

★ Member-only story

Top 7 PostgreSQL System Views Every Database Admin Should Know for Effective Monitoring



howtouselinux · [Follow](#)

5 min read · Jan 13, 2025

Listen

Share

More



As a database admin, managing a PostgreSQL database can sometimes feel like navigating through a maze.

Whether it's tracking down a slow query, dealing with locking issues, or ensuring smooth replication, the complexity can be overwhelming.

I remember a time when I faced a particularly frustrating issue with monitoring our database's performance, and it felt like every attempt to troubleshoot was a blind guess.

One day, while digging through the logs and trying to optimize some queries, I had an epiphany — **PostgreSQL's system views**.

These powerful, built-in views are like a treasure trove of information, revealing everything from active connections to replication health.

And just like that, what seemed like an unmanageable database became an open book.

In this article, I want to share my journey of discovering the most useful PostgreSQL system views and how they helped me monitor and troubleshoot our database operations more effectively.

Trust me, once you tap into these views, you'll feel like you have superpowers for your PostgreSQL management!

Why You Should Care About PostgreSQL System Views

PostgreSQL, being a robust relational database management system (RDBMS), offers system views that provide deep insights into the database's inner workings.

These views give you real-time data on everything from resource usage and locks to query performance, helping you diagnose issues and optimize performance. And the best part? They're available by default, no complex setup required.

Let me take you through some of the most useful system views that transformed how I monitor and troubleshoot PostgreSQL.

1. **pg_stat_activity**: Tracking Active Connections

The first system view that truly changed the game for me was **pg_stat_activity**. I'd been struggling to figure out which queries were hogging resources and causing

slowdowns. Then I realized that this view shows the activity of every active connection. It was like I was given a magnifying glass to peer into every ongoing process in the database.

Key Insight: You can see the queries that are actively running, which helps you identify long-running or problematic ones.

Example Query:

```
SELECT pid, username, application_name, client_addr, state, query
FROM pg_stat_activity
WHERE state = 'active';
```

This simple query helped me spot exactly which queries were taking up too much time, and I was able to optimize them.

2. pg_stat_user_tables: Understanding Table Usage

Have you ever wondered how well your tables are performing? I did. After a while, I noticed that certain tables seemed sluggish, and others weren't being queried as often as I thought they should be. That's when I turned to **pg_stat_user_tables**. This view gave me detailed statistics on table usage, including how many rows were read, updated, or deleted.

Key Insight: This view helps you identify tables that are being under-utilized, which could be a sign that you need better indexing or optimization.

Example Query:

```
SELECT relname, seq_scan, idx_scan, n_tup_ins, n_tup_upd, n_tup_del
FROM pg_stat_user_tables;
```

With this query, I was able to see which tables needed a little more TLC in terms of indexing.

3. pg_locks: Identifying Lock Contention

Locking issues are one of the most common problems that plague databases, and I had my fair share. I kept noticing that some queries were taking a long time to complete, but couldn't figure out why. It turns out, it was a **locking issue** — transactions were waiting for locks to be released.

That's when I turned to **pg_locks**. This view helped me monitor the current locks in the system and identify where transactions were getting blocked.

Key Insight: You can quickly identify lock contention and resolve performance bottlenecks caused by excessive locking.

Example Query:

```
SELECT pid, locktype, relation::regclass, mode, granted
FROM pg_locks
WHERE NOT granted;
```

This query helped me identify which locks were holding up transactions, allowing me to address the underlying issues and speed up database performance.

4. pg_stat_database: Monitoring Database-Level Health

For a broader view, **pg_stat_database** came in handy. This view gave me general statistics for each database, such as the number of transactions, deadlocks, and I/O operations. It was essential when I wanted to understand the health of the entire database.

Key Insight: This view helps you get a sense of overall database performance and quickly spot issues like deadlocks or high I/O operations.

Example Query:

```
SELECT datname, xact_commit, xact_rollback, blks_read, blks_hit, deadlocks
FROM pg_stat_database;
```

With this query, I got a clear picture of database activity and performance, helping me prioritize where to focus my optimization efforts.

5. pg_stat_replication: Ensuring Replication Health

In a setup where replication was critical, I used **pg_stat_replication** to monitor replication status. Whether it was streaming or logical replication, this view showed me the replication lag and whether any standby servers were falling behind.

Key Insight: Monitoring replication is essential for ensuring data consistency across servers and avoiding sync issues.

Example Query:

```
SELECT pid, application_name, state, write_lag, sync_state
FROM pg_stat_replication;
```

This query helped me track replication delays, ensuring our standby servers were up-to-date and that there were no data synchronization issues.

6. pg_user: Managing User Roles and Permissions

When it came to managing users and their permissions, **pg_user** was a lifesaver. I needed to audit who had superuser privileges, who could create databases, and who had replication rights. This view gave me a complete overview of the user roles in the database.

Key Insight: This view is invaluable for auditing user roles and ensuring the security of your PostgreSQL environment.

Example Query:

```
SELECT username, usesysid, usecreatedb, usesuper, userepl, usebypassrsls  
FROM pg_user;
```

With this query, I easily identified users who had more privileges than they should, helping me tighten security in the system.

7. pg_class: Understanding Database Objects

Finally, I turned to `pg_class` to get a clear view of the structure of the database itself. This view provided metadata about tables, views, and indexes — essential for anyone who needs to manage database objects.

Key Insight: Use this view to quickly get metadata about your relations (tables, indexes, etc.) and monitor their size and performance.

Example Query:

```
SELECT relname, relkind, relnamespace::regnamespace, relowner::regrole, reltupl  
FROM pg_class;
```

This query gave me a detailed overview of every object in the database, helping me identify large tables or indexes that needed optimization.

Conclusion: PostgreSQL System Views Are a Game Changer

PostgreSQL's system views are powerful tools that I now rely on regularly. Whether I'm monitoring active queries, tracking locks, or auditing users, these views give me the insights I need to keep our database running smoothly.

If you're not already using them, I highly recommend you start exploring. They can make your job as a DBA or developer a lot easier and far more efficient.

By leveraging these system views, you can catch potential issues early, optimize performance, and ensure your PostgreSQL database is running at its best.

Database

AI

Postgresql

Data Science



Follow

Written by howtouselinux

1.1K Followers · 17 Following

We bring real-world experience and DevOps tips here. Linux training

Responses (2)



Gvadakte

What are your thoughts?



Rajan Sahu

Jan 16



This is a great article for understanding the insight stats of a query. Thank you for sharing!



10

[Reply](#)



Dan Toomey

Mar 25



It would be nice to have sample output where the reader could understand how you decided to make a change

[Reply](#)

More from howtouselinux



In Level Up Coding by howtouselinux

Using Curl command Every Day? Discover Advanced Features to Simplify Your Workflow!

If you're a curl user, you're likely familiar with basic commands like `curl -X GET` or `curl -X POST`.



Mar 24



315



6





NetworkManager

A tool for managing network connections in various Linux distributions.



systemd-networkd

A system service to manage network configurations in Linux.



Netplan

A utility for configuring network settings in Ubuntu and similar

 In Level Up Coding by howtouselinux

Never Get Confused About Your Linux Network Setup Again: Learn NetworkManager, systemd-networkd...

My journey with Linux networking began years ago. At first, I thought learning ifconfig or ip commands would be enough to handle any network...

★ Mar 12 🖱️ 151 💬 1



Advanced
Techniques



Starting
Services



Open in app ↗

Medium

🔍 Search



Checking

Stopping

 In Level Up Coding by howtouselinux

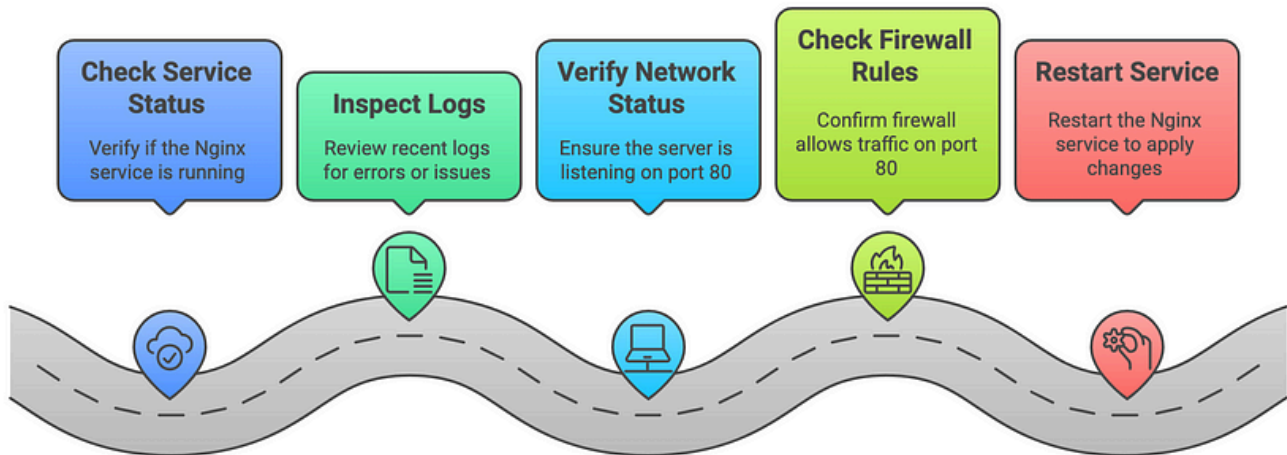
Stop Using systemctl Blindly: Master Advanced Service Management Techniques!

If you're a Linux user, you're likely familiar with systemctl.

★ Jan 31 🖱 549 💬 4



Troubleshooting Nginx Web Server



 In Level Up Coding by howtouselinux

Visual Guide to Troubleshooting Linux: Commands & Strategies

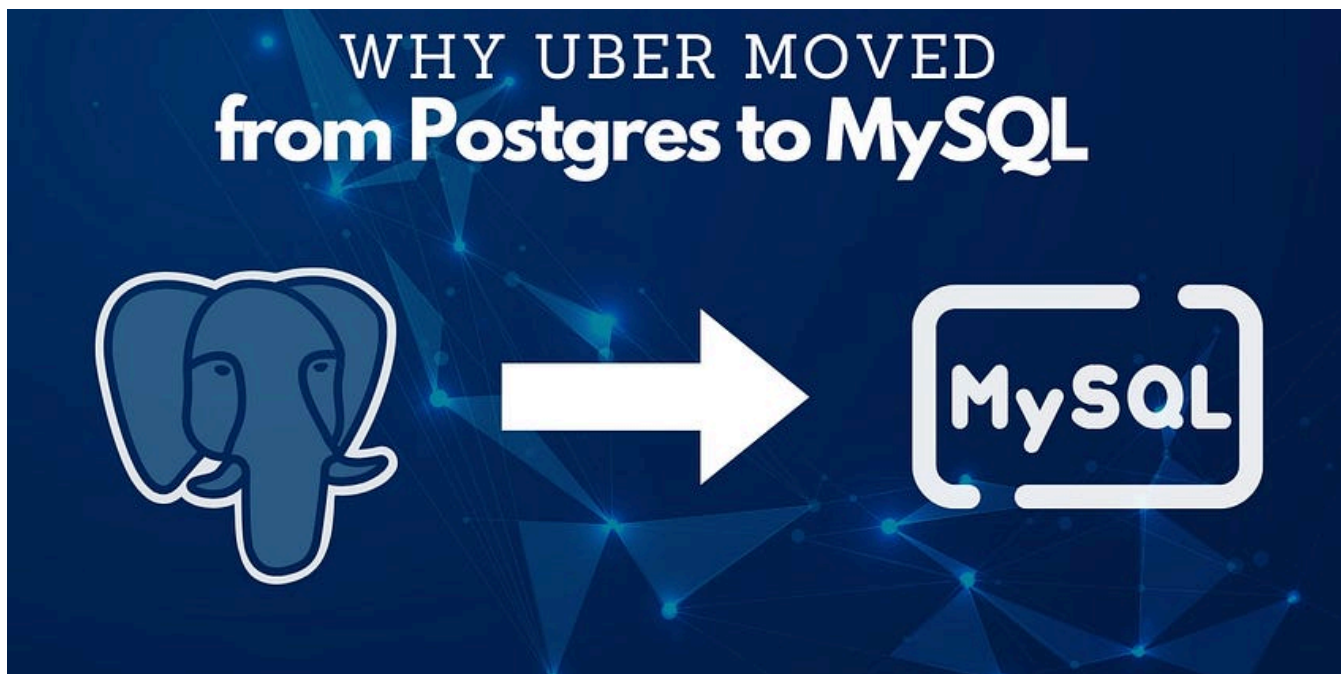
Ever struggled with a system crash? Wondered why a service won't start? Or found yourself stuck debugging weird network problems?

★ Mar 7 🖱 105 💬 3



See all from howtouselinux

Recommended from Medium

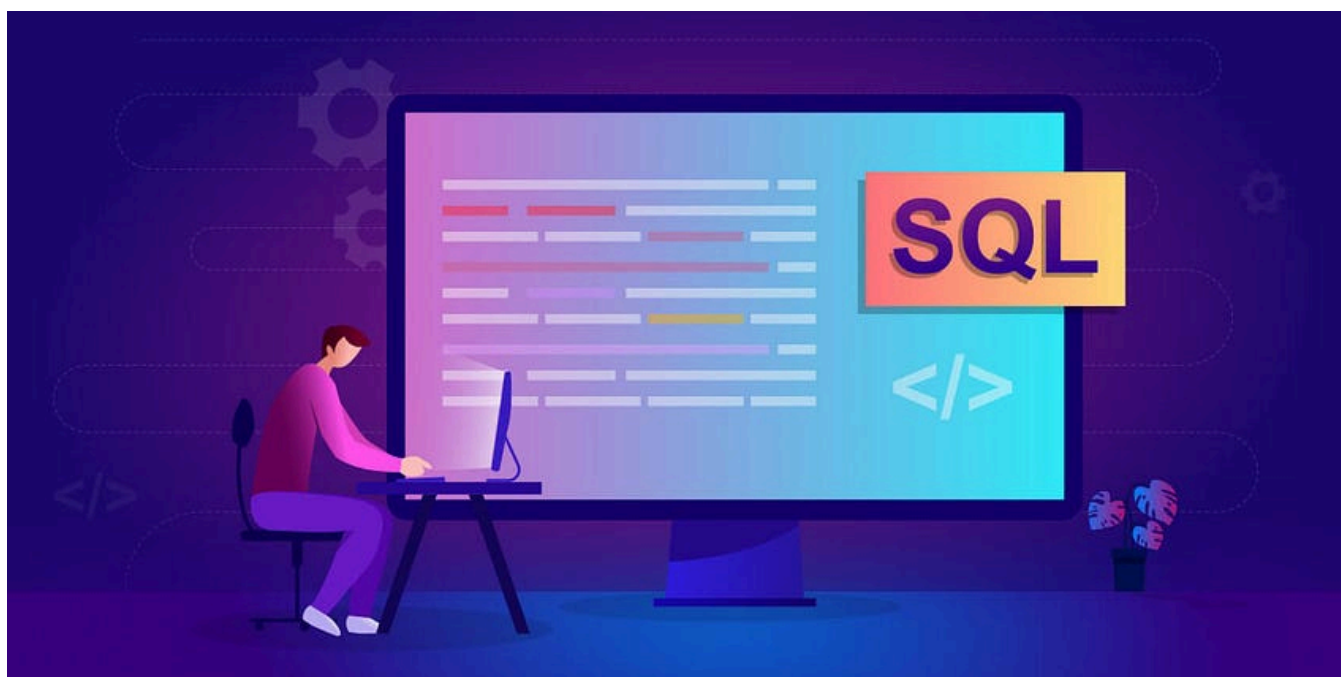


 In Databases by Sergey Egorenkov

Why Uber Moved from Postgres to MySQL

How PostgreSQL's architecture clashed with Uber's scale—and why MySQL offered a better path forward

Mar 29  228  7



 In Hack the Stack by Coders Stop

9 Database Optimization Tricks SQL Experts Are Hiding From You

Most developers learn enough SQL to get by—SELECT, INSERT, UPDATE, DELETE, and maybe a few JOINS. They might even know how to create...

★ Mar 27 🖱 185 💬 5



SaaS Killer

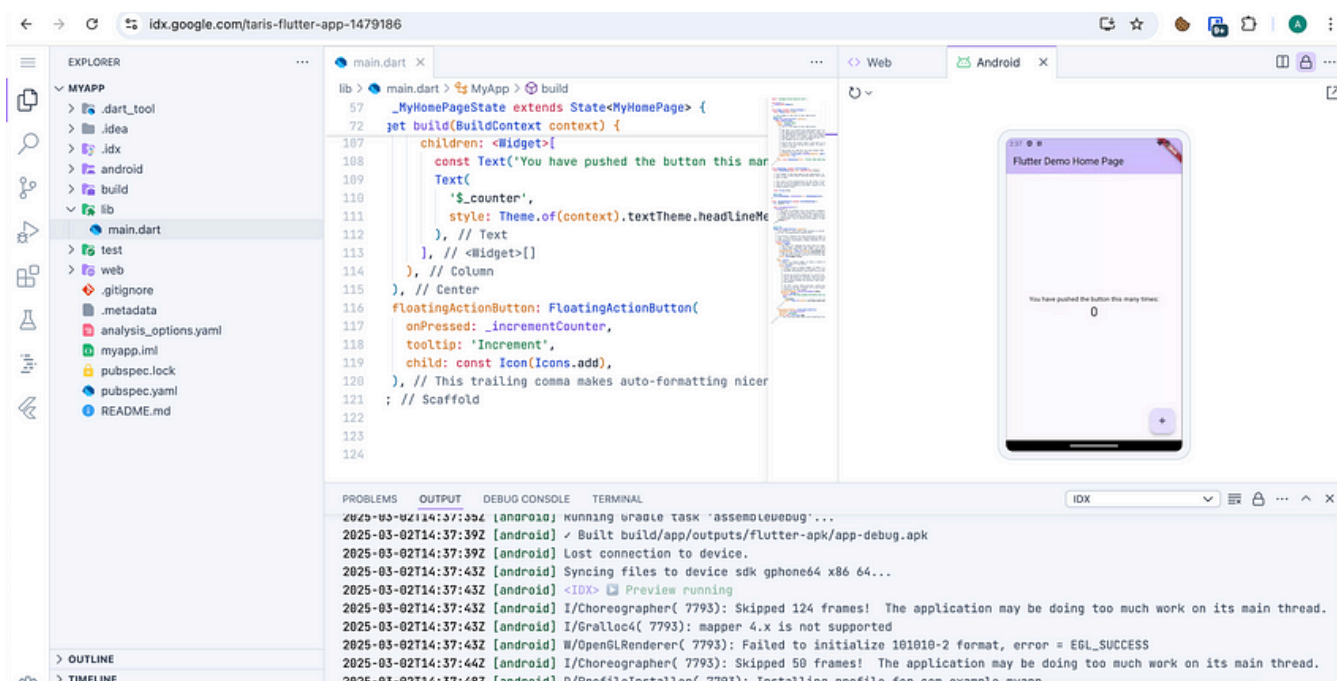


 Dipanshu

Paying for software is stupid ..Open-Source tools to Destroy Your SaaS Expenses

These 40 Open-Source Tools Will Make Your SaaS Subscriptions Look Obsolete

★ Mar 27 🖱 2K 💬 26



 In Coding Beauty by Tari Ibaba

This new IDE from Google is an absolute game changer

This new IDE from Google is seriously revolutionary.

★ Mar 12 🖱️ 3.7K 💬 200



 Utsav Madaan


5 Free & Open-Source Tools That Are Total Game Changers for Developers in 2025 🚀

Tired of the same old dev tools? 😴 Discover 5 awesome free and open-source gems that are shaking things up in 2025!

Mar 29 🖱️ 139 💬 3



What it Means	Best Used For
Store directly in the row	Simple data like INT
Store in the row (unless large)	Larger types, but try
Compress + store out-of-row	Long texts, large ob
Store out-of-row, no compression	When compression

 Udbhav Singh

Storages in PostgreSQL

What Are Storage Types in PostgreSQL — And Why Should You Care?

6d ago  18

See more recommendations