



Evolving How We Tune Postgres

Lukas Fittl
lukas@pganalyze.com

A Typical Organization Using Postgres

The Application Teams



Database A

Database B

Database C

Database D

The Data Platform Team / DBAs



A Typical Organization Using Postgres

The Application Teams



The Data Platform Team / DBAs

Database A
Database B
Database C
Database D
Database E
Database F
Database G
Database H
Database I
Database J
Database K



Postgres Problems?

"Just Throw an LLM at it"

A Typical Organization Using Postgres

The Application Teams



Database A

Database B

Database C

Database D

Database E

Database F

Database G

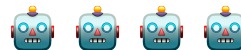
Database H

Database I

Database J

Database K

"AI DBAs"



LLMs are good at producing something reasonable looking.

They're still pretty bad at abstractions, and they are not deterministic.

LLMs are a tool, and we should use them for the right problems.



Engine VS Workload Tuning

Engine Tuning

For **Engine Tuning**, We Need To Build The Right Abstractions

Ideally, fix Postgres to do things the right way.

If we can't, expose enough data so that external tools or operators can make informed decisions.

Parameter Tuning - It Really Should Be Done By Postgres

I/O workers are a good example of that in PG 19.

Sometimes its about re-designing the abstraction, for example `random_page_cost`.

Plan Tuning - It Kinda Should Be Done By The Database

But when the engine doesn't, having mechanisms like **pg_plan_advice** is a practical stop-gap.

There is more to be done. **Join our session tomorrow:**

11:25 – 12:15 - The Future of Plan Management and
Feedback Loops (Auditorium 3)

For **Workload Tuning**, We Need To
Enable Engineers and Their Tools.

Workload Tuning

What's in a Workload?



Workload Tuning

What's in a Workload?



Really this is about monitoring
and feedback loops.

**How do engineers know that
they need to adjust their workload?**

At pganalyze, we've invested in
better tooling for Postgres for over a decade now.

Workload Tuning

Here are three of my favorite pganalyze features
that **help engineers work better**:

Plan Samples and Statistics



WITH q AS (...), t1 AS (...), t2 AS (...), t3 AS (...), t4 AS (...), t5 AS (...) SELECT \$3

Avg Time 1,209.47ms Calls Per Minute 4.69 / min
☐ Compare to 7 days ago

fingerprint 4ae589340b7bdbeb role pgaweb_workers line /app/services/delete... job Storage::DeleteOldSn... [View all query tags](#)

[Overview](#) [Index Advisor](#) [Query Samples](#) **5+** [EXPLAIN Plans](#) **5+** [Query Tags](#) **5+** [Log Entries](#) **100+**

Check-Up

1 new issue

[Info](#) Query #44015223 takes 990 ms on average (762612 ms max, 5.84 MB read from disk per call, 6721 calls in last 24h)

Plan Statistics

PLAN	EST. COST	AVG RUNTIME	PLAN SAMPLES	PLAN NODES
8df92c6	1,251,044	248,559.21ms	11	Result · ModifyTable · Nested Loop +26 more
8598ed3	385,575	40,129.54ms	39	Result · ModifyTable · Nested Loop +25 more

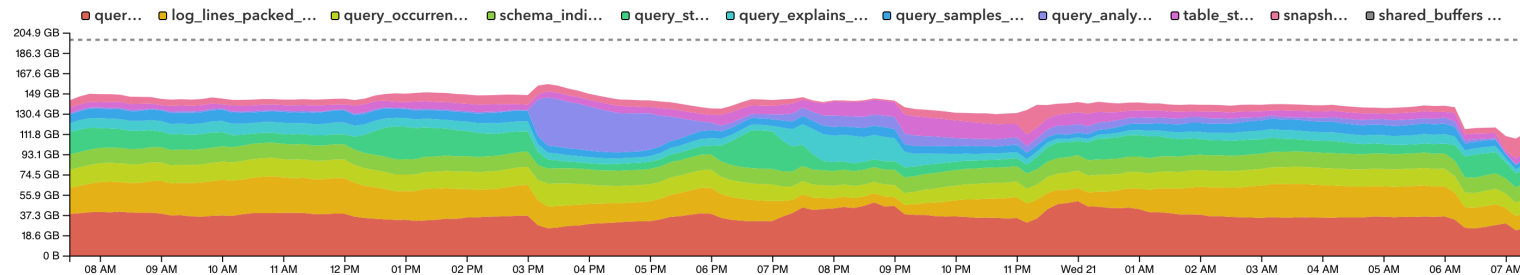
Plan Samples (50+)

EXECUTED AT	PLAN	EST. COST	RUNTIME	I/O READ TIME	READ FROM DISK	PLAN NODES
☐ 2026-10-20 11:54:00pm PDT	8df92c6	1,456,754	835,603.75ms	0.00ms	0 B	Result · ModifyTable · Nested Loop +26 more
☐ 2026-10-20 11:49:19pm PDT	8df92c6	1,580,387	33,241.46ms	0.00ms	0 B	Result · ModifyTable · Nested Loop +26 more
☐ 2026-10-20 11:48:50pm PDT	8df92c6	1,580,387	35,214.68ms	0.00ms	0 B	Result · ModifyTable · Nested Loop +26 more
☐ 2026-10-20 11:48:13pm PDT	8df92c6	1,580,387	3,146.17ms	0.00ms	0 B	Result · ModifyTable · Nested Loop +26 more

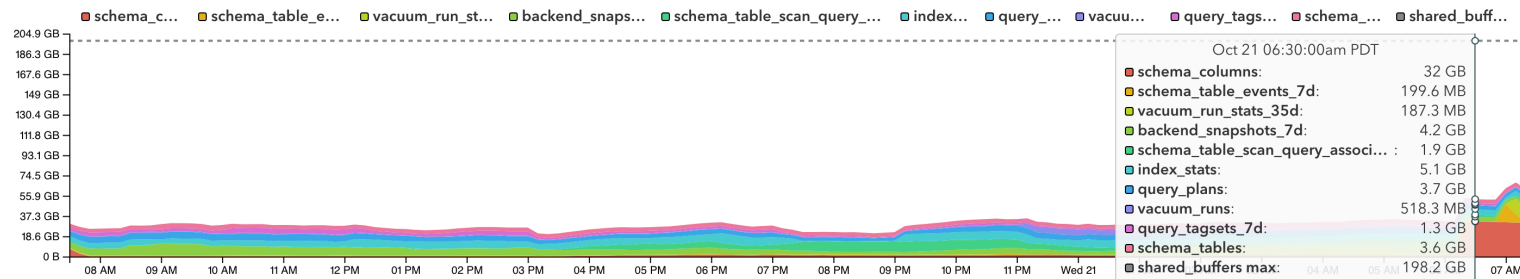
Buffer Cache Monitoring

Postgres Buffer Cache

Top 10 Tables



Top 10 Outliers



Cluster-Aware Index Advisor

Index Advisor

[Overview](#) [Missing Indexes](#) [Unused Indexes](#) [Configure](#) [Status](#)

Total Data Size

905 MB

▲ 12 MB

Total Index Size

1.2 GB

Table Writes

24

per minute

Avg. Index Write Overhead

0.68

index bytes per table byte ⓘ

Insights for Database in Cluster (3)

☐ Hide Acknowledged Insights

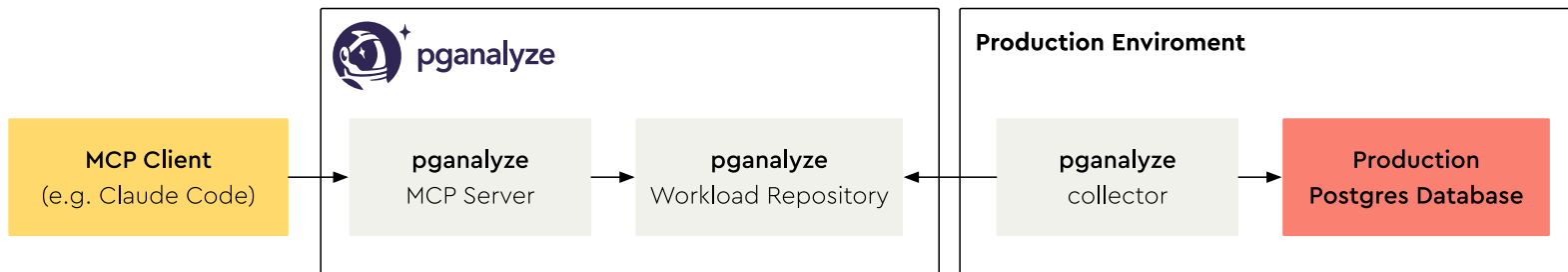
Search...

IMP...	TABLE	INDEX	INDEX SIZE	LAST USED	TABLE WRITES / MIN	INDEX WRITE OVERHEAD
	bluebox.customer_status_log	idx_csl_customer (+2 more)	8.2 MB	2026-02-27	0.000	-0.58
	bluebox.zip_code_info	us_postal_code_geog_gix	2.5 MB	2026-02-27	0.000	-0.00
	bluebox.film	film_film_genre_ids_idx	88 kB	2026-02-27	0.000	-0.00

AI Tools are an inflection point in how we write software,
and how we operate our systems, including Postgres.

**We need high quality data
about Postgres now more than ever.**

The pganalyze MCP Server



But even with perfect monitoring data,
and a workload repository + MCP server to access it,
we're missing something in the age of AI.

Let's look at an example Claude output:

- Here's the full analysis of the two plan variants:



Query

```
WITH ranked AS (  
  SELECT qe.*,  
         row_number() OVER (PARTITION BY qe.query_fingerprint, qe.database_id  
                           ORDER BY qe.runtime_ms DESC) AS rn  
  FROM query_explains_7d qe  
 WHERE qe.server_id = $1  
       AND qe.seen_at BETWEEN $2 AND $3  
       AND jsonb_array_length(qe.node_insights) > $4  
)  
SELECT * FROM ranked WHERE rn <= $5
```

Plan Variant 1: Index Scan (11.9s, the common plan)

WindowAgg (actual: 11,121ms..11,126ms, rows=450)

Buffers: hit=19,013 read=14,027 I/O: 10,984ms

└─ Sort (quicksort, 14MB memory)

└─ Append (actual: 1ms..11,096ms, rows=13,365)

└─ Index Scan query_explains_7d_20260219 (964ms, 1,039 rows, 7,199 filtered)

└─ Index Scan query_explains_7d_20260220 (1,051ms, 1,061 rows, 8,447 filtered)

└─ Index Scan query_explains_7d_20260221 (970ms, 906 rows, 5,057 filtered)

└─ Index Scan query_explains_7d_20260222 (928ms, 980 rows, 4,206 filtered)

└─ Index Scan query_explains_7d_20260223 (2,593ms, 1,867 rows, 7,253 filtered)

└─ Index Scan query_explains_7d_20260224 (2,713ms, 3,419 rows, 9,422 filtered)

└─ Index Scan query_explains_7d_20260225 (1,797ms, 3,575 rows, 7,590 filtered)

The problem is clear - almost all execution time is spent on disk I/O (10,984ms out of 11,873ms = 92.5% I/O wait):

1. The index only narrows by (server_id, seen_at), which returns ~50K-65K rows across all 8 daily partitions
2. The jsonb_array_length(node_insights) > 0 filter is applied as a post-fetch filter, requiring PostgreSQL to read all the TOAST data for the node_insights JSONB column from disk for every matching row
3. Only ~20% of rows pass the filter (13,365 kept out of ~64K fetched) - so ~80% of I/O is wasted
4. Buffer hit ratio is terrible because the JSONB TOAST pages are large and rarely cached

Recommendations

1. Add a stored/generated column like has_node_insights boolean GENERATED ALWAYS AS (jsonb_array_length(node_insights) > 0) STORED and include it in the index. This avoids detoasting the JSONB just to check its length.
2. Alternative: partial index on (server_id, seen_at) with WHERE jsonb_array_length(node_insights) > 0. This would eliminate the 80% of rows that are filtered out, at the cost of maintaining the index on writes.
3. Alternative: expression index including jsonb_array_length(node_insights) as an indexed column, e.g. CREATE INDEX ON query_explains_7d (server_id, seen_at, jsonb_array_length(node_insights)) WHERE jsonb_array_length(node_insights) > 0.

Option 2 (partial index) is likely the best tradeoff - it would cut I/O by ~80% with minimal write overhead, since query_explains_7d is partitioned daily and old partitions stop receiving writes.

Chat can be a reasonable interactive experience,
but its an unusable artifact to look at.

So we've asked ourselves,
what could agents produce that is useful?

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



Keiko Oda
Engineering



Maciek Sakrejda
Engineering



Sean Linsley
Engineering



Jens Nikolaus
Design



Lukas Fittl
Product

Published on:
January 08, 2025

📁 Query Performance

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](#)

pganalyze Workbooks



pganalyze

ORGANIZATION
pganalyze ▼

[Dashboard](#)

[Query Performance](#)

[Query Advisor](#)

[Index Advisor](#)

[VACUUM Advisor](#)

[Workbooks](#)

[Schema Statistics](#)

[Log Insights](#)

[Connections](#)

[Config Settings](#)

[System](#)

[Alerts & Check-Up](#)

[Settings](#)

[What's New](#)

Jens Nikolaus

Server
● prod-db-main Primary

Database
pgaweb

Query #7340988975

Overview Compare Plans Parameter Sets Activity

Baseline With parameter names ▼

```
1 WITH data AS (  
2   SELECT collected_at, state, waiting_for_lock, count($count_arg) AS count  
3   FROM  
4     backend_snapshots_7d s,  
5     unnest(s.database_ids, s.backend_types, s.states, s.waiting_for_locks)  
6   b(database_id, backend_type, state, waiting_for_lock)  
7   WHERE  
8     s.server_id = $server_id  
9     AND s.collected_at BETWEEN $collected_at AND $collected_at_2  
10    AND b.database_id = $database_id  
11    AND b.backend_type = $backend_type  
12  GROUP BY 1, 2, 3  
13  UNION  
14  SELECT  
    collected_at, $param_2 AS state, $param_3 AS waiting_for_lock,
```

Show all code

Query Plans

	PLAN	VARIANT	PARAMETER SET	EST. COST	RUNTIME
☐	bb64751	Baseline	Parameter Set 1	145,494	3,359.55ms
☐	6f49cc5	Baseline	Parameter Set 2	2,403	⚡ 57.51ms

Compare selected plans No plans selected

+ New Query Variant

All Query Plans

Baseline

Query Advisor
No insights found

Query
#7340988975

Query tags
fingerprint 03d4e32f13eb0203

pganalyze Workbooks



pganalyze

ORGANIZATION
pganalyze ▼

Dashboard

Query Performance

Query Advisor

Index Advisor

VACUUM Advisor

Workbooks

Schema Statistics

Log Insights

Connections

Config Settings

System

Alerts & Check-Up

Settings

What's New

Jens Nikolaus

Server
● prod-db-main Primary X ▼

Database
pgaweb X ▼

Query #7340988975

Variant: Baseline ▼ Parameter Set: Parameter Set 1 ▼

Overview Compare Plans Parameter Sets Activity

Node Tree Grid Text JSON

7 Aggregate 277.59ms

8 Nested Loop 159.53ms

9 Append 5.44ms

10 Index Scan (Forward) stale_stats 11.76ms

11 Index Scan (Forward) stale_stats 11.30ms

12 Index Scan (Forward) on backe... 11.69ms

13 Index Scan (Forward) on backe... 12.09ms

14 Index Scan (Forward) on backe... 12.25ms

15 Index Scan (Forward) on backe... 11.77ms

16 Index Scan (Forward) on backe... 8.78ms

17 Memoize 321.25ms 58K

18 f() Function Scan 662.44ms 51K

19 Subquery Scan 130.88ms

20 Nested Loop 164.08ms

21 Append 5.47ms

22 Index Scan (Forward) o... stale_stats 12.21ms

23 Index Scan (Forward) o... stale_stats 12.14ms

24 Index Scan (Forward) on backend... 12.10ms

25 Index Scan (Forward) on backend... 12.82ms

26 Index Scan (Forward) on backend... 12.76ms

27 Index Scan (Forward) on backend... 12.70ms

28 Index Scan (Forward) on backend... 9.26ms

29 Memoize 263.38ms 58K

30 f() Function Scan 662.44ms 51K

Function Call: unnest(s.database_ids), unnest(...
Filter: ((b.database_id = '-662599962'::intge...
Rows Removed by Filter: 12

Runtime 3,359.55ms I/O Read Time 0.00ms

Read From Disk 0 B Total Est. Cost 145,494

Sep 26 01:10:39pm MDT · Plan Fingerprint: bb64751

Insights

Query Advisor

No Query Advisor insights found for this plan.

Stale Statistics ▼

10 referenced a table that was last analyzed 7 days ago

11 referenced a table that was last analyzed 6 days ago

22 referenced a table that was last analyzed 7 days ago

23 referenced a table that was last analyzed 6 days ago

Safely enable AI Coding Tools to use EXPLAIN ANALYZE on production: Introducing Workbooks in the pganalyze MCP Server

Connecting AI coding tools to production is typically ill-advised: it's not clear what might happen, and how they might cause availability or other issues. But in practice that often means the feedback loop that AI tools rely on is broken, and requires a lot of manual intervention. In the context of database query optimization, the practical problem becomes "How do I run EXPLAIN ANALYZE on production" for a suggested query rewrite.

For humans it's often just a matter of getting the right access and sign-offs, being extra careful not to impact the system, and potentially running on replicas or clones. For AI tools, that level of access is typically not available, and for good reason. But what if we designed a purpose-built system that enabled safe running of EXPLAIN ANALYZE for AI tools, fed them back the resulting query plan, and also recorded it for easier reference?

Today we're excited to release query benchmarking with Workbooks in the [pganalyze](#)



Ryan Booz
Solutions Engineering



Keiko Oda
Engineering

Published on:
October 15, 2026

AI & Automation

In this article:

- [Why AI query fixes need a tuning loop](#)
- [How Workbooks benchmarking works in the pganalyze MCP Server](#)
 - [See the loop in action](#)

pganalyze Workbooks - Now in the MCP Server



TableTreeMap: schema_table_infos lateral (baseline vs pruned)

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

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
☐  84f94a7	Baseline	Parameter Set 1	10,432	142.95ms
☐  84f94a7	Baseline	Parameter Set 2	10,432	150.94ms
☐  168d9fb	Bounded collected_at (partition pru...	Parameter Set 1	2,346	 13.73ms
☐  09aaa1d	Bounded collected_at (partition pru...	Parameter Set 2	3,562	27.97ms

pganalyze Workbooks - Now in the MCP Server



TableTreeMap: schema_table_infos lateral (baseline vs pruned)

Overview Compare Plans Parameter Sets Activity

Plan Comparison

Select plans

Compare: ☐ Est. Cost ☒ Runtime ☐ Rows ☐ Buffers ☐ Reads ☐ Writes

Plan A

Baseline - Parameter Set 1

Plan B

Bounded collected_at (partition pruning) -
Parameter Set 1

Plan A

Runtime

Plan B

Runtime

-> Result

-> Nested Loop

-> Index Scan³⁸ on schema_tables

-> Limit

-> Append

-> Index Scan³⁴ on schema_tabl...

-> Index Scan⁵ on schema_tabl...

-> Index Scan¹⁶ on schema_tabl...

-> Index Scan³³ on schema_tabl...

-> Index Scan¹⁵ on schema_tabl...

-> Index Scan³² on schema_tabl...

-> Index Scan³¹ on schema_tabl...

-> Index Scan³⁰ on schema_tabl...

-> Index Scan²⁹ on schema_tabl...

-> Index Scan²⁸ on schema_tabl...

-> Index Scan²⁷ on schema_tabl...

-> Index Scan⁴ on schema_tabl...

-> Index Scan²⁶ on schema_tabl...

-> Index Scan²⁵ on schema_tabl...

-> Index Scan²⁴ on schema_tabl...

-> Nested Loop

-> Index Scan³⁸ on schema_tables

-> Limit

-> Append

-> Index Scan³⁴ on schema_table...

-> Index Scan⁵ on schema_table...

-> Index Scan¹⁶ on schema_table...

-> Index Scan³³ on schema_table...

0.48ms

1.78ms

4.10ms

2.09ms

0.00ms

2.09ms

4.19ms

4.19ms

4.19ms

4.19ms

4.19ms

2.09ms

2.09ms

4.19ms

4.19ms

4.19ms

4.19ms

4.19ms

4.19ms

4.19ms

0.25ms

2.83ms

0.00ms

4.19ms

0.00ms

2.09ms

2.09ms

2.09ms

2.09ms

pganalyze Workbooks - Now in the MCP Server



The Human Variant

Edit name and description

1 Rewrite Query and Edit Planner 2 Run EXPLAIN

Edit Unified Diff Split Diff

```
1 WITH server_schema_table_infos AS MATERIALIZED (  
2   SELECT info.schema_table_id, info.estimated_table_bytes,  
3         info.ideal_table_bytes, info.toast_reltuples, info.collected_at  
4   FROM schema_table_infos_35d info  
5  WHERE server_id = $server_id  
6 )  
7 SELECT  
8   t.*, info.estimated_table_bytes, info.estimated_bloat_bytes,  
9   info.estimated_pct_bloat, info.toast_reltuples  
10 FROM  
11   schema_tables t  
12  LEFT JOIN LATERAL (  
13    SELECT  
14      *,  
15      GREATEST(estimated_table_bytes - ideal_table_bytes, $param_5) AS estimated_bloat_bytes,  
16      CASE estimated_table_bytes  
17        WHEN $param_6 THEN $param_7  
18        ELSE round((GREATEST(estimated_table_bytes - ideal_table_bytes, $param_9) * $param_8) / estimated_table_bytes)  
19      END AS estimated_pct_bloat  
20  FROM server_schema_table_infos  
21  WHERE schema_table_id = t.id  
22  ORDER BY collected_at DESC  
23  LIMIT $param_2  
24 ) info ON $param  
25 WHERE  
26   database_id = $database_id  
27  AND removed_at IS NULL
```

pganalyze Workbooks - Now in the MCP Server



The Human Variant

Edit name and description

Cancel

1 Rewrite Query and Edit Planner 2 Run EXPLAIN

Manual Workflow

Collector Workflow

```
WITH server_schema_table_infos AS MATERIALIZED (  
  SELECT  
    info.schema_table_id, info.estimated_table_bytes, info.ideal_table_bytes,  
    info.toast_reltuples, info.collected_at  
  FROM schema_table_infos_35d info  
  WHERE server_id = $server_id
```

RUN

EXECUTED

RUN

☐ Parameter Set 1

-

☐ Parameter Set 2

-

Collect Plans

pganalyze Workbooks - Now in the MCP Server



TableTreeMap: schema_table_infos lateral (baseline vs pruned)

Overview **Compare Plans** Parameter Sets Activity

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
☐  84f94a7	Baseline	Parameter Set 1	10,432	142.95ms
☐  84f94a7	Baseline	Parameter Set 2	10,432	150.94ms
☐  168d9fb	Bounded collected_at (partition pruned)	Parameter Set 1	2,346	13.73ms
☐  09aaa1d	Bounded collected_at (partition pruned)	Parameter Set 2	3,562	27.97ms
☐  966e3b0	The Human Variant	Parameter Set 1	488,380	 4.70ms
☐  966e3b0	The Human Variant	Parameter Set 2	488,380	4.99ms

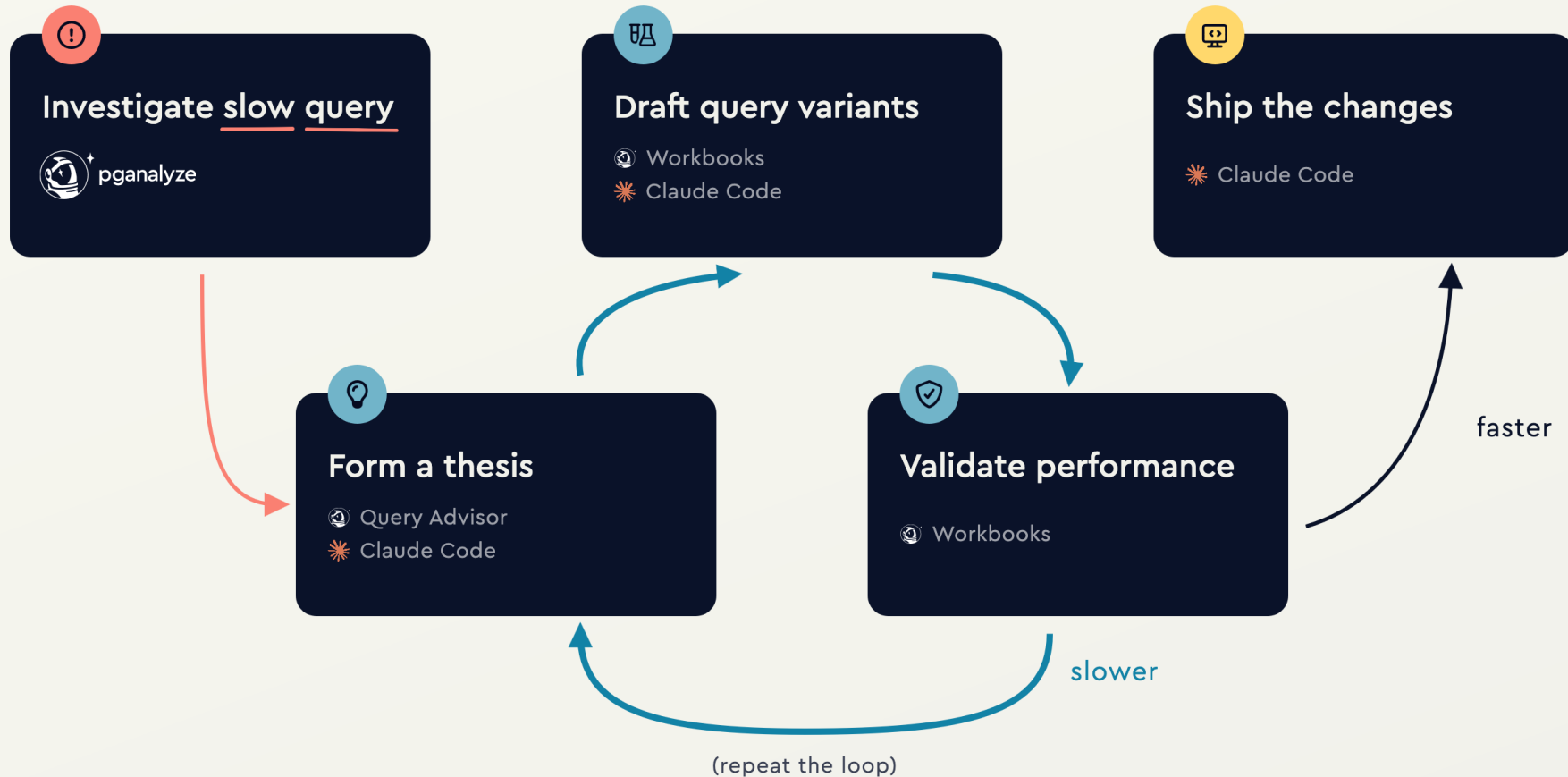
TableTreeMap: schema_table_infos lateral (baseline vs pruned)

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

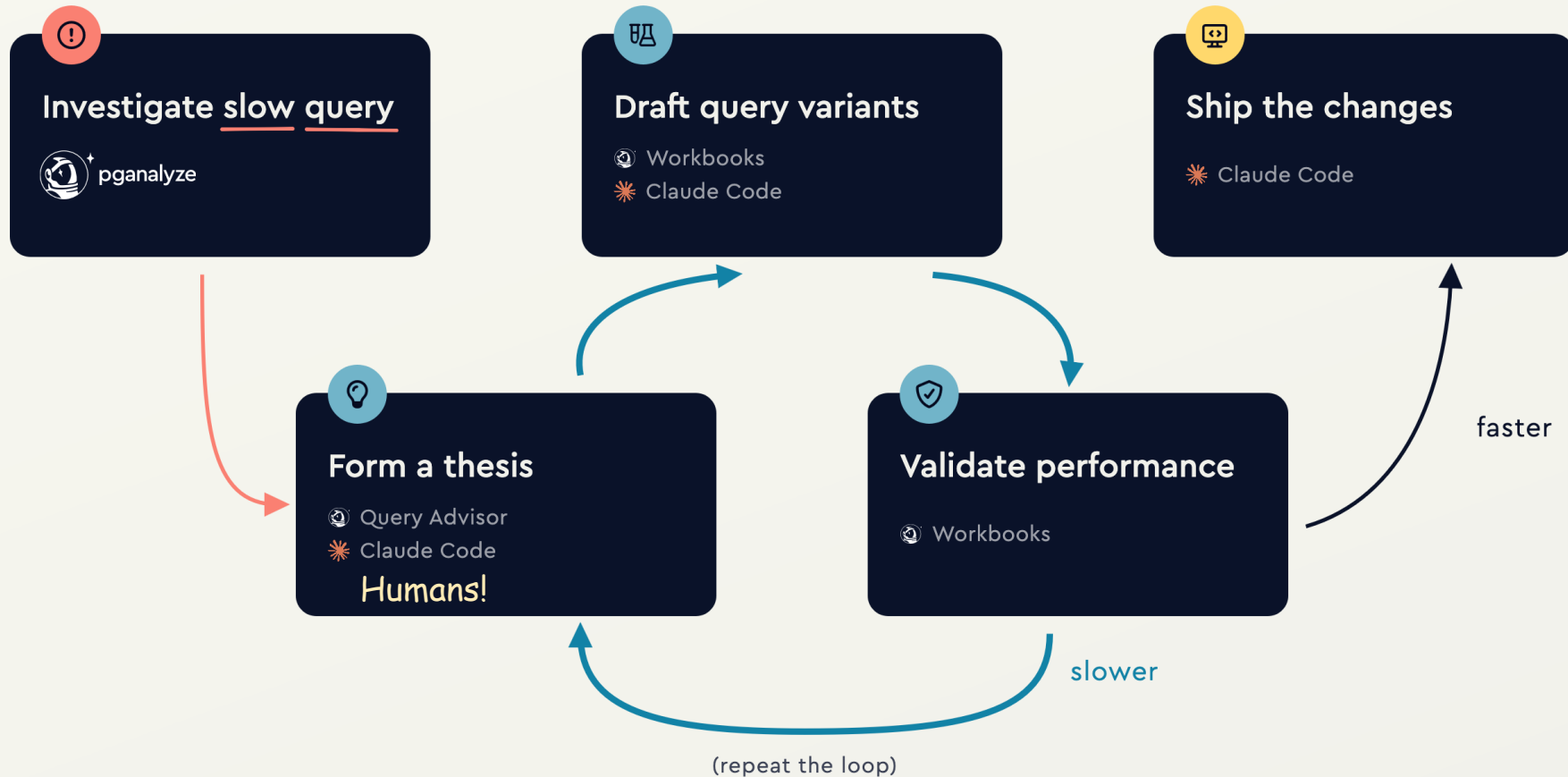
Activity

- 📌 **2026-07-30 08:57:18am PDT**: new variant "The Human Variant" created by  Lukas Fittl
- ✎ **2026-07-30 08:53:52am PDT**: workbook description updated by  Lukas Fittl MCP
- 📌 **2026-07-30 08:50:51am PDT**: new variant "Bounded collected_at (partition pruning)" created by  Lukas Fittl MCP
- 📌 **2026-07-30 08:50:16am PDT**: new variant "Baseline" created by  Lukas Fittl MCP
- ☰ **2026-07-30 08:50:16am PDT**: workbook created by  Lukas Fittl MCP

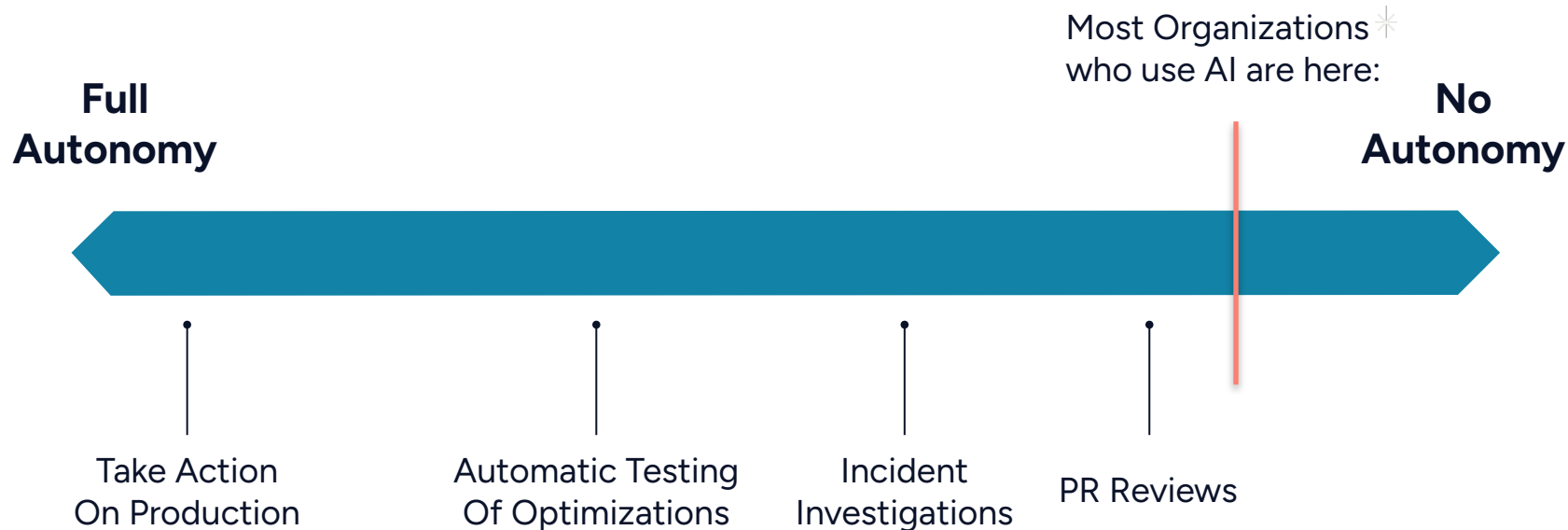
The Query Tuning Loop



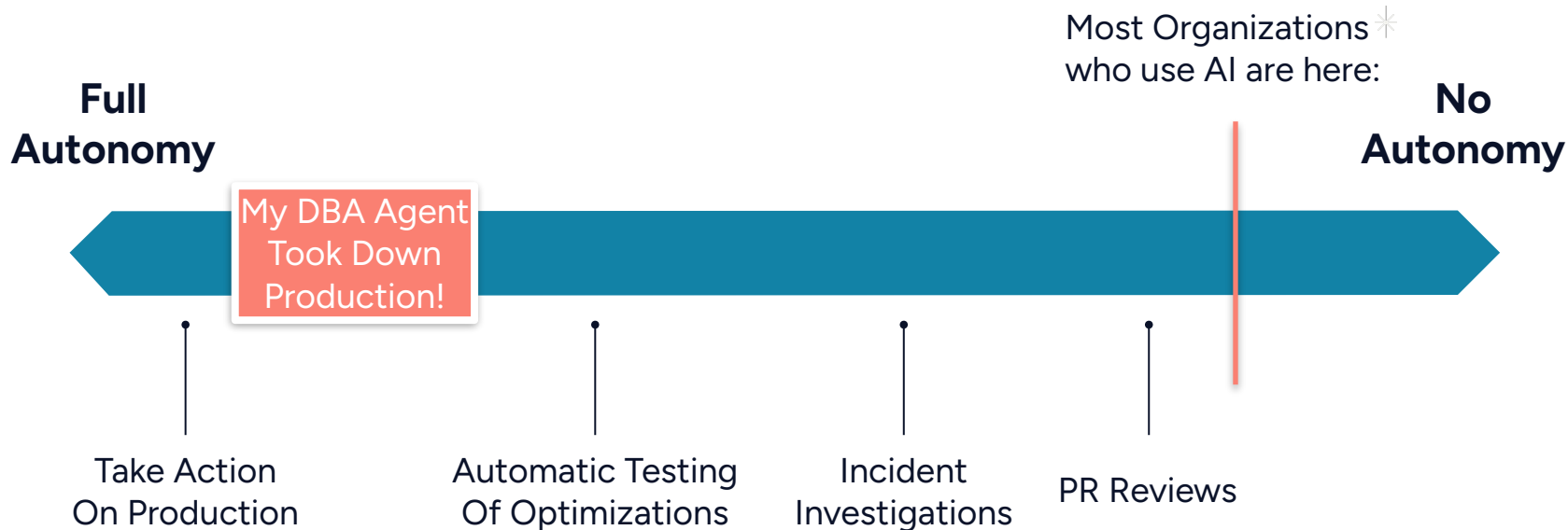
The Query Tuning Loop



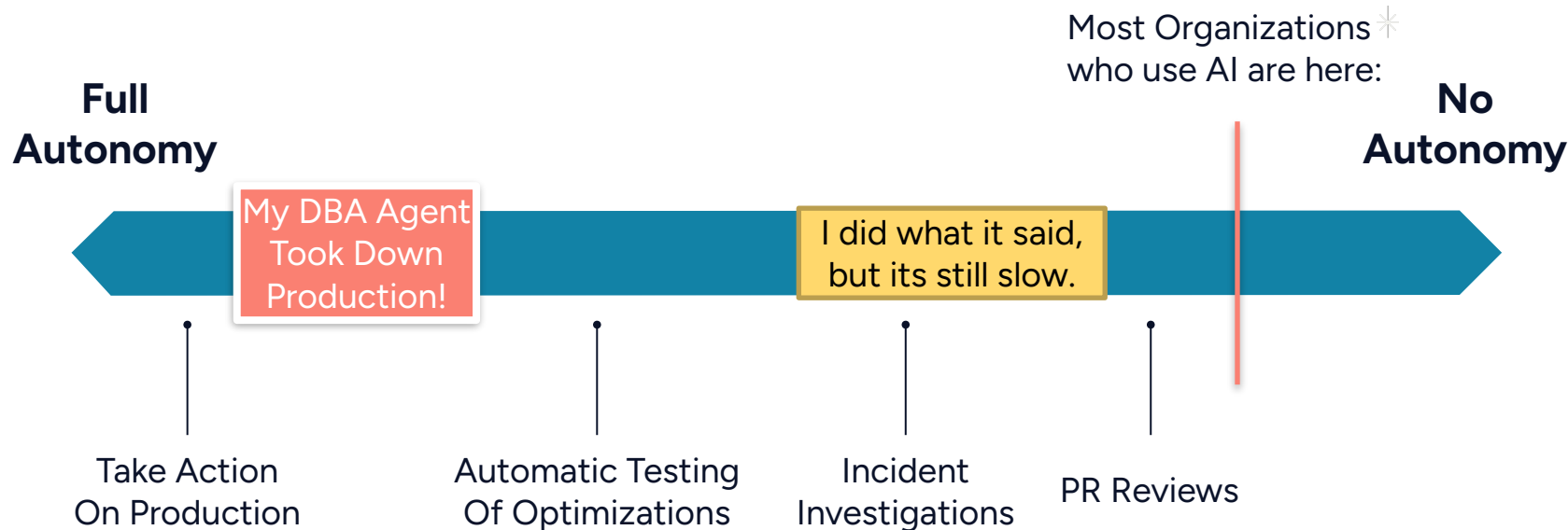
The Agent Autonomy Problem



The Agent Autonomy Problem



The Agent Autonomy Problem



The Agent Autonomy Problem

**Full
Autonomy**

My DBA Agent
Took Down
Production!

Take Action
On Production

pganalyze MCP
with Workbooks

Automatic Testing
Of Optimizations

I did what it said,
but its still slow.

Incident
Investigations

PR Reviews

Most Organizations
who use AI are here:

**No
Autonomy**



Its about How **We** Tune Postgres

Not how AI Tools Tune Postgres

Someone needs to take responsibility for how a system works.

Better Artifacts help document what was tested,
so we can reason about changes after the fact.



Thank you! ✨

Learn more at pganalyze.com

Contact us: lukas@pganalyze.com