
8	→ Nested Loop (cost=32.44..16153.28 rows=6 width=52)
9	→ Nested Loop (cost=32.14..16148.16 rows=6 width=48)
10	→ Index Scan using logistic_centers_pkey on logistic_centers lc (cost=0.14..8.16 rows=1 width=36)
11	Index Cond: (id = 1)
12	→ Bitmap Heap Scan on deliveries d (cost=32.00..16139.99 rows=1 width=44)
13	Filter: ((delivery_timestamp ≥ '2024-09-16 21:00:00'::timestamp without time zone) AND (delivery_timestamp < '2024-09-16 21:00:00'::timestamp without time zone))
14	→ Bitmap Index Scan on idx_deliveries_geom (cost=0.00..32.00 rows=994 width=0)
15	Index Cond: (point && st_expand(lc.location, '0.1'::double precision))
16	→ Index Scan using orders_pkey on orders o (cost=0.29..0.85 rows=1 width=8)
17	Index Cond: (id = d.order_id)
18	→ Index Scan using couriers_pkey on couriers c (cost=0.28..0.30 rows=1 width=8)
19	Index Cond: (id = o.courier_id)
20	Filter: (logistic_center_id = 1)

Планы запроса точек рядом с логистическим центром (без индекса)

1	Subquery Scan on b (cost=169956.29..171497.17 rows=3 width=36)
2	→ Subquery Scan on ranked_deliveries (cost=169956.29..171497.14 rows=3 width=44)
3	→ WindowAgg (cost=169956.29..171495.24 rows=3 width=56)
4	Run Condition: (row_number() OVER (?) ≤ 10)
5	→ Gather Merge (cost=169956.29..171495.19 rows=3 width=44)
6	Workers Planned: 2
7	→ Incremental Sort (cost=168956.27..170494.82 rows=2 width=44)
8	Sort Key: d.order_id, d.delivery_timestamp
9	Presorted Key: d.order_id
10	→ Merge Join (cost=167417.79..170494.73 rows=1 width=44)
11	Merge Cond: (d.order_id = o.id)
12	Join Filter: st_dwithin(lc.location, d.point, '0.1'::double precision)
13	→ Sort (cost=164818.88..164824.89 rows=2405 width=44)
14	Sort Key: d.order_id
15	→ Parallel Seq Scan on deliveries d (cost=0.00..164683.81 rows=2405 width=44)
16	Filter: ((delivery_timestamp ≥ '2024-09-16 21:00:00'::timestamp without time zone) AND (delivery_timestamp < '2024-09-17 21:00:00'::timestamp without time zone))
17	→ Sort (cost=2598.92..2623.92 rows=10000 width=36)
18	Sort Key: o.id
19	→ Nested Loop (cost=21.90..1934.53 rows=10000 width=36)
20	→ Index Scan using logistic_centers_pkey on logistic_centers lc (cost=0.15..8.17 rows=1 width=36)
21	Index Cond: (id = 1)
22	→ Hash Join (cost=21.75..1826.36 rows=10000 width=8)
23	Hash Cond: (o.courier_id = c.id)
24	→ Seq Scan on orders o (cost=0.00..1541.00 rows=100000 width=8)
25	→ Hash (cost=20.50..20.50 rows=100 width=8)
26	→ Seq Scan on couriers c (cost=0.00..20.50 rows=100 width=8)
27	Filter: (logistic_center_id = 1)
28	TTT

Размер индексов 🤔

```
-- размер таблицы  
✓ SELECT pg_size_pretty(pg_total_relation_size('deliveries')) AS table_size,  
        pg_size_pretty(pg_total_relation_size('idx_deliveries_geom')) AS index_size;
```

	table_size 📄	index_size 📄
1	1408 MB	392 MB

Подводные камни:

Значение параметра tolerance в запросе влияет на использование индекса


```
SELECT ST_AsText(point) coordinates, order_id orderId
FROM deliveries
WHERE ST_DWithin(
    ..... geog1 'SRID=4326;POINT(32.0401 54.7818)', geog2 point, tolerance 0.032
    ..... )
order by delivery_timestamp;
```



Подводные камни:

Наполнение базы тестовыми данными (используйте sql, а не бэк):

```
42  --Generate deliveries
43  DO $$
44  ....DECLARE
45  ....center_location geography;
46  ....rec RECORD;
47  ....BEGIN
48  ....FOR rec IN (SELECT id, courier_id, order_timestamp FROM orders) LOOP
49  ....SELECT location INTO center_location FROM logistic_centers lc
50  ....JOIN couriers c ON lc.id = c.logistic_center_id
51  ....WHERE c.id = rec.courier_id;
52
53  ....FOR i IN 1..(50 + floor(random() * 100)) LOOP
54  ....INSERT INTO deliveries (order_id, point, delivery_timestamp, year)
55  ....VALUES (
56  ....order_id rec.id,
57  ....point ST_SetSRID( geog ST_MakePoint(
58  ....ST_X(center_location::geometry) + (i * 0.001) + (random() * 0.001),
59  ....ST_Y(center_location::geometry) + (i * 0.001) + (random() * 0.001)
60  ....), sr_id 4326),
61  ....delivery_timestamp rec.order_timestamp + (i * interval '30 seconds'),
62  ....year EXTRACT(YEAR FROM rec.order_timestamp)
63  ....);
64  ....END LOOP;
65  ....END LOOP;
66  ....END $$;
```



Выводы



1. Индексы для геоданных работают



2. Копилот и другие AI инструменты очень помогают



3. ..но иногда могут и навредить.

Вопросы и рекомендации



если есть вопросы



если вопросов нет

Спасибо за внимание!

