

Fixing & Detecting Slow Postgres Queries with **pganalyze Query Advisor**

- 1. Re-introducing pganalyze Workbooks**
- 2. What's new with Workbooks?**
- 3. Introducing pganalyze Query Advisor (Demo!)**
- 4. Common Problems detected by Query Advisor**
- 5. Behind the Scenes of Query Advisor**
- 6. Early Access & Looking Ahead**





Re-introducing pganalyze Workbooks

Introducing Query Tuning Workbooks: Safely Tune Postgres Queries on Production with pganalyze

At some point, every engineering team finds itself grappling with the complexity of query optimization. One query may run perfectly well for a particular customer's parameters, yet degrade performance for another's dataset. A small tweak that improves latency in staging might have unforeseen consequences in production. Until now, the standard approach to experimentation—perhaps running `EXPLAIN ANALYZE` in a sandbox environment, copying and pasting results into an editor—hasn't given developers or DBAs the full, real-world picture they need.

Today, **we're excited to introduce the new Query Tuning Workbooks feature in pganalyze**. Currently in beta, Query Tuning Workbooks let you benchmark a query that is slow, experiment with rewrites, planner settings, or planner hints, and compare the resulting `EXPLAIN` plans.

You can either upload `EXPLAIN ANALYZE` results to a workbook, or you can opt-in to running `EXPLAIN ANALYZE` through the pganalyze collector, making it easier to test different query variants on production, benchmark the same query with multiple different input parameters, and share the results with your team.

A dedicated environment for tuning Postgres queries

At a high level, the workflow for tuning Postgres queries can be split into three steps:

- a. Record details of the execution of a particular query with a given set of input parameters using `EXPLAIN (ANALYZE, BUFFERS)`
- b. Form a hypothesis on which parts of the query are slower than necessary, and how to tweak them
- c. Try out a change, such as rewriting the query, or turning of a planner setting to see a different plan



Keiko Oda
Engineering



Maciek Sakrejda
Engineering



Sean Linsley
Engineering



Jens Nikolaus
Design



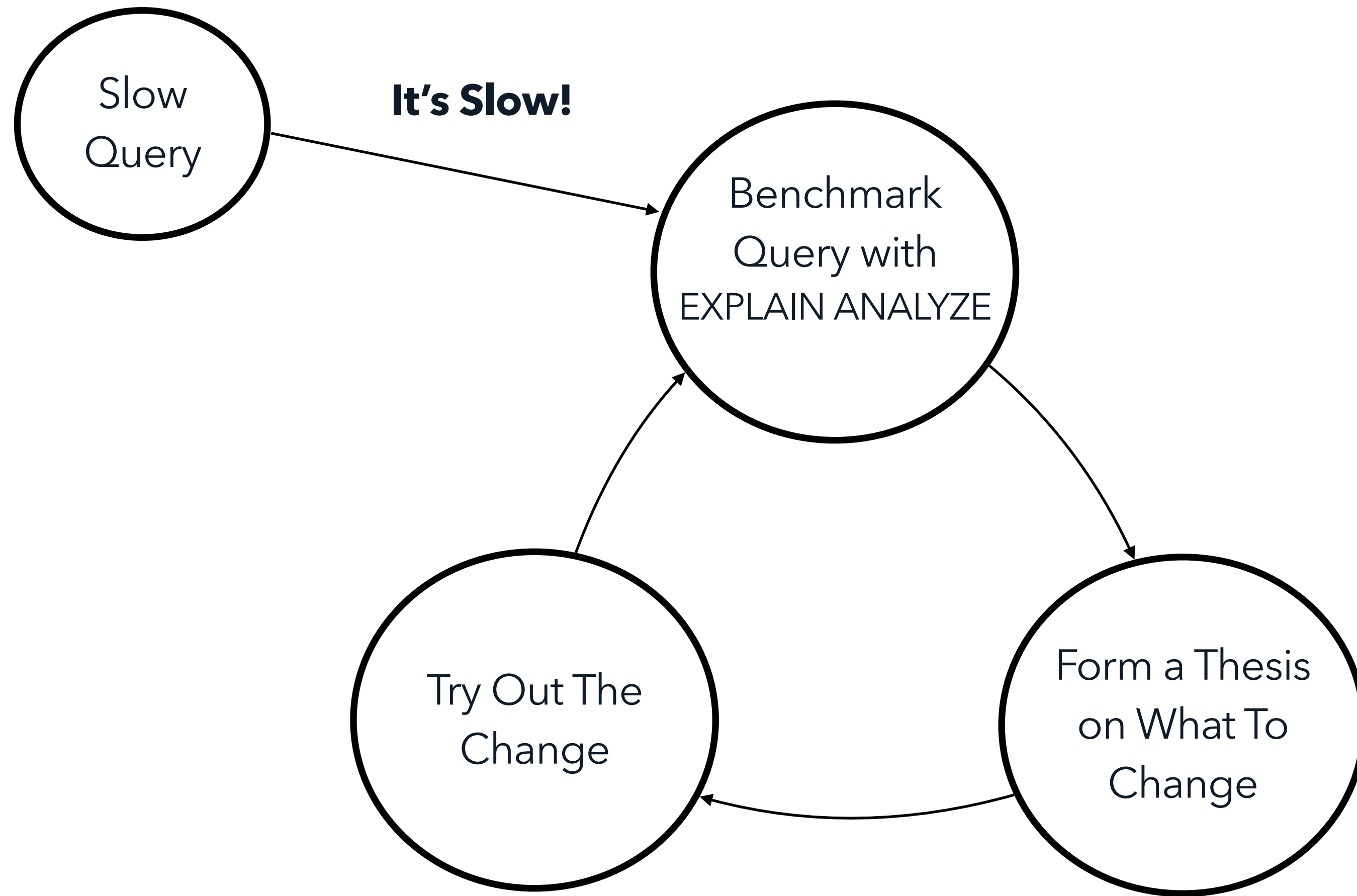
Lukas Fittl
Product

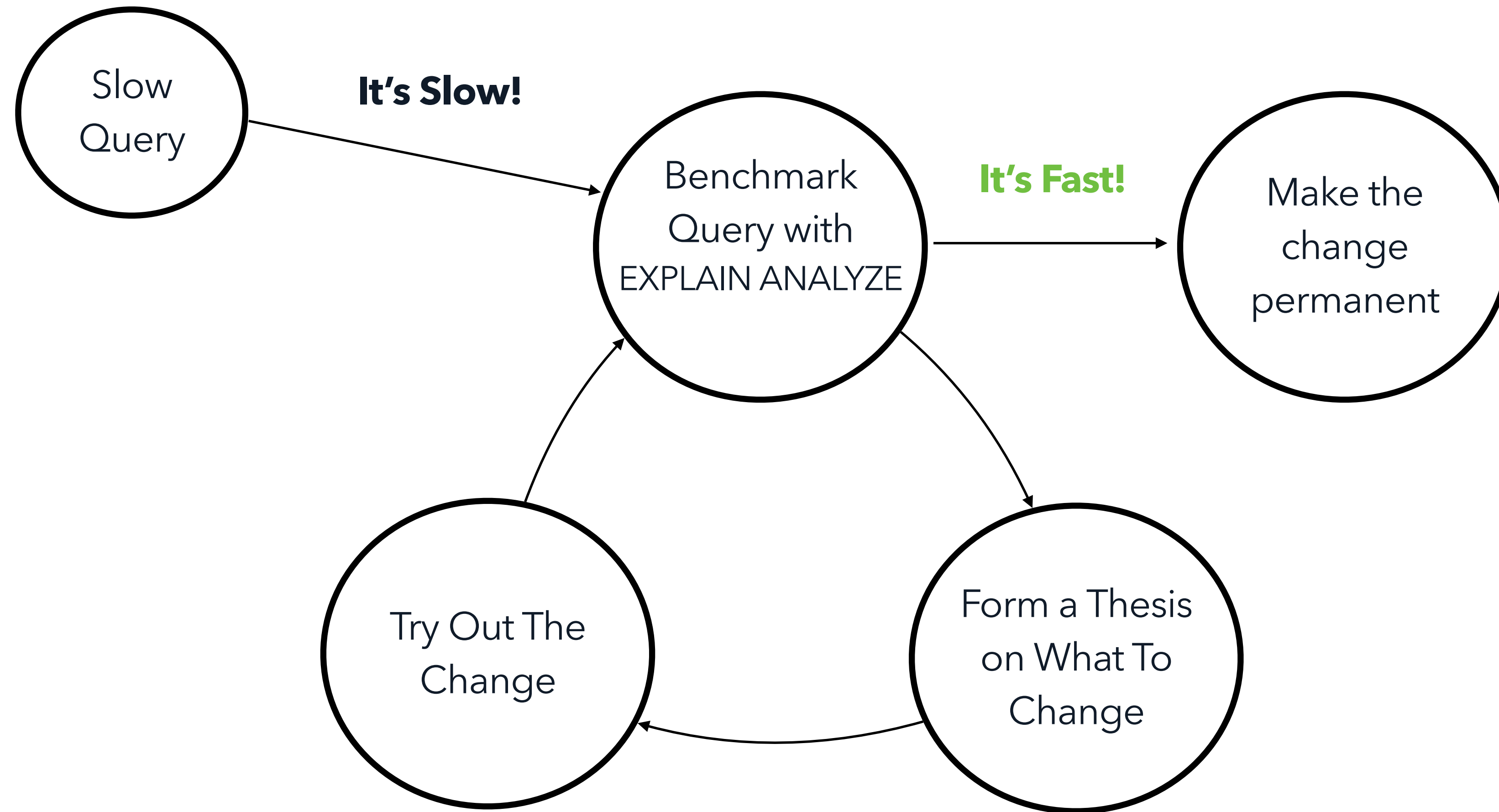
Published on:
January 08, 2025

In this article:

- [A dedicated environment for tuning Postgres queries](#)
- [Measuring the impact of different parameters on query plans](#)
- [Running `EXPLAIN ANALYZE` in a safe and structured manner](#)
- [Viewing plans, `EXPLAIN` insights, and comparing plans](#)
- [Improving performance through query variants](#)







Speed up role filter in query_stats_for_database #4963

Edit <> Code

Merged seanlinsley merged 1 commit into main from query-stats-for-databases-role-filter-speedup on Jun 20

Conversation 1 Commits 1 Checks 7 Files changed 1 +5 -2

seanlinsley commented on Jun 18 · edited

<https://pganalyze.zendesk.com/agent/tickets/6770> reported a query timeout when using the filter on the Query Performance list view. This regression was introduced as part of #4807

Example Ruby code:

```
Dataload::Queries.query_stats_for_database(Database.find(-663636933),
  start_ts: Time.at(1750253727), end_ts: Time.at(1750275327), statement_types: ["read", "write", "other"],
  filter: "test", offset: 0, limit: 100, sort_by: "pctOfTotal", sort_direction: "desc", compare: true
).count
```

The problem is this part of the explain plan, where a nested loop join has to filter out 74 million rows:

```
-> Sort (cost=756.23..756.24 rows=1 width=258) (actual time=49170.982..49171.001 rows=100 loops=1)
  Sort Key: (sum(qd.pct_of_total)) DESC NULLS LAST
  Sort Method: top-N heapsort Memory: 376kB
  Buffers: shared hit=96303
-> Nested Loop (cost=177.69..756.22 rows=1 width=258) (actual time=1260.248..49170.092 rows=513 loops=1)
  Join Filter: ((qd.fingerprint = query_data_early_sort_limit.fingerprint) AND ((q.normalized_query_data_early_sort_limit.fingerprint = qd.fingerprint) OR (q.normalized_query_data_early_sort_limit.fingerprint = qd.fingerprint)))
  Rows Removed by Join Filter: 74097721
  Buffers: shared hit=96300
-> Nested Loop (cost=108.09..677.00 rows=1 width=550) (actual time=1072.749..1157.073 rows=87 loops=1)
  Buffers: shared hit=88734
-> HashAggregate (cost=107.50..112.00 rows=200 width=112) (actual time=1072.693..1083.400 rows=200 loops=1)
  Group Key: qd.fingerprint
  Batches: 1 Memory Usage: 8609kB
  Buffers: shared hit=45019
-> CTE Scan on query_data_early_sort_limit qd (cost=0.00..43.00 rows=2150 width=112) (actual time=0.000..0.000 rows=2150 loops=1)
  Buffers: shared hit=45019
-> Index Scan using index_queries_on_database_id_and_fingerprint on queries q (cost=0.50..42.50 rows=1 width=112) (actual time=0.000..0.000 rows=1 loops=1)
  Index Cond: ((database_id = '-663636933'::integer) AND (fingerprint = ("right"((qd.fingerprint)::text, 16) || '00000000000000000000000000000000')::text))
  Filter: ((statement_types && '{SelectStmt}'::text[]) OR (statement_types && '{InsertStmt}'::text[]))
  Buffers: shared hit=43713
-> GroupAggregate (cost=69.60..73.72 rows=200 width=64) (actual time=0.002..4.844 rows=8477 loops=1)
  Group Key: query_data_early_sort_limit.fingerprint
  Buffers: shared hit=6
```

Reviewers Ifittl ✓

Assignees No one—assign yourself

Labels None yet

Projects None yet

Milestone No milestone

Development Successfully merging this pull request may close these issues. None yet

Notifications Unsubscribe



You're receiving notifications because you're watching this repository.

3 participants



Let's Tune The Query!

WITH total_times AS (...), fingerprints AS (...), raw_query_data AS (...), q


 fingerprint b33bede238bc4de1  role pgaweb_app line /app/services/dataload/queries/query_stats_for_tab... cc

[Overview](#) [Index Advisor ?](#) [Query Samples 5+](#) [EXPLAIN Plans 5+](#) [Query Tags 5+](#) [Log](#)

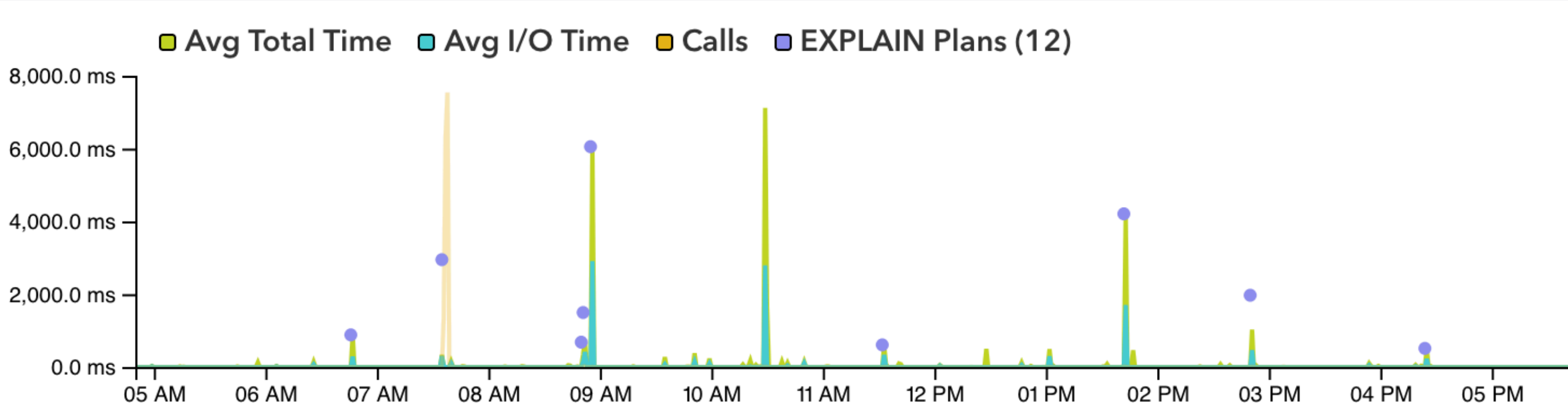
SQL Statement

```
/*controller:graphql,action:graphql,line:/app/services/dataload/queries/query_stats_for_tabl  
d,traceparent:00-c196797a...
```

[Show full query text](#)

 [Tune query in workbook](#)

Avg Time & Calls



Legend: Avg Total Time (green), Avg I/O Time (teal), Calls (orange), EXPLAIN Plans (12) (purple)

Time	Avg Total Time (ms)	Avg I/O Time (ms)	Calls	EXPLAIN Plans (12)
07 AM	~100	~100	~100	~1000
07:45 AM	~3000	~3000	~7500	~3000
09 AM	~1500	~1500	~1500	~6000
11 AM	~7500	~7500	~7500	~7500
11:45 AM	~1000	~1000	~1000	~1000
01:45 PM	~4000	~4000	~4000	~4000
02:45 PM	~1000	~1000	~1000	~2000
04:45 PM	~1000	~1000	~1000	~1000

Automatic Naming of Parameters

1 Review query 2 Choose parameters 3 Run EXPLAIN

The query was reformatted, normalized, and positional parameters (\$1) were turned into named parameters (\$param). Names are auto-assigned and suffixed with consecutive numbers—for example, param , param_2 , and param_3 . Parameters can be merged by renaming to an existing name.

```
/*job:Storage::RecheckMissingIndexWorker,line:/app/services/dataload/queries/query_stats_for_index_advisor.rb:6
5:in `query_stats_for_index_advisor',sentry_trace_id:fb91171cbe64102a85bad6585d7a95c,tracestate:pganalyze=t:1753
833646.5048623*/
WITH total_times AS (
  SELECT
    sum(query_stats_blk_read_time_sum + query_stats_blk_write_time_sum) AS total_iotime,
    sum(query_stats_total_time_sum) AS total_runtime
  FROM query_overview_stats_35d
  WHERE
    database_id = $database_id_2
    AND collected_at BETWEEN $collected_at_3 AND $collected_at_4
), raw_query_data AS (
  (
    (
      SELECT
        fingerprint, sum(calls) AS calls, sum(ROWS) AS ROWS,
        sum(total_time) AS total_time,
        sum(shared_blks_hit) AS shared_blks_hit,
        sum(shared_blks_read) AS shared_blks_read,
        sum(blk_read_time) AS blk_read_time,
        sum(blk_write_time) AS blk_write_time
      FROM unpack_query_stats(database_ids := ARRAY[$param_10], start_ts := $start_ts_2, end_ts := $end_ts_2)
      WHERE
        collected_at >= $collected_at_5
        AND CASE
          WHEN $param_4 >= $param_5 THEN collected_at < $collected_at_13
          ELSE collected_at <= $collected_at_8
        END
      END
```

Choose Parameters

💡 36 parameters detected

Rename Parameters ⓘ

- array_to_string_arg
- database_id
- right_arg
- database_id_2
- collected_at_3
- collected_at_4
- param_10
- start_ts_2
- end_ts_2
- collected_at_5
- param_4
- param_5
- collected_at_13
- collected_at_8
- database_id_5
- collected_at_11

PR 4963 Dataload::Queries.query_stats_for_database optimization

[Overview](#) [Compare Plans](#) [Parameter Sets](#)

+ New Query Variant

All Query Plans

Baseline

Variant 1



Variant 2



Variant 4



Variant 5 - materialize roles



Query Advisor

No insights found

Query

No match with existing queries

Query tags

No query tags

Baseline

With parameter names ▾

```
1 WITH total_times AS (  
2   SELECT  
3     sum(query_stats_blk_read_time_sum + query_stats_blk_write_time_sum) AS total_iotime,  
4     sum(query_stats_total_time_sum) AS total_runtime  
5   FROM query_overview_stats_35d qos  
6   WHERE  
7     qos.database_id = $database_id_2
```

▾ Show all code

Query Plans

PLAN	VARIANT	PARAMETER SET	EST. COST	RUNTIME
<input type="checkbox"/> 9d7d557	Baseline	Parameter Set 1	79,656	50,499.50ms



PR 4963 Dataload::Queries.query_stats_for_database optimization

Variant: Baseline

Parameter Set: Parameter Set 1

[Overview](#) [Compare Plans](#) [Parameter Sets](#)

[Node Tree](#) [Grid](#) [Text](#) [JSON](#)

SQL Statement

```
WITH total_times AS (  
SELECT  
sum(query_stats_blk_read_time_sum + query_stats_blk_write_time_sum) AS total_iotime,  
sum(query_stats_total_time_sum) AS total_runtime  
FROM query_overview_stats...
```

[Show full query text](#)

Show: Est. Cost Runtime Rows Buffers Reads Writes

All metrics exclude children, except Rows. [Learn more](#) about reading EXPLAIN plans.

Plan	Runtime
1 Limit mis-estimate	0.03ms
47 Sort mis-estimate	1.10ms
48 Nested Loop mis-estimate	5,812.04ms
49 Nested Loop inefficient nested loop mis-estimate	16.91ms
50 Aggregate	18.77ms
51 CTE Scan on query_data_early_sort_limit	1,071.16ms
52 Index Scan (Forward) on queries	69.93ms 8.7k
53 Aggregate	37,463.93ms 8.7k
Group Key: query_data_early_sort_limit.fingerprint Partial Mode: Simple Strategy: Sorted	
54 Sort 8.7k	6,034.93ms 8.7k
55 Aggregate	4.32ms
56 CTE Scan on query_data_early_sort_limit	0.78ms
CTE total_times	
2 Aggregate	0.03ms
3 Index Scan (Forward) on query_overview_stats_35d_20250618	0.17ms
CTE raw_query_data	
4 Append	1.84ms
5 Subquery Scan mis-estimate	7.19ms

[Summary](#) [Node Details](#) [Node Source](#)

Aggregate

Performs grouped or ungrouped aggregation on output of its child.

Group Key

query_data_early_sort_limit.fingerprint

Partial Mode

Simple

Strategy

Sorted

I/O & Buffers

Include cumulative costs of node children

	Shared	Local	Temp
Hit	0 B	0 B	-
Read	0 B	0 B	0 B
Dirtied	0 B	0 B	-
Written	0 B	0 B	0 B

I/O Read Time: 0.00ms
I/O Write Time: 0.00ms

Output Columns

- query_data_early_sort_limit.fingerprint
- array_agg(query_data_early_sort_limit.postgres_role_id
ORDER BY (sum(query_data_early_sort_limit.t...



Untitled Variant

[Edit name and description](#)

[Cancel workflow](#)

1 Rewrite Query and Edit Planner 2 Run EXPLAIN


Edit Unified Diff Split Diff

```
1 WITH total_times AS (  
2   SELECT  
3     sum(query_stats_blk_read_time_sum + query_stats_blk_write_time_sum) AS total_iotime,  
4     sum(query_stats_total_time_sum) AS total_runtime  
5   FROM query_overview_stats_35d qos  
6   WHERE  
7     qos.database_id = $database_id_2  
8     AND qos.collected_at BETWEEN $collected_at_5 AND $collected_at_6  
9 ), raw_query_data AS (  
10  (  
11  (  
12  (  
13    SELECT  
14      fingerprint, postgres_role_id, sum(calls) AS calls,  
15      sum(total_time) AS total_time,  
16      sum(shared_blks_hit) AS shared_blks_hit,  
17      sum(shared_blks_read) AS shared_blks_read,  
18      sum(blk_read_time) AS blk_read_time,  
19      sum(blk_write_time) AS blk_write_time  
20    FROM unpack_query_stats(database_ids := ARRAY[$param_30], start_ts := $start_ts_3, end_ts :=  
21 $end_ts_3) qs  
22    WHERE  
23      collected_at >= $collected_at_7  
24      AND CASE  
25        WHEN $param_18 >= $param_19 THEN collected_at < $collected_at_23  
26        ELSE collected_at <= $collected_at_13  
27      END  
28    GROUP BY 1, 2  
29    UNION ALL  
30    SELECT  
31      fingerprint, postgres_role_id, sum(calls) AS calls,
```

Verify Query

Query Advisor

No insights found

 Customize planner behavior by changing the settings and hints. Settings are only applied to the current session.

[Edit Planner Settings](#)

[Edit Planner Hints](#)







No custom planner settings



Use query variants to test hypothesis

All Query Plans

Choose a query variant to see its query text and settings, or create a new variant. Variants can be used to test Postgres planner behavior or rewrite queries to improve performance. [Learn more.](#)

Query Plans					Filter by Parameter Set...
PLAN	VARIANT	PARAMETER SET	EST. COST	RUNTIME	
<input type="checkbox"/>  9d7d557	Baseline	Parameter Set 1	79,656	50,499.50ms	
<input type="checkbox"/>  661d018	Variant 1	Parameter Set 1	79,671	2,868.94ms	
<input type="checkbox"/>  d73907f	Variant 2	Parameter Set 1	78,822	2,844.51ms	
<input type="checkbox"/>  3212a67	Variant 4	Parameter Set 1	78,824	2,128.92ms	
<input type="checkbox"/>  1df7df8	Variant 5 - materialize roles	Parameter Set 1	78,807	 1,423.22ms	

See the impact of plan changes

Plan Comparison		Select plans	
Compare ⓘ: <input type="radio"/> Est. Cost <input checked="" type="radio"/> Runtime <input type="radio"/> I/O Read Time <input type="radio"/> Rows <input type="radio"/> Buffers			
Plan A	Plan B	Plan A	Plan B
Baseline - Parameter Set 1	Variant 5 - materialize roles - Parameter Set 1	Runtime	Runtime
-> Limit	-> Limit	0.03ms	0.03ms
-> Sort	-> Sort	1.10ms	0.40ms
	-> Hash Join		270.94ms
-> Nested Loop	-> Nested Loop	5,812.04ms	1.73ms
-> Nested Loop		16.91ms	
-> Aggregate	-> Aggregate	18.77ms	11.15ms
-> CTE Scan on query_data...	-> CTE Scan on query_data...	1,071.16ms	0.82ms
-> Index Scan ² on queries	-> Index Scan ² on queries	69.93ms	34.96ms
-> Aggregate		37,463.93ms	
-> Sort		6,034.93ms	
-> Aggregate		4.32ms	
-> CTE Scan on query_dat...		0.78ms	
	-> Hash		1.42ms
	-> CTE Scan on roles_sorted		1,096.15ms
CTE total_times			
-> Aggregate	-> Aggregate	0.03ms	0.03ms
-> Index Scan ³ on query_overv...	-> Index Scan ³ on query_overv...	0.17ms	0.15ms
CTE raw_query_data			
-> Append	-> Append	1.84ms	1.80ms
-> Subquery Scan	-> Subquery Scan	2.19ms	2.22ms
-> Aggregate	-> Aggregate	23.28ms	23.64ms
-> Function Scan on unpack_...	-> Function Scan on unpack_...	115.14ms	116.98ms
-> Subquery Scan	-> Subquery Scan	1.14ms	1.15ms
-> Aggregate	-> Aggregate	9.71ms	11.13ms
-> Bitmap Heap Scan on quer...		4.07ms	
-> Bitmap Index Scan ⁵		6.37ms	
	-> Index Scan ⁵ on query_sta...		14.24ms
-> Subquery Scan	-> Subquery Scan	0.00ms	0.00ms
-> Aggregate	-> Aggregate	0.00ms	0.00ms
-> Result	-> Result	0.00ms	0.00ms
-> Subquery Scan	-> Subquery Scan	0.00ms	0.00ms
-> Aggregate	-> Aggregate	0.00ms	0.00ms
-> Result	-> Result	0.00ms	0.00ms

Summary

Node Details

Node Source

Plan A: Baseline - Parameter Set 1

Seen At	Total Est. Cost	Runtime
Jun 18 05:25pm	79,656	50,499.50ms
Plan Fingerprint	Read From Disk	I/O Read Time
9d7d557	0 B	0.00ms

Plan B: Variant 5 - materialize roles - Parameter Set 1

Seen At	Total Est. Cost	Runtime
Jun 20 03:28pm	78,807	1,423.22ms
Plan Fingerprint	Read From Disk	I/O Read Time
1df7df8	0 B	0.00ms

Index usage

A B Index

- ✓✓✓ 1. index_postgres_roles_on_server_id_and_name
- ✓✓✓ 2. index_queries_on_database_id_and_fingerprint
- ✓✓✓ 3. query_overview_stats_35d_20250618_pkey
- ✓✓✓ 4. query_stats_hourlies_35d_20250611_pkey
- ✓✓✓ 5. query_stats_hourlies_35d_20250618_pkey





What's new with Workbooks?

New "Grid" EXPLAIN View

Node Tree **Grid** Text JSON

Show: Est. Cost Runtime Rows Buffers Reads Writes

All metrics exclude children, except Rows. [Learn more](#) about reading EXPLAIN plans.

Plan	Runtime
1 Limit mis-estimate	0.03ms
47 Sort mis-estimate	1.10ms
48 Nested Loop mis-estimate	5,812.04ms
49 Nested Loop inefficient nested loop mis-estimate	16.91ms
50 Aggregate	18.77ms
51 CTE Scan on query_data_early_sort_limit	1,071.16ms
52 Index Scan (Forward) on queries	69.93ms ↻8.7k
53 Aggregate	37,463.93ms ↻8.7k
54 Sort	6,034.93ms ↻8.7k
55 Aggregate	4.32ms
56 CTE Scan on query_data_early_sort_limit	0.78ms
CTE total_times	
2 Aggregate	0.03ms
3 Index Scan (Forward) on query_overview_stats_35d_20250618	0.17ms
CTE raw_query_data	
4 Append	1.84ms
5 Subquery Scan mis-estimate	2.19ms
6 Aggregate mis-estimate	23.28ms
7 f(x) Function Scan on unpack_query_stats(...) mis-estimate	115.14ms
8 Subquery Scan	1.14ms
9 Aggregate	9.71ms



Built-in Diff to compare against Baseline Query Text


Index Unused check

[Overview](#) [Compare Plans](#) [Parameter Sets](#)

[+ New Query Variant](#)

All Query Plans

Baseline

Variant 1 

Query Advisor
No new insights found

Query
No match with existing queries

Query tags
No query tags

Variant description

[Edit variant name and description](#)

[Remove variant](#)


Variant 1 With parameter names ▾

SQL Query **Unified Diff** Split Diff


```
76 unchanged lines
77 WHERE
78     server_id = $server_id_3
79     AND NOT hidden
80     ), ts := $ts, index_ids := ARRAY(
81     SELECT id
82     FROM unused_indexes
83     )) s ON s.schema_index_id = ui.id
84     ), ts := $ts) s ON s.schema_index_id = ui.id
85 WHERE COALESCE(s.size_bytes, $size_bytes) > $param
```

[^ Show less code](#)

Query Plans

PLAN	VARIANT	PARAMETER SET	EST. COST	RUNTIME
<input type="checkbox"/> 5d37d2d	Variant 1	Parameter Set 1	5,256	 2,830.76ms

[Compare selected plans](#) No plans selected

 [Get Help](#)

Built-in SQL formatting

1 Rewrite Query and Edit Planner 2 Run EXPLAIN

Edit Unified Diff Split Diff

```
1 WITH total_times AS (  
2   SELECT  
3     sum(query_stats_blk_read_time_sum + query_stats_blk_write_time_sum) AS total_iotime,  
4     sum(query_stats_total_time_sum) AS total_runtime  
5   FROM query_overview_stats_35d qos  
6   WHERE  
7     qos.database_id = $database_id_2  
8     AND qos.collected_at BETWEEN $collected_at_5 AND $collected_at_6  
9 ), raw_query_data AS (  
10  (  
11  (  
12  (  
13    SELECT  
14      fingerprint, postgres_role_id, sum(calls) AS calls,  
15      sum(total_time) AS total_time,  
16      sum(shared_blks_hit) AS shared_blks_hit,  
17      sum(shared_blks_read) AS shared_blks_read,  
18      sum(blk_read_time) AS blk_read_time,  
19      sum(blk_write_time) AS blk_write_time  
20    FROM unpack_query_stats(database_ids := ARRAY[$param_30], start_ts :=  
21      $start_ts_3, end_ts := $end_ts_3) qs  
22    WHERE
```

Verify Query

Better IN list handling in Parameter Sets

column1 IN (\$1, \$2, \$3, \$4, \$5)

=>

column1 = ANY(\$1)



Planner Settings

Edit Planner Settings



Custom planner settings

enable_incremental_sort ⓘ

off (was on)

enable_memoize ⓘ

off (was on)

Select a planner setting to edit

work_mem



204800

(unit: kB)

Save setting

work_mem (integer)

Sets the base maximum amount of memory to be used by a query operation (such as a sort or hash table) before writing to temporary disk files. If this value is specified without units, it is taken as kilobytes. The default value is four megabytes (4MB). Note that a complex query might perform several sort and hash operations at the same time, with each operation generally being allowed to use as much memory as this value specifies before it starts to write data into temporary files. Also, several running sessions could be doing such operations concurrently. Therefore, the total memory used could be many times the value of work_mem; it is necessary to keep this fact in mind when choosing the value. Sort operations are used for ORDER BY, DISTINCT, and merge joins. Hash tables are used in hash joins, hash-based aggregation, memoize nodes and hash-based processing of IN subqueries.

Hash-based operations are generally more sensitive to memory availability than equivalent sort-based operations. The memory limit for a hash table is computed by multiplying work_mem by [hash_mem_multiplier](#). This makes it possible for hash-based operations to use an amount of memory that exceeds the usual work_mem base amount.

Source: [PostgreSQL 16 documentation](#)

© PostgreSQL Global Development Group



Planner Hints

Edit Planner Hints



pg_hint_plan settings

IndexScan(schema_tables schema_tables_database_id_schema_name_table_name_idx) 



Select a pg_hint_plan hints to edit

IndexScan



(badtable bad_index)

Save hint

 This hint will not be used. Please check if the object (table, index) name is correct.

Forces index scan on the table. Restricts to specified indexes if any.

Scan method hints enforce specific scanning methods on the target table. pg_hint_plan recognizes the target table by alias names if any.

Example

```
/*+ SeqScan(t1) IndexScan(t2 t2_pkey) */  
SELECT * FROM table1 t1 JOIN TABLE table2 t2 ON (t1.key = t2.key);
```



pganalyze Query Tuning Workbooks



pganalyze ~~Query Tuning~~ Workbooks



pganalyze Workbooks





Introducing pganalyze Query Advisor

[Live Demo]





Common Problems detected by Query Advisor

Inefficient Nested Loop

EXPLAIN Plan

Show: Est. Cost Runtime Rows Buffers Reads Writes
All metrics exclude children, except Rows. [Learn more](#) about reading EXPLAIN plans.

Plan	Rows	Est. Rows
1 Nested Loop <small>nested loop</small>	0	1
2 Index Scan (Forward) on issues <small>outer mis-estimate</small> on public.issues using index_issues_on_server_id_and_severity Filter: ((issues.state = ANY (\$1::integer[])) AND ((issues.database_id = ANY (\$2::bigint[... Index Cond: ((issues.server_id = ANY (\$1::uuid[])) AND (issues.server_id = \$2::uuid)) Rows Removed by Filter: 0 Rows Removed by Index Recheck: 0 Scan Direction: Forward	175	1
3 Index Scan (Forward) on issue_references <small>inner</small>	0	1 175

Query Summary

Runtime	I/O Read Time
667.00ms	665.24ms
Read From Disk	Total Est. Cost
648 kB	15

Jul 25 06:03:20am PDT · Plan Fingerprint: db5ccf2

Insights (2)

Query Advisor

Inefficient Nested Loop

- Estimated 1 loops, but 175 were executed.
- Inner side of this nested loop takes 666.40ms (100% of total)

[Create Workbook](#)

Other

Inefficient Nested Loop

Advisor Testing - Query #43934766 - Table List With Vacuum Settings

Overview Compare Plans Parameter Sets


+ New Query Variant

All Query Plans

Baseline

Variant 1 - MATERIALIZE function scan 

Variant 2 - Remove Function Scan 

Variant 4 - MATERIALIZE Initial Join + function scan 

Variant 6 - incremental_sort = off 

Query Advisor

No new insights found

Query

No match with existing queries

```
24 unchanged lines |
25     ORDER BY collected_at DESC
26     LIMIT $param_2
27 ) stis
)
SELECT
  st.database_id, d.datname AS database_name, st.id AS table_id, st.schema_name,
  st.table_name, sts.size_bytes, sti.last_vacuum, sti.last_autovacuum,
  sti.estimated_table_bytes, sti.ideal_table_bytes,
28 ) , stats materialized AS MATERIALIZED (
29   SELECT *
30   FROM
31     unpack_schema table_stats(database_ids := CASE
32       WHEN $param_4 IS NOT NULL THEN ARRAY[$param_5]::bigint[]
33     ELSE
34       ARRAY(
35         SELECT id
36         FROM databases
37         WHERE server_id = $server_id_3
38       )
39     END, ts := $ts)
40 ) , tables materialized AS MATERIALIZED (
41   SELECT st.*, d.datname, d.server_id
42   FROM
43     schema tables st
44   JOIN databases d ON st.database_id = d.id
45   WHERE
```

Inefficient Nested Loop

Advisor Testing - Query #43934766 - Table List With Vacuum Settings

Overview **Compare Plans** Parameter Sets

Plan Comparison Select plans

Compare **Est. Cost** Runtime I/O Read Time Rows Buffers

Plan A	Plan B	Plan A Runtime	Plan B Runtime
Baseline - Parameter Set 1	Variant 4 - MATERIALIZE Initial Join + function scan - Parameter Set 1		
-> Incremental Sort		4.17ms	
-> Nested Loop	-> Sort		10.32ms
	-> Nested Loop	2.32ms	1.39ms
	-> Hash Join		1.85ms
	-> CTE Scan on stats_materi...		8.01ms
	-> Hash		1.30ms
-> Nested Loop	-> Nested Loop	1.50ms	2.48ms
-> Nested Loop	-> Nested Loop	2.63ms	1.43ms
-> Nested Loop	-> Nested Loop	2.51ms	2.09ms
-> Nested Loop	-> Nested Loop	1,221.68ms	3.06ms
-> Nested Loop	-> Nested Loop	4.44ms	0.00ms
-> Nested Loop	-> Nested Loop	5.86ms	0.47ms
-> Index Scan ² on data...		0.05ms	
-> Materialize		4.93ms	
-> Nested Loop		0.00ms	
-> Nested Loop		0.41ms	
-> Index Scan ² on d...		0.03ms	
-> Index Scan ⁴³ on ...		2.74ms	
	-> CTE Scan on table...		6.02ms
-> Index Scan ⁴⁴ on s...	-> Index Scan ⁴⁴ on s...	8.08ms	7.87ms
	-> Index Scan ¹ on dat...		7.87ms

Summary Node Details Node Source

Plan A: Baseline - Parameter Set 1

Seen At	Total Est. Cost	Runtime
Feb 24 09:40pm	21,202	2,223.34ms
Plan Fingerprint	Read From Disk	I/O Read Time
# 1ed58d5	0 B	0.00ms

Plan B: Variant 4 - MATERIALIZE Initial Join + function scan - Parameter Set 1

Seen At	Total Est. Cost	Runtime
Feb 24 10:19pm	103,869	97.32ms
Plan Fingerprint	Read From Disk	I/O Read Time
# 63affef	0 B	0.00ms

Index usage

A B Index

- ✓ 1. databases_pkey
- ✓✓ 2. index_databases_on_server_id_and_datname
- ✓✓✓ 3. postgres_settings_server_id_name_idx
- ✓ 4. schema_table_infos_35d_20250201
- ✓✓✓ 5. schema_table_infos_35d_20250201
- ✓✓✓ 6. schema_table_infos_35d_20250201



ORDER BY + LIMIT Wrong Index Use

EXPLAIN Plan

Show: Est. Cost Runtime Rows Buffers Reads Writes
All metrics exclude children, except Rows. [Learn more](#) about reading EXPLAIN plans.

Plan	Runtime
1 Limit	0.08ms
2 Index Scan (Forward) on schema_tables limit/offset inefficient index	75,704.83ms

on public.schema_tables using schema_tables_pkey
Filter: ((schema_tables.removed_at IS NULL) AND (schema_tables.parent_table_id IS NULL) A...
Rows Removed by Filter: 10484856
Scan Direction: Forward

Query Summary

Runtime	I/O Read Time
75,705.95ms	68,306.44ms
Read From Disk	Total Est. Cost
1.4 GB	15,717

Jul 29 07:33:07pm PDT · Plan Fingerprint: [c4b7931](#)

Insights (2)

Query Advisor

ORDER BY + LIMIT Wrong Index

- The ORDER BY + LIMIT clause is causing an inefficient index to be selected by the planner
- Try rewriting the query by adding +0 to the ORDER BY column to use a different index scan

Create Workbook

Other >

ORDER BY + LIMIT Wrong Index Use

Query #43934723 ORDER BY LIMIT

Overview Compare Plans Parameter Sets

+ New Query Variant

All Query Plans

Baseline

Variant 1

Apply ORDER BY rewrite

Apply rewrite

Query Advisor

No new insights found

Query

No match with existing queries

Query tags

Variant 1

With parameter names

SQL Query Unified Diff Split Diff

```
4 unchanged lines
5 schema_tables.database_id = $database_id
6 AND schema_tables.removed_at IS NULL
7 AND schema_tables.parent_table_id IS NULL
8 ORDER BY schema_tables.id ASC
9 ORDER BY schema_tables.id + $zero ASC
10 LIMIT $param
```

Show all code

Query Plans

PLAN	VARIANT	PARAMETER SET	EST. COST	RUNTIME
<input type="checkbox"/> 0b26989	Variant 1	Parameter Set 1	34,091	⚡ 0.97ms
<input type="checkbox"/> 697cbe3	Variant 1	Parameter Set 2	135,419	4.83ms



ORDER BY + LIMIT Wrong Index Use

Plan Comparison Select plans

Compare ③: Est. Cost Runtime I/O Read Time Rows Buffers

Plan A	Plan B	Plan A	Plan B
Baseline - Parameter Set 2	Variant 1 - Parameter Set 2	Runtime	Runtime
-> Limit	-> Limit	0.00ms	0.00ms
-> Index Scan ² on schema_tables		6,683.13ms	
	-> Gather Merge		4.56ms
	-> Sort		0.03ms
	-> Bitmap Heap Scan on sche...		0.16ms
	-> Bitmap Index Scan ¹		0.07ms

Summary Node Details Node Source

Plan A: Baseline - Parameter Set 2

Seen At	Total Est. Cost	Runtime
Jul 24 12:31am	8,577	6,683.23ms

Plan Fingerprint	Read From Disk	I/O Read Time
c4b7931	0 B	0.00ms

Plan B: Variant 1 - Parameter Set 2

Seen At	Total Est. Cost	Runtime
Jul 24 12:37am	135,419	4.83ms

Plan Fingerprint	Read From Disk	I/O Read Time
697cbe3	0 B	0.00ms

Index usage

A B Index

- ✓ 1. schema_tables_database_id_schema_name_table_...
- ✓ 2. schema_tables_pkey

Additional insights in the works:

- OR => UNION transformation
- Planner hint suggestions
- Memoize mis-estimates
- GROUP BY column ordering
- Settings changes
 - work_mem
 - random_page_cost
 - etc.

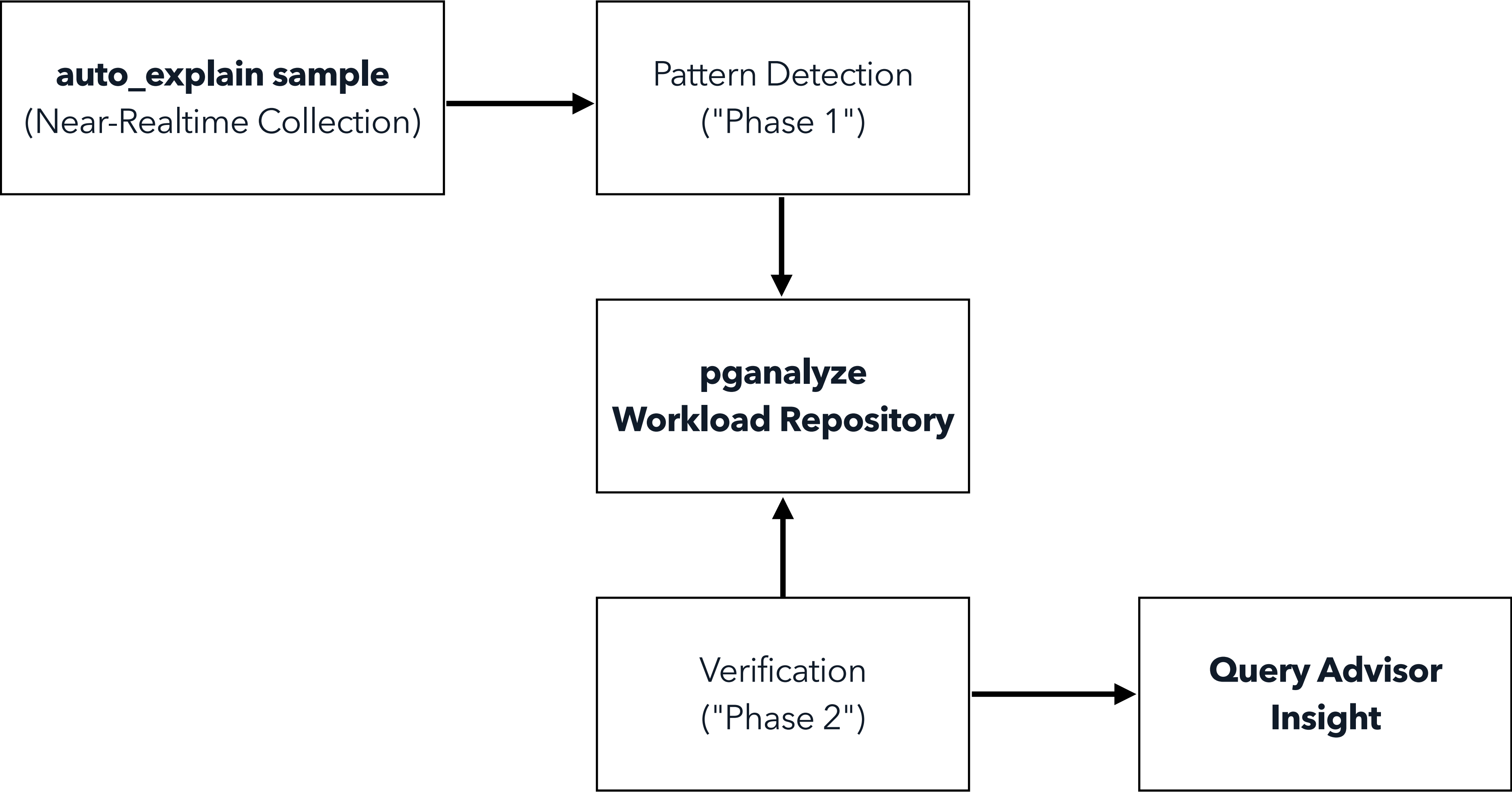


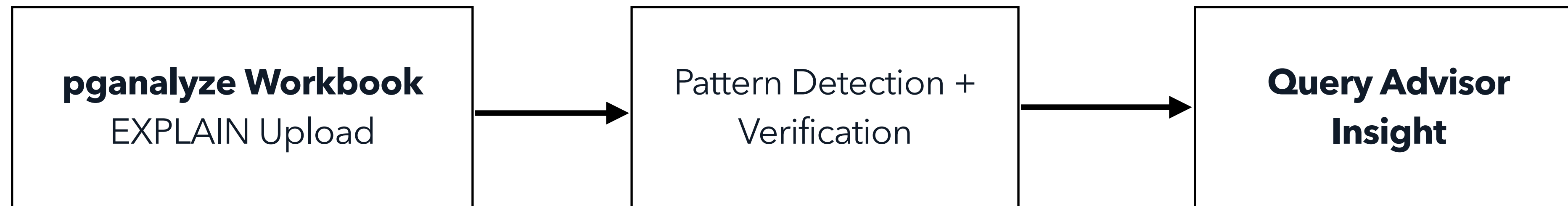
Behind the scenes of pganalyze Query Advisor

Query Tuning Advisor at a High Level

- **Built for scale, running behind the scenes**
- **Currently processing 7 million samples per day**
- **Custom code, not using LLM-driven analysis**
(LLMs don't scale for automated analysis)

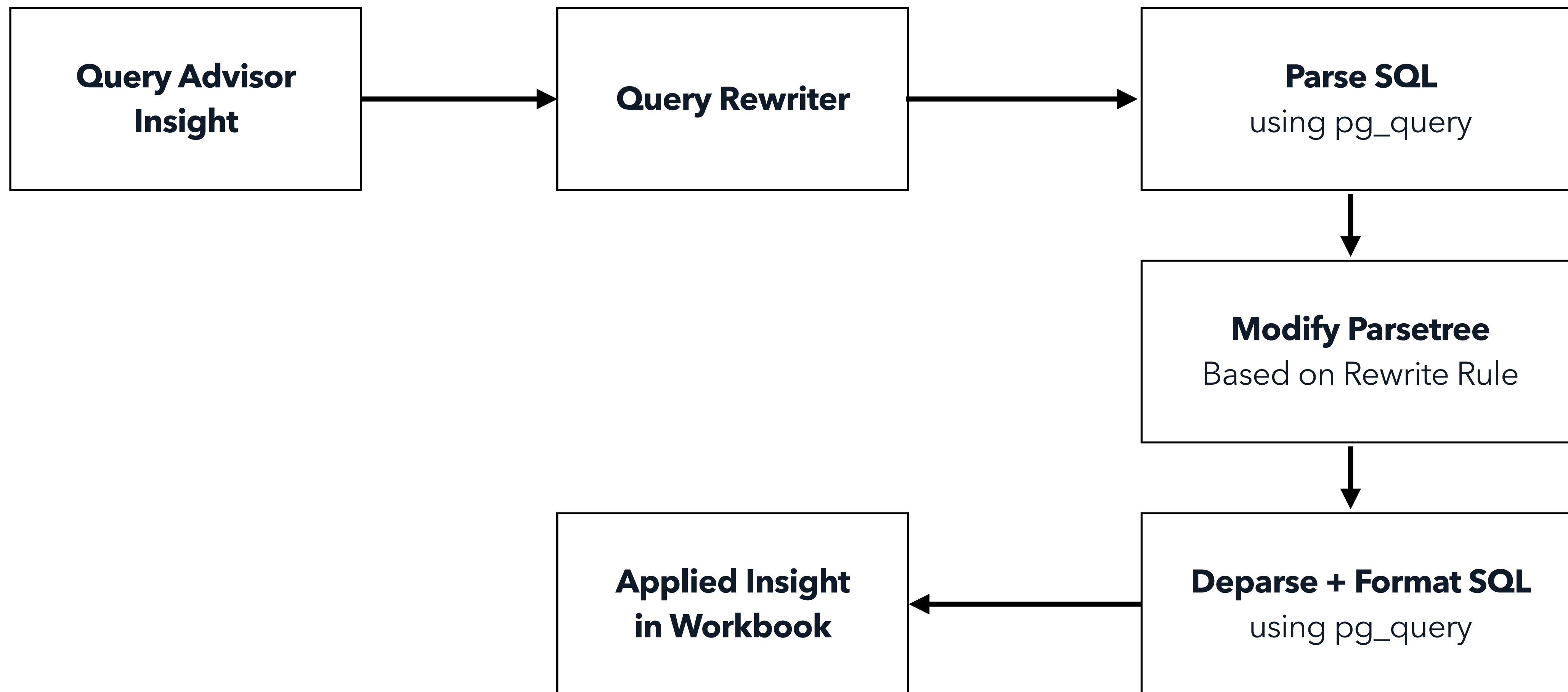






"Codemods" for queries







Early Access & Looking Ahead

For best results:

- Detecting problematic queries in the background requires `auto_explain` with `log_analyze` (but not `log_timing`)
- Workbook Collector workflow makes testing easy



Early Access Program starts August 11th

Let us know if you're interested through the post-webinar survey, or by sending an email.



Looking Ahead:

- Expanded coverage of common Postgres problems
- Automatic testing of rewrites in the background (opt-in)
- Exploring how Workbooks and Query Advisor can be a tool that integrates into pre-production tests or LLM-based workflows





Thank you!

Try out pganalyze:

[PGANALYZE.COM](https://pganalyze.com)

Reach out for any questions:

lukas@pganalyze.com
