

pganalyze in Action

The Latest Features for Tuning Postgres



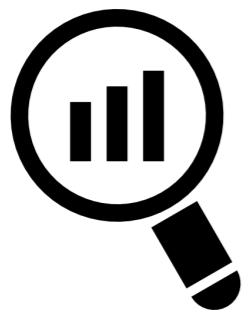
Lukas Fittl

CEO & Founder



Why pganalyze?

When Postgres Slows Down, You Feel It Everywhere



Slow
Query



App
Latency



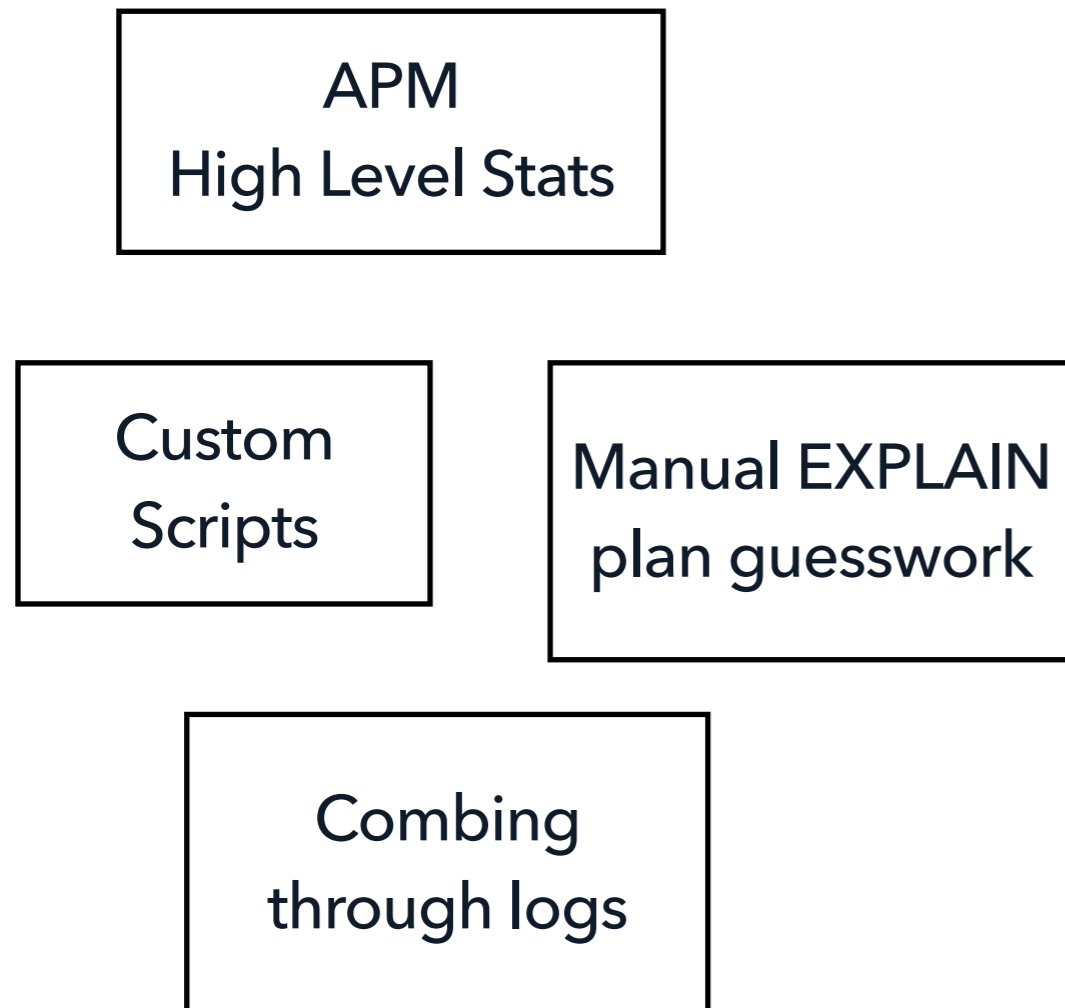
User
Complaints



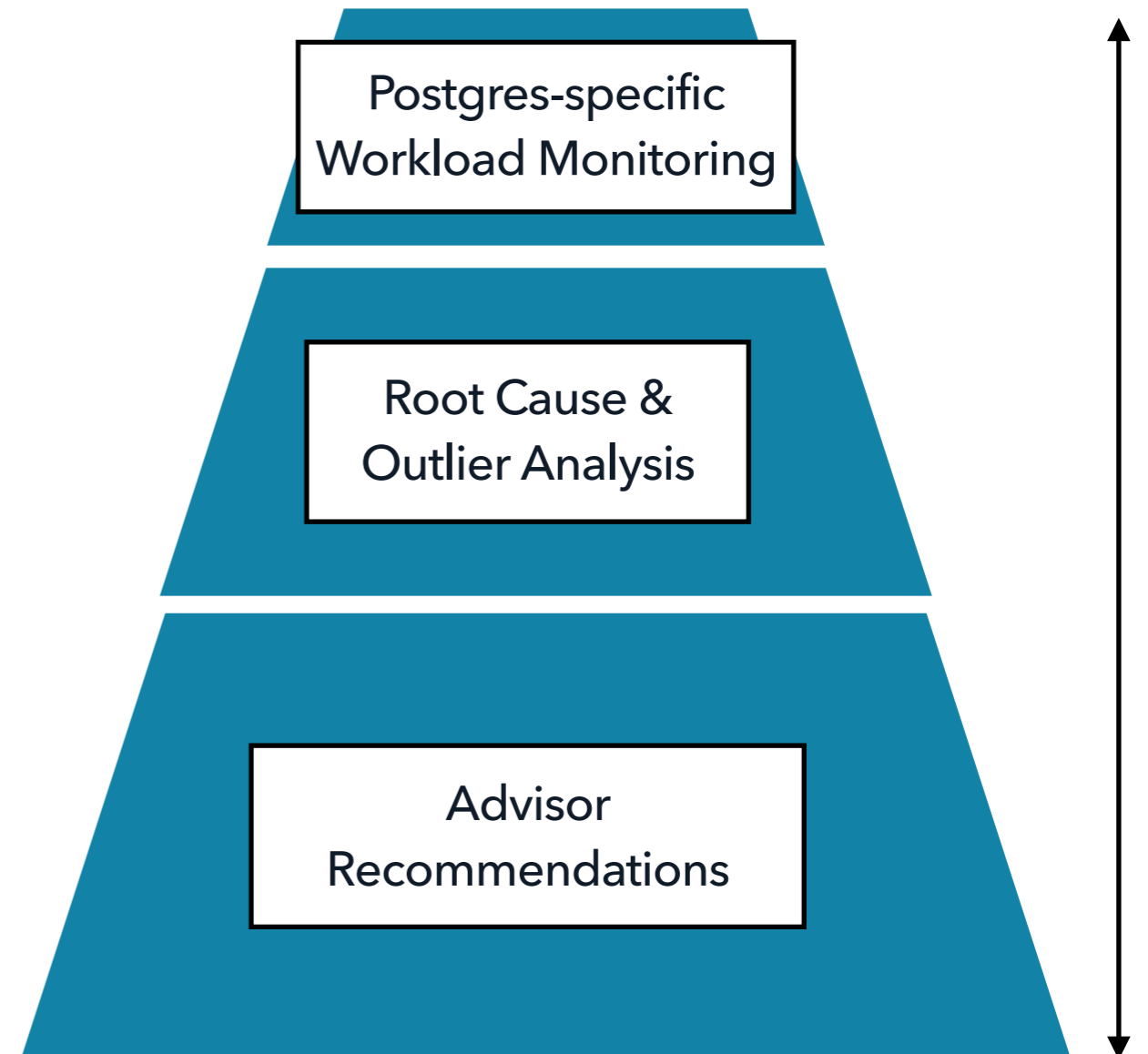
Developer
Scramble

Engineering Teams Succeed With an End-to-End Solution Built for Postgres

Generic tools:
fragmented & hard to use



pganalyze: unified &
purpose-built for Postgres



How Notion Runs Postgres at Scale with pganalyze

 Notion



- * 733% performance improvement
- * 8x query runtime reduction
- * Prevents production incidents



“ During incidents, pganalyze gives us the visibility to pinpoint what’s happening. It’s especially helpful for understanding spikes in CPU or query load and distinguishing between root causes and symptoms. ”

– Arka Ganguli, Engineering Manager, Notion



Use pganalyze in the cloud, and on-premise

 **Amazon Web Services**

RDS and Aurora

 **Microsoft Azure**

Flexible Server

 **Google Cloud**

Cloud SQL and AlloyDB

 **Self-Managed**

VM, Container, On-Premise

 **Heroku**

Heroku Postgres

 **Crunchy Data**

Crunchy Bridge

 **Aiven**

Managed PostgreSQL

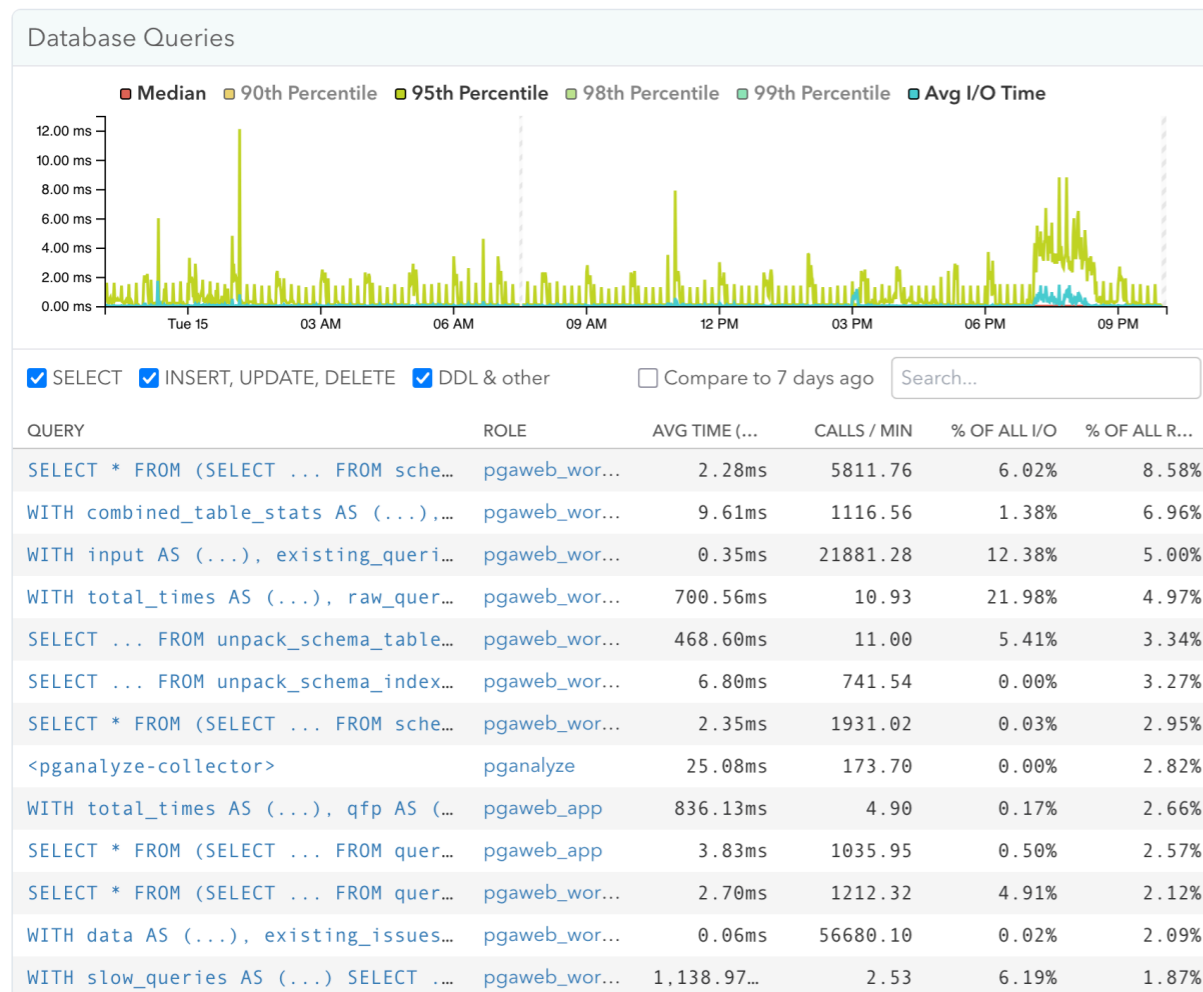
 **pganalyze**



Feature Overview

Query Performance Monitoring

- Captures all queries executed, fast and slow.
- Detailed information for each query, over time.
- Queries are grouped by query fingerprint (powered by pg_query)



Automated EXPLAIN

- Automatically gather query plans for slow executions
- Utilizes auto_explain or slow query log (log_min_duration_statement)
- Best data source for knowing the root cause of a slow outlier

Plan Statistics				
PLAN	EST. COST	AVG RUNTIME	PLAN SAMPLES	PLAN NODES
8788ad7	1,421,278	6,923.74ms	1	Aggregate · Incremental Sort · Merge Join +72 more
d0a9713	499,499	3,673.80ms	1	Aggregate · Incremental Sort · Merge Join +30 more
21aed81	215,971	2,247.24ms	7	Aggregate · Incremental Sort · Merge Join +45 more
e6ae5cc	129,128	1,444.80ms	6	Aggregate · Incremental Sort · Merge Join +29 more
c16b598	84,519	769.49ms	5	Aggregate · Incremental Sort · Merge Join +22 more

Plan Samples (20)							Search plan fingerprint
EXECUTED AT	PLAN	EST. COST	RUNTIME	I/O READ TIME	READ FROM D...	PLAN NODES	
<input type="checkbox"/> 2025-04-15 02:57:55pm PDT	21aed81	215,943	1,495.30ms	1,416.30ms	95%	28.2 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 02:57:18pm PDT	21aed81	215,945	1,442.38ms	1,362.72ms	94%	27.1 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 02:26:52pm PDT	21aed81	216,027	2,538.95ms	2,406.13ms	95%	46.9 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 02:06:43pm PDT	e6ae5cc	129,095	1,178.93ms	1,138.55ms	97%	22.5 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 01:52:46pm PDT	e6ae5cc	129,119	1,580.38ms	1,486.63ms	94%	27.2 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 01:23:31pm PDT	c16b598	84,544	776.30ms	739.35ms	95%	14.4 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 12:04:23pm PDT	e6ae5cc	129,068	747.63ms	546.70ms	73%	10.5 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 11:53:14am PDT	c16b598	84,524	674.02ms	643.26ms	95%	12.6 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 10:56:45am PDT	c16b598	84,554	607.72ms	579.92ms	95%	11.1 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 09:30:51am PDT	c16b598	84,478	603.09ms	576.96ms	96%	11.1 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 09:25:00am PDT	21aed81	215,956	3,499.13ms	3,343.30ms	96%	64.2 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 06:45:43am PDT	21aed81	215,968	3,105.81ms	2,979.14ms	96%	57.3 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 06:00:50am PDT	e6ae5cc	129,151	2,342.47ms	2,285.89ms	98%	23.1 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 05:48:07am PDT	e6ae5cc	129,138	1,403.26ms	1,339.86ms	95%	26.3 MB	Aggregate · Incremei
<input type="checkbox"/> 2025-04-15 05:45:24am PDT	e6ae5cc	129,196	1,416.10ms	1,358.64ms	96%	27.3 MB	Aggregate · Incremei

Node Tree Text JSON Compare Plans

Nested Loop 5

Actual Time: 521.78ms
I/O Time: 0.00ms
Est. Cost: 27,536
Actual Rows: 9,812 · est. 8,100

f(x) Function Scan 6

Actual Time: 499.95ms
I/O Time: 0.00ms
Est. Cost: 100
Actual Rows: 9,812 · est. 10,000

Index Scan (Forward) 7

on public.schema_tables AS st
using schema_tables_pkey

Executed 9812 times:

Metric	Total	Average
Actual Time:	19.62ms	0.002ms
I/O Time:	0.00ms	0.00ms
Est. Cost:	-	3
Actual Rows:	9,812	1 · est. 1

CTE Scan 8

child_table_size

Summary Node

Runtime
1,049.63ms

Read From Disk
0 B

Apr 15 03:43:39pm PDT

Insights (3)

- 3 was estimated to be
- 4 had rows over-estim
- 10 had rows under-esti

Node Fields

- Actual Time
- I/O Time
- Est. Cost
- Actual Rows

Plan Comparison (new!)

- Purpose-built diff mechanism for EXPLAIN plans
- Shows structural differences in the plan shape
- Allows comparison between different per-node metrics across plans

Node Tree Text JSON **Compare Plans**

Plan Comparison [Select plans](#)

Compare: Est. Total Cost (Self) Runtime (Self) I/O Read Time (Self) Rows Buffers (Self) ⓘ

Plan A	Plan B	Plan A	Plan B
2025-04-15 09:52:40pm PDT	2025-04-15 10:42:01am PDT	Runtime	Runtime
-> Merge Join	-> Merge Join	0.00ms	0.00ms
-> Sort	-> Sort	0.25ms	1.20ms
-> Result	-> Result	0.34ms	1.45ms
-> Append	-> Append	0.14ms	0.61ms
-> Index Only Scan ⁸ on query_ov...		0.25ms	
-> Index Only Scan ⁹ on query_ov...		0.05ms	
	-> Index Only Scan ² on query_ov...		0.17ms
	-> Index Only Scan ³ on query_ov...		0.19ms
	-> Index Only Scan ⁴ on query_ov...		0.19ms
	-> Index Only Scan ⁵ on query_ov...		0.19ms
	-> Index Only Scan ⁶ on query_ov...		0.19ms
	-> Index Only Scan ⁷ on query_ov...		0.06ms
-> Materialize	-> Materialize	0.09ms	1.22ms
-> Sort	-> Sort	0.17ms	1.67ms
-> CTE Scan on query_times	-> CTE Scan on query_times	662.28ms	2,712.29ms
CTE query_times			
-> Aggregate	-> Aggregate	1.21ms	12.24ms
-> Sort	-> Sort	0.18ms	1.71ms
-> Nested Loop	-> Nested Loop	59.95ms	299.42ms
-> Index Scan ¹ on queries	-> Index Scan ¹ on queries	0.01ms	0.01ms
-> Function Scan on unpack_query...	-> Function Scan on unpack_query...	600.71ms	2,396.66ms

Summary Node Details Node Source

Plan A: 2025-04-15 09:52:40pm PDT

Seen At	Total Est. Cost	Runtime
Apr 15 09:52pm	486	664.76ms

Plan Fingerprint **Read From Disk** **I/O Read Time**

✚ a787875	0 B	0.00ms
-----------	-----	--------

Plan B: 2025-04-15 10:42:01am PDT

Seen At	Total Est. Cost	Runtime
Apr 15 10:42am	892	2,726.80ms

Plan Fingerprint **Read From Disk** **I/O Read Time**

✚ 3c9c792	0 B	0.00ms
-----------	-----	--------

Index usage

A B Index

- ✓✓ 1. queries_pkey
- ✓ 2. query_overview_stats_35d_20250407_pkey
- ✓ 3. query_overview_stats_35d_20250408_pkey
- ✓ 4. query_overview_stats_35d_20250409_pkey
- ✓ 5. query_overview_stats_35d_20250410_pkey
- ✓ 6. query_overview_stats_35d_20250411_pkey
- ✓ 7. query_overview_stats_35d_20250412_pkey
- ✓ 8. query_overview_stats_35d_20250415_pkey
- ✓ 9. query_overview_stats_35d_20250416_pkey

Plan Statistics (new!)

- Allows tracking of different plans used for a query over time
- Aimed at tracking plan shapes, best used in combination with `auto_explain` for tracking outliers and their execution statistics
- Currently requires AWS Aurora's `aurora_stat_plans` extension
- `pganalyze` will release a new `pg_stat_plans` extension later this year



Query Tuning (new!)

- Allows running EXPLAIN ANALYZE benchmarks through pganalyze
- Explicit testing of different parameter values for the same query, to ensure fixing the query for one customer doesn't regress for others
- Query Tuning Advisor coming later this year that makes recommendations on query rewrites or other ways to fix bad plans

Advisor Testing - Query #43934705 - Table Writes Per Minute

Overview Compare Plans Parameter Sets

+ New Query Variant

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"/>	3318ec3	Baseline	Parameter Set 1	120,161	1,104.48ms
<input type="checkbox"/>	2e9368b	Variant 1 - MATERIALI...	Parameter Set 1	273,823	951.21ms
<input type="checkbox"/>	9af8fd8	Variant 2 - MATERIALI...	Parameter Set 1	529,130	839.23ms
<input type="checkbox"/>	c69a301	Variant 3 - MATERIALI...	Parameter Set 1	421,230	799.88ms

Baseline

```
WITH stats AS (  
SELECT collected_at, next_multi_xact_id, current_xact_id  
FROM postgres_server_stats_35d  
WHERE server_id = $server_id_2  
ORDER BY collected_at DESC  
),  
schema_table_infos AS (  
S...
```

Show full query text

Parameter Sets Show only pa

	PARAMETER SET 1	PARAMETER SET 2	P
\$collected_at	'2025-02-24...	'2025-02-24...	'
\$server_id	'5a4bb1a9-...	'6d8d4af4-5...	'
\$param	NULL	NULL	N
\$id	NULL	NULL	N

Owner
Lukas Fittl

Workbook description
No description

Index Advisor

- Detects missing and unused indexes in your database
- Missing index detection is based on a Constraint Programming model that analyzes all queries on a given table for their WHERE/JOIN conditions and finds the optimal set of indexes
- Currently supports B-Tree and GIST indexes, other types planned

Index Advisor

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

Total Data Size
4.8 TB
▼ 36.5 GB

Total Index Size
1.4 TB

Table Writes
838,345
per minute

Avg. Index Write Overhead
0.35
index bytes per table byte ⓘ

Insights for Database (25)

INSIGHT	NEW LAST 7 DAYS	RESOLVED LAST 7 DAYS
17 Missing Indexes ⓘ Based on query activity in last 7 days	0	1
8 Unused Indexes ⓘ Not recently used (default 35 days)	0	2

Create Index (3) [Copy 3 commands](#)

INDEX	SCAN COST C...	INDEX WRITE ...	AFFECTED...
I10 btree (email)	-3,698	+0.05	2
I6 btree (auth_organization_id)	-3,524	+0.07	2
I11 btree (unconfirmed_email)	-1,849	+0.06	1

Drop Index (0)

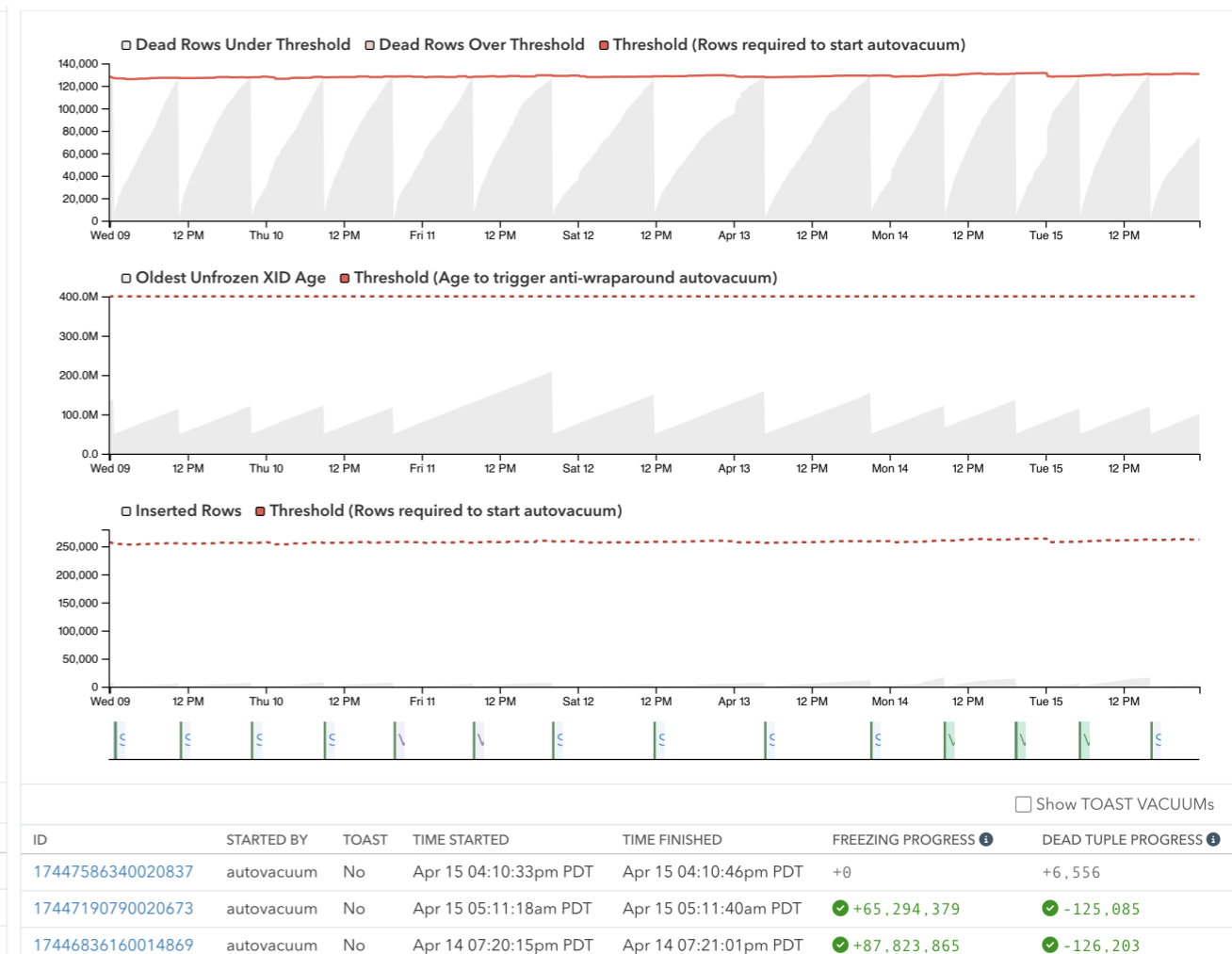
Drop index insights not enabled. Use custom configuration settings to explore index consolidation/removal.

Keep Existing Indexes (5)

INDEX	INDEX WRITE ...	AFFECTED...
I1 btree (id) primary key	0.02	26
I2 btree (auth_uid, auth_provider, COALES... unique	0.07	2
I3 btree (confirmation_token) unique	0.11	1
I4 btree (reset_password_token) unique	0.10	2
I5 btree (COALESCE(auth_organization_id, ... unique	0.04	0

VACUUM Monitoring

- Tracks detailed VACUUM information on a per-table basis, over time
- Can clearly show why Postgres started an autovacuum on a table (whether it was based on dead rows, inserted rows, or TXIDs)
- Helps identify problematic VACUUM behavior, e.g. too few VACUUMs



VACUUM Advisor

- Detects common configuration problems with autovacuum
- Estimates bloat on tables, and notifies you when new bloat occurred and how to tune autovacuum settings to avoid it
- Notifies of blocked VACUUMs & identifies the root cause (e.g. idle TX)

VACUUM Advisor

Overview | Bloat | Freezing | Performance | Activity

Avg. Autovacuum Workers
0.15
out of 4 max workers ⓘ

Total Autovacuum Count
1,160
in the last 24 hours

Total Anti-Wraparound Autovacuum
417 out of 1,160
in the last 24 hours

Insights for Server (0)

Based on table activity in last 7 days

INSIGHT	NEW LAST 7 DAYS	RESOLVED LAST 7 DAYS
Bloat <ul style="list-style-type: none">Insufficient VACUUM FrequencyVACUUM Blocked By Xmin Horizon Learn more in documentation	0	0
Freezing <ul style="list-style-type: none">Approaching TXID WraparoundApproaching MXID Wraparound Learn more in documentation	0	0
Performance <ul style="list-style-type: none">Inefficient Index Phase Learn more in documentation	0	0

Issue #21019: VACUUM: Bloat - Insufficient VACUUM Frequency

Overview

Severity ⓘ Info | Check Frequency 🗓️ Daily | Last Updated 2025-04-08 07:00:51pm PDT | State ✓ Resolved

Description
Insufficient vacuuming is causing avoidable growth on table `public.issue_references`

Guidance

```
ALTER TABLE public.issue_references SET (autovacuum_vacuum_scale_factor = 0.01);
```

[Copy ALTER TABLE command](#)

Avoidable table growth due to insufficient VACUUM frequency was detected. You can adjust the frequency of autovacuum by changing relevant config settings. Lowering `autovacuum_vacuum_threshold` or `autovacuum_vacuum_scale_factor` will increase the frequency of autovacuum in general.

> How to apply changes

> What to check following ALTER TABLE

To confirm the impact of changes, it is also recommended to `run pg_repack` to reclaim disk space. You can check out a graph of estimated bloat over time on the [VACUUM/ANALYZE Activity](#) page.

⚠️ Insufficient VACUUM frequency typically results in table bloat
Table bloat can cause slow queries, increased disk space, memory usage, and increased I/O.
[Learn more in documentation](#)

Recommendation

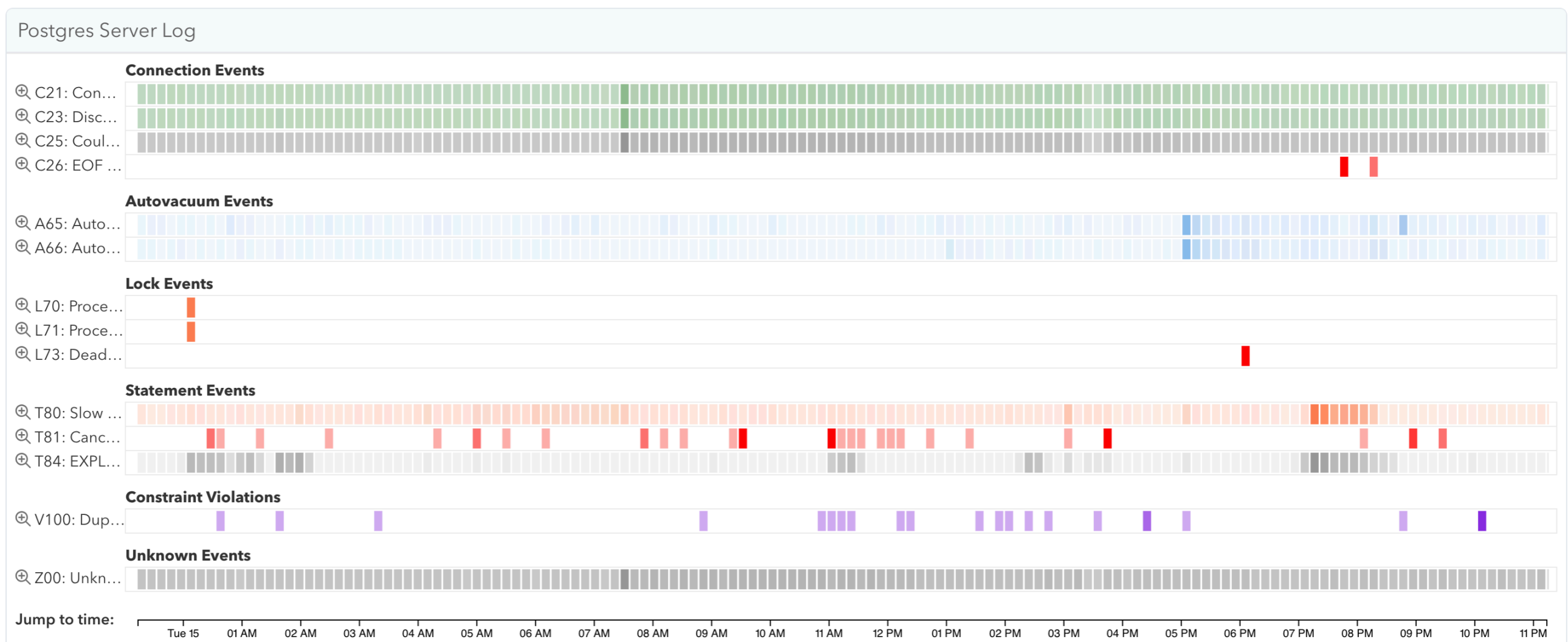
	Current	Recommended
<code>autovacuum_vacuum_threshold</code>	50	
<code>autovacuum_vacuum_scale_factor</code>	0.10	

Estimated Improvement

	Current	Estimated
Total autovacuum count ⓘ	0	
Avoidable growth ⓘ		
Rows	535,638	280,000
Percentage ⓘ	40.1%	20.0%
Bytes ⓘ	113.9 MB	59.0 MB

Log Insights

- Tracks 100+ common Postgres log events with predefined filters
- Can filter out sensitive information from the logs and redact it
- Streams logs continuously from the database for immediate insights on deadlocks, timed out queries, checkpoints and more



HOT (Heap-Only Tuples) Analysis

- HOT = Optimization that makes UPDATES in Postgres significantly cheaper by avoiding VACUUM overhead
- Not possible if any indexes are present on columns (except for BRIN)
- Considered for Index Advisor, and also analyzed for each column

Table: public.issue_references

Statistics Partitions Queries Columns Indexes Index Advisor Constraints VACUUM/ANALYZE Activity VACUUM Simulator

Row Statistics





Average Row Size 217 B Last Analyzed Apr 15 08:04:57pm PDT · 3 hours ago

Columns

NAME	TYPE	NULL % ⓘ	AVG. SIZE ⓘ	NUM. DISTINCT ⓘ	UPDATES / MIN ⓘ	INDEXED? ⓘ	HOT? ⓘ	MODIFIERS
id	uuid	0.00%	16	7,623,002*	-	Yes	-	NOT NULL DEFAULT public.gen_random_uu...
created_at	timestamp ...	0.00%	8	36,530	-	No	-	NOT NULL DEFAULT now()
resolved_at	timestamp ...	37.96%	8	22,166	0.45	Yes	No ⚠	
organization_id	uuid	0.00%	16	913	-	No	-	NOT NULL
server_id	uuid	0.00%	16	2,880	-	No	-	NOT NULL
issue_id	integer	0.00%	4	19,037	-	Yes	-	NOT NULL
referent_type	text	0.00%	8	9	-	Yes	-	NOT NULL
referent_id	text	0.00%	12	1,110,375*	-	Yes	-	NOT NULL
details	jsonb	0.00%	105	18,500	0.00	No	Yes ✓	NOT NULL DEFAULT '{}'::jsonb

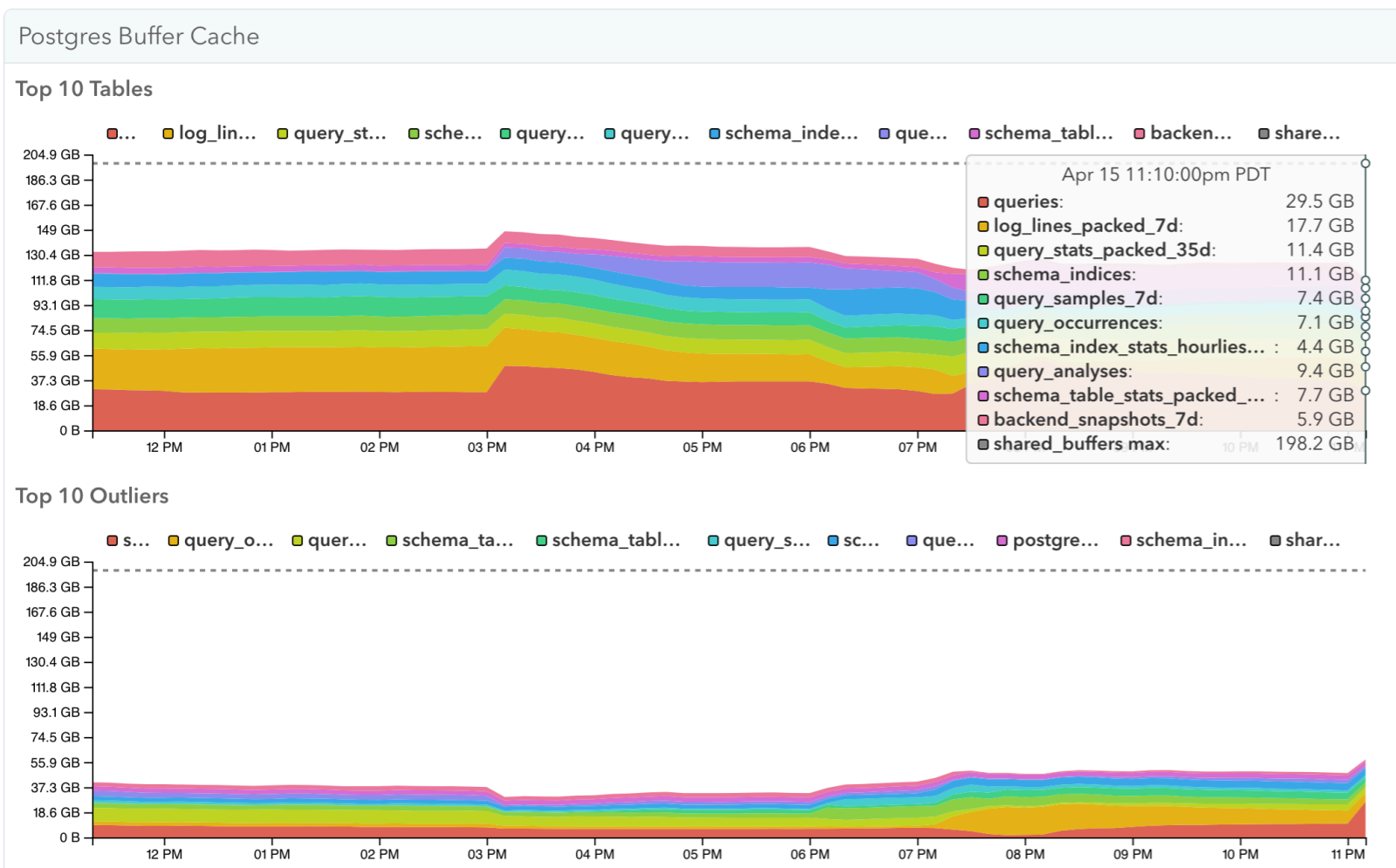
Lock Monitoring

- Part of the Connection Tracing functionality ("Connections" page)
- Automatically analyzes lock graphs in Postgres to find the root query that is blocking activity
- Built-in alerting that notifies if a certain # of queries are blocked

19189	pgaweb_wor...	active (waiting)	<5 seconds	>  3 INSERT INTO backend_snapshots_7d (...) VALUES (...)	Lock / extend
8836	pgaweb_wor...	active	<5 seconds	SELECT query_analyses.* FROM query_analyses WHERE q query_analyses.query_id = \$1 LIMIT \$2	Client / ClientRead
12488	pgaweb_wor...	active	<5 seconds	WITH data AS (...), existing_issues AS (...), stale _issues AS (...), insert_issues AS (...), upda...	Client / ClientRead
6708	pgaweb_wor...	active	<5 seconds	WITH child_tables AS (...), child_table_size AS (...) SELECT ... FROM unpack_schema_table_stats (d...	-
12561	pgaweb_wor...	active	<5 seconds	▼  3 INSERT INTO backend_snapshots_7d (...) VALUES (...) Blocking 12359, 12489, 19189	IO / DataFileExtend
12489	pgaweb_wor...	active (waiting)	<5 seconds	>  2 INSERT INTO backend_snapshots_7d (...) VALUES (...)	Lock / extend
12359	pgaweb_wo...	active (waiti...	<5 seconds	>  1 INSERT INTO backend_snapshots_7d (...) VALUES (...)	Lock / extend

Buffer Cache Monitoring (new!)

- Requires the `pg_buffercache` extension for Postgres
- Automatically analyzes cache contents over time to help identify performance regressions due to cache changes
- Shows breakdown of how much of a table and its indexes are cached



Alerting

- Built-in alerting for common Postgres problems
- Supports email notifications, Slack and PagerDuty
- API available for integrating with other systems
- Additional customization options coming later this year

Alerts & Check-Up

[Summary Report](#) [Configure](#)

Check List

STATUS ▾	CHECK	TRIGGERED ISSUES	ACKNOWLEDGE
Info	Schema Statistics: Unused Indexes	8	0
Info	Query Performance: New Slow Queries	36	0
Info	Index Advisor: Missing Indexes	17	0
Okay	Schema Statistics: Invalid Indexes	0	0
Okay	Config Tuning: Disabled planner features	0	0
Okay	Config Tuning: Disabled fsync	0	0
Okay	Config Tuning: Too small shared_buffers	0	0
Okay	Config Tuning: Disabled stats collection	0	0
Okay	Config Tuning: Too small work_mem	0	0
Okay	System: Out of Disk Space	0	0
Okay	Replication: High Replication Lag	0	0
Okay	Replication: Missing HA Follower	0	0
Okay	VACUUM: Bloat - VACUUM Blocked By Xmin Horizon	0	0
Okay	VACUUM: Freezing - Approaching TXID Wraparound	0	0

 **pganalyze** APP 7:01 PM
💡 [New Issue #21016: New Slow Queries on server "prod-db-main"](#)

Severity

Info

State

Triggered Resolved

Description

Query #43948123 takes 437 ms on average (17992 ms max, 5.41 MB read from disk per call, 108 calls in last 24h)

SQL

```
WITH total_times AS (...), raw_query_data AS (...), raw_query_data_delta AS (...), query_data AS ...
```

[View issue in pganalyze](#)

 [2 replies](#) Last reply 11 days ago

 **pganalyze** APP 7:01 PM
💡 [New Issue #21022: Missing Indexes on server "prod-db-main"](#)

Severity

Info

State

Triggered

Description

Table public.users has 2 missing index insights

[View issue in pganalyze](#)



Demo



Thank you!

Get a free trial of pganalyze

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

Get free pganalyze eBooks and Postgres blog posts

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

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

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