1) ускорить простой запрос, добиться времени выполнения < 10ms

EXPLAIN ANALYZE select name from t1 where id = 50000;

QUERY PLAN

Seq Scan on t1 (cost=0.00..208396.61 rows=1 width=30) (actual time=20.014..626.076 rows=1 loops=1)

Filter: (id = 50000)

Rows Removed by Filter: 9999999

Planning Time: 0.113 ms

JIT:

Functions: 4

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 0.583 ms, Inlining 0.000 ms, Optimization 0.508 ms, Emission 7.256 ms, Total 8.347 ms

Execution Time: 626.769 ms

Таблица содержит t1 содержит 10 миллионов строк, и мы ищем строку по id = 50000, а PostgreSQL читает всю таблицу подряд. потому что нет индекса на id

CREATE INDEX idx_t1_id ON t1(id);

EXPLAIN ANALYZE select name from t1 where id = 50000;

QUERY PLAN

Index Scan using idx_t1_id on t1 (cost=0.43..8.45 rows=1 width=30) (actual time=1.622..1.626 rows=1 loops=1)

Index Cond: (id = 50000)

Planning Time: 0.884 ms

Execution Time: 1.670 ms

И если обновить статистику, то получим лучшие результаты

ANALYZE t1;

EXPLAIN ANALYZE select name from t1 where id = 50000;

QUERY PLAN

Index Scan using idx_t1_id on t1 (cost=0.43..8.45 rows=1 width=30) (actual time=0.027..0.029 rows=1 loops=1)

Index Cond: (id = 50000)

Planning Time: 0.386 ms

Execution Time: 0.064 ms

2) ускорить запрос "max + left join", добиться времени выполнения < 10ms

EXPLAIN ANALYZE select max(t2.day) from t2 left join t1 on t2.t_id = t1.id and t1.name like 'a%';

QUERY PLAN

Aggregate (cost=385687.58..385687.59 rows=1 width=32) (actual time=3172.917..3172.920 rows=1 loops=1)

-> Hash Left Join (cost=218335.76..373187.36 rows=5000090 width=9) (actual time=797.647..2499.138 rows=5000000 loops=1)

Hash Cond: (t2.t_id = t1.id)

-> Seq Scan on t2 (cost=0.00..81872.90 rows=5000090 width=13) (actual time=23.368..439.048 rows=5000000 loops=1)

-> Hash (cost=208392.00..208392.00 rows=606061 width=4) (actual time=773.677..773.678 rows=625914 loops=1)

Buckets: 262144 Batches: 4 Memory Usage: 7573kB

-> Seq Scan on t1 (cost=0.00..208392.00 rows=606061 width=4) (actual time=0.125..690.283 rows=625914 loops=1)

Filter: (name ~~ 'a%'::text)

Rows Removed by Filter: 9374086

Planning Time: 0.459 ms

JIT:

Functions: 13

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 1.369 ms, Inlining 0.000 ms, Optimization 2.021 ms, Emission 21.193 ms, Total 24.583 ms

Execution Time: 3174.437 ms

Создал индексы на t_id в t2 и name в t1, но ощутимого результата это не принесло

CREATE INDEX idx_t2_t_id ON t2(t_id);

CREATE INDEX idx_t1_name ON t1(name);

Так как имеется условие t1.name LIKE 'a%', то LEFT JOIN можно заменить INNER JOIN

EXPLAIN ANALYZE SELECT MAX(t2.day) FROM t2 JOIN t1 ON t2.t_id = t1.id WHERE t1.name LIKE 'a%';

QUERY PLAN

Aggregate (cost=348264.75..348264.76 rows=1 width=32) (actual time=2199.551..2199.554 rows=1 loops=1)

-> Hash Join (cost=192656.88..347507.18 rows=303030 width=9) (actual time=743.409..2151.255 rows=313147 loops=1)

Hash Cond: (t2.t_id = t1.id)

-> Seq Scan on t2 (cost=0.00..81872.00 rows=5000000 width=13) (actual time=17.440..405.823 rows=5000000 loops=1)

-> Hash (cost=182713.12..182713.12 rows=606061 width=4) (actual time=725.316..725.317 rows=625914 loops=1)

Buckets: 262144 Batches: 4 Memory Usage: 7573kB

-> Bitmap Heap Scan on t1 (cost=20520.67..182713.12 rows=606061 width=4) (actual time=71.125..640.099 rows=625914 loops=1)

Filter: (name ~~ 'a%'::text)

Rows Removed by Filter: 3718371

Heap Blocks: exact=50119 lossy=33177

-> Bitmap Index Scan on idx_t1_name_like (cost=0.00..20369.15 rows=594059 width=0) (actual time=64.695..64.695 rows=625914 loops=1)

Index Cond: ((name ~>=~ 'a'::text) AND (name ~<~ 'b'::text))

Planning Time: 1.184 ms

JIT:

Functions: 13

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 1.405 ms, Inlining 0.000 ms, Optimization 0.752 ms, Emission 16.724 ms, Total 18.881 ms

Execution Time: 2201.124 ms

Настройки PsotgreSQL были по умолчанию, поэтому решил увеличить work_mem до 400 Mb, снизил random_page_cost до 1.1, увеличил max_parallel_workers и parallel_setup_cost в 3 раза

EXPLAIN ANALYZE SELECT MAX(t2.day) FROM t2 JOIN t1 ON t2.t_id = t1.id WHERE t1.name LIKE 'a%';

QUERY PLAN

Aggregate (cost=219273.04..219273.05 rows=1 width=32) (actual time=1938.820..1938.824 rows=1 loops=1)

-> Hash Join (cost=114255.97..218389.21 rows=353535 width=9) (actual time=730.962..1893.646 rows=313147 loops=1)

Hash Cond: (t2.t_id = t1.id)

-> Seq Scan on t2 (cost=0.00..81847.92 rows=4999992 width=13) (actual time=0.029..326.765 rows=5000000 loops=1)

-> Hash (cost=105417.52..105417.52 rows=707076 width=4) (actual time=729.507..729.509 rows=625914 loops=1)

Buckets: 1048576 Batches: 1 Memory Usage: 30197kB

-> Bitmap Heap Scan on t1 (cost=13420.08..105417.52 rows=707076 width=4) (actual time=104.655..587.783 rows=625914 loops=1)

Filter: (name ~~ 'a%'::text)

Heap Blocks: exact=83296

-> Bitmap Index Scan on idx_t1_name_like (cost=0.00..13243.31 rows=693075 width=0) (actual time=87.721..87.721 rows=625914 loops=1)

Index Cond: ((name ~>=~ 'a'::text) AND (name ~<~ 'b'::text))

Planning Time: 3.909 ms

JIT:

Functions: 14

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 0.592 ms, Inlining 0.000 ms, Optimization 0.560 ms, Emission 5.262 ms, Total 6.415 ms

Execution Time: 1958.242 ms

Переписал запрос с использованием фильтрации EXIST, но большого результата это не дало

EXPLAIN ANALYZE SELECT MAX(t2.day) FROM t2 WHERE EXISTS (SELECT 1 FROM t1 WHERE t1.id = 2.t_id AND t1.name LIKE 'a%');

QUERY PLAN

Aggregate (cost=208940.05..208940.06 rows=1 width=32) (actual time=1739.567..1739.571 rows=1 loops=1)

-> Hash Semi Join (cost=109838.37..208182.48 rows=303030 width=9) (actual time=573.996..1693.283 rows=313147 loops=1)

Hash Cond: (t2.t_id = t1.id)

-> Seq Scan on t2 (cost=0.00..81847.92 rows=4999992 width=13) (actual time=0.045..354.127 rows=5000000 loops=1)

-> Hash (cost=102262.72..102262.72 rows=606052 width=4) (actual time=572.875..572.877 rows=625914 loops=1)

Buckets: 1048576 Batches: 1 Memory Usage: 30197kB

-> Bitmap Heap Scan on t1 (cost=11503.08..102262.72 rows=606052 width=4) (actual time=76.671..457.391 rows=625914 loops=1)

Filter: (name ~~ 'a%'::text)

Heap Blocks: exact=83296

-> Bitmap Index Scan on idx_t1_name_like (cost=0.00..11351.57 rows=594051 width=0) (actual time=61.572..61.572 rows=625914 loops=1)

Index Cond: ((name ~>=~ 'a'::text) AND (name ~<~ 'b'::text))

Planning Time: 0.993 ms

JIT:

Functions: 14

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 2.481 ms, Inlining 0.000 ms, Optimization 0.297 ms, Emission 3.864 ms, Total 6.642 ms

Execution Time: 1742.282 ms

3) ускорить запрос "anti-join", добиться времени выполнения < 10sec

После проделанных ранее манипуляций, время выполнения запроса уже составляло < 10 sec

EXPLAIN ANALYZE select day from t2 where t_id not in (select t1.id from t1);

QUERY PLAN

Seq Scan on t2 (cost=208332.23..302680.12 rows=2499996 width=9) (actual time=6409.593..6409.594 rows=0 loops=1)

Filter: (NOT (hashed SubPlan 1))

Rows Removed by Filter: 5000000

SubPlan 1

-> Seq Scan on t1 (cost=0.00..183332.58 rows=9999858 width=4) (actual time=0.188..897.341 rows=10000000 loops=1)

Planning Time: 1.543 ms

JIT:

Functions: 17

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 1.627 ms, Inlining 0.000 ms, Optimization 1.824 ms, Emission 13.632 ms, Total 17.083 ms

Execution Time: 6415.750 ms

Я пересоздал индексы:

CREATE INDEX idx_t1_id ON t1(id);

CREATE INDEX idx_t2_t_id ON t2(t_id);

EXPLAIN ANALYZE select day from t2 where t_id not in (select t1.id from t1);

QUERY PLAN

Seq Scan on t2 (cost=208334.00..302682.00 rows=2500000 width=9) (actual time=5616.914..5616.915 rows=0 loops=1)

Filter: (NOT (hashed SubPlan 1))

Rows Removed by Filter: 5000000

SubPlan 1

-> Seq Scan on t1 (cost=0.00..183334.00 rows=10000000 width=4) (actual time=0.178..899.932 rows=10000000 loops=1)

Planning Time: 1.272 ms

JIT:

Functions: 17

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 1.481 ms, Inlining 0.000 ms, Optimization 0.320 ms, Emission 6.346 ms, Total 8.146 ms

Execution Time: 5623.263 ms

И получил еще лучший результат

4) ускорить запрос "semi-join", добиться времени выполнения < 10sec

EXPLAIN ANALYZE select day from t2 where t_id in (select t1.id from t1 where t2.t_id = t1.id) and day > to_char(date_trunc('day',now()- '1 months'::interval),'yyyymmdd');

QUERY PLAN

Seq Scan on t2 (cost=0.00..7881848.00 rows=420000 width=9) (actual time=1474.077..13709.700 rows=833505 loops=1)

Filter: ((day > to_char(date_trunc('day'::text, (now() - '1 mon'::interval)), 'yyyymmdd'::text)) AND (SubPlan 1))

Rows Removed by Filter: 4166495

SubPlan 1

-> Index Only Scan using idx_t1_id on t1 (cost=0.43..2.65 rows=1 width=4) (actual time=0.010..0.010 rows=1 loops=833505)

Index Cond: (id = t2.t_id)

Heap Fetches: 833505

Planning Time: 2.307 ms

JIT:

Functions: 10

Options: Inlining true, Optimization true, Expressions true, Deforming true

Timing: Generation 1.452 ms, Inlining 492.206 ms, Optimization 756.720 ms, Emission 224.746 ms, Total 1475.124 ms

Execution Time: 13763.970 ms

Создал индекс на day, для ускорения фильтрации

CREATE INDEX idx_t2_day ON t2(day);

EXPLAIN ANALYZE select day from t2 where t_id in (select t1.id from t1 where t2.t_id = t1.id) and day > to_char(date_trunc('day',now()- '1 months'::interval),'yyyymmdd');

QUERY PLAN

Index Scan using idx_t2_day on t2 (cost=0.44..1320753.31 rows=420000 width=9) (actual time=101.790..6062.928 rows=833505 loops=1)

Index Cond: (day > to_char(date_trunc('day'::text, (now() - '1 mon'::interval)), 'yyyymmdd'::text))

Filter: (SubPlan 1)

SubPlan 1

-> Index Only Scan using idx_t1_id on t1 (cost=0.43..2.65 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=833505)

Index Cond: (id = t2.t_id)

Heap Fetches: 833505

Planning Time: 0.752 ms

JIT:

Functions: 13

Options: Inlining true, Optimization true, Expressions true, Deforming true

Timing: Generation 2.049 ms, Inlining 35.657 ms, Optimization 38.337 ms, Emission 27.705 ms, Total 103.748 ms

Execution Time: 6108.618 ms

И получил удовлетворяющий условию результат