

★ Member-only story

# The PostgreSQL Performance Playbook: Solving Slow Queries with Simple Steps



howtouselinux · [Follow](#)

6 min read · Jan 18, 2025

Listen

Share

More



It was a typical Tuesday morning when the first sign of trouble came through.

I checked my inbox, and there it was: a flood of user complaints about slow page loads. We had just released a new feature, and now our application was crawling.

As a db owner who had only recently started diving into PostgreSQL, I knew the issue might be related to our database, but I had no idea where to start.

Slowly, I began to understand what was happening: there were inefficiencies in how the database was processing queries, and I needed to fix them.

By the end of that day, I had successfully optimized those slow queries, and the app performance improved dramatically.

Now, I'm here to share what I learned and guide you through the process of diagnosing and fixing slow queries in PostgreSQL.

## **Slow Queries: What's the Big Deal?**

In the world of web applications, speed is everything. Slow queries in your PostgreSQL database can lead to sluggish page loads, frustrated users, and even lost revenue.

Whether you're working on a small app or managing a large-scale system, optimizing your queries is crucial for keeping your app responsive and efficient.

But the good news is, slow queries don't have to be a mystery. With the right tools and techniques, you can easily pinpoint the problem areas and improve performance.

This guide will walk you through the steps of diagnosing slow queries, finding the bottlenecks, and applying strategies to make your queries run faster.

By the end, you'll have a solid toolkit for optimizing your PostgreSQL queries and keeping your database running smoothly. So, let's dive in!

## **Step 1: Identifying Long-Running Queries**

Before you can fix slow queries, you need to identify them. The most effective way to start is by pinpointing which queries are taking too long.

### **Query to Identify Slow Queries**

Start by running a simple query to find queries that have been running for more than 5 minutes. The following query will give you the necessary information:

```
SELECT
  pid,
  now() - pg_stat_activity.query_start AS duration,
  query,
  state
FROM pg_stat_activity
WHERE (now() - pg_stat_activity.query_start) > interval '5 minutes';
```

This will give you the `pid`, duration, the actual query, and its state (active or idle).

**Queries that are idle don't need your attention, but active queries might be the source of your database slowness.**

You can also run another query to identify long-running active queries sorted by execution time:

```
SELECT current_timestamp - query_start AS runtime, datname, username, query
FROM pg_stat_activity
WHERE state = 'active'
ORDER BY runtime DESC;
```

This query will help you spot queries that are monopolizing system resources. Once you identify the slow queries, you can start troubleshooting further.

## Step 2: Analyzing Table Statistics and Query Output

Sometimes, slow performance isn't because of poorly written queries but because PostgreSQL is making inefficient decisions. This can be due to outdated or missing statistics.

To update your table statistics and give PostgreSQL a better understanding of your data, run the following command:

```
ANALYZE my_table;
```

Next, verify whether your query is pulling too much data. If your query is returning a large result set, it could be contributing to the slowdown. Try adding a `LIMIT` clause to restrict the result size:

```
SELECT * FROM my_table LIMIT 100;
```

Another good practice is to run the slow query in isolation. If other queries are running simultaneously, they could be competing for CPU or memory, causing your slow query to run even slower.

### Step 3: Investigating Common Performance Bottlenecks

If the basic troubleshooting steps haven't improved performance, it's time to dig deeper.

Common bottlenecks in PostgreSQL include resource utilization, caching, and table/index bloat.

#### System Resource Utilization

Monitor the system's CPU, memory, and disk I/O. An overloaded system can have a direct impact on query performance.

Tools like `htop`, `iostat`, or PostgreSQL's built-in `pg_stat_activity` view can help identify where your system might be under strain.

#### The Caching Effect

PostgreSQL uses caching to speed up frequently executed queries. If your query is executed often, caching can improve its performance.

To investigate caching, you can use the `pg_stat_statements` extension to get detailed statistics on cache hits and reads:

```
SELECT * FROM pg_stat_statements;
```

## Table and Index Bloat

Over time, tables and indexes accumulate “dead” rows due to updates and deletes. This bloat increases the I/O required to execute queries. To check for potential bloat, run this query:

```
SELECT pg_relation_size(relid) AS tablesize, schemaname, relname, n_live_tup
FROM pg_stat_user_tables
WHERE relname = 'your_table_name';
```

If you notice a significant difference between the table size and live tuples, it's time to run a `VACUUM` or `REINDEX` operation to reclaim space.

## Step 4: Using PostgreSQL Extensions for Advanced Analysis

If you're still struggling to find the cause of slow queries, PostgreSQL offers several extensions for advanced analysis.

`pg_stat_statements`

This extension provides detailed statistics about query execution, including execution count, total execution time, and I/O statistics. To install it, run:

```
CREATE EXTENSION pg_stat_statements;
```

This will allow you to track which queries are consuming the most resources.

`pg_stat_activity`

The `pg_stat_activity` view gives you real-time information on active queries, their start time, state, and execution duration. It's a useful tool for pinpointing queries that might be bottlenecking your system.

### pg\_statviz **Extension**

If you prefer to visualize your PostgreSQL performance metrics, `pg_statviz` can be a lifesaver.

It provides visualizations of I/O operations, cache hits, and lock statistics, helping you understand performance trends over time.

```
CREATE EXTENSION pg_statviz;
```

### pg\_buffercache

For a deep dive into your buffer cache, use the `pg_buffercache` extension. This extension helps identify whether your queries are frequently reading from disk or benefiting from cached data.

```
CREATE EXTENSION pg_buffercache;
```

## Step 5: Optimizing Queries

Once you've identified the root cause of slow queries, it's time to optimize them. Below are several strategies you can apply to improve query performance.

### Revisit Query Design

Start by revisiting your query design. Inefficient joins, unnecessary subqueries, or bad filtering conditions often result in slow queries.

Try to simplify the query where possible. Avoid unnecessary subqueries and joins, and only select the data you really need.

### Index Optimization

Indexes are crucial for speeding up queries. Ensure that your frequently queried columns — especially those used in `WHERE` clauses and joins—are indexed. For example:

```
CREATE INDEX idx_column_name ON my_table(column_name);
```

Use `EXPLAIN` to verify that your indexes are being used effectively.

### Database Configuration Tuning

Tuning PostgreSQL's configuration parameters can make a significant difference. For example, adjusting the `work_mem` setting allows PostgreSQL to allocate more memory for sorting and hashing operations. You can set this on a per-session basis:

```
SET work_mem TO '256MB';
```

### Regular Maintenance

[Open in app](#) ↗

Medium

 Search



```
VACUUM ANALYZE my_table;
```

## Advanced Query Optimization Techniques

### Query Rewrite

If a query is inherently slow, consider rewriting it. For example, breaking complex queries into smaller parts or reordering joins can sometimes drastically improve performance.

### Common Table Expressions (CTEs)

While CTEs can improve query readability, they may lead to performance issues due to repeated scanning of the same data. Test your queries with and without CTEs to see if they have an impact on performance.

```
WITH recent_orders AS (  
    SELECT order_id, customer_id, order_date  
    FROM orders  
    WHERE order_date > '2024-01-01'  
)  
SELECT * FROM recent_orders WHERE customer_id = 123;
```

### Use `EXPLAIN ANALYZE`

Always use `EXPLAIN ANALYZE` to check the actual execution plan of your queries. This helps you identify which part of the query is the bottleneck.

```
EXPLAIN ANALYZE SELECT * FROM my_table WHERE column_name = 'value';
```

## Conclusion: Maintaining High-Performance Queries in PostgreSQL

Slow queries are an inevitable part of working with databases, but with the right tools and techniques, you can keep them under control.

By identifying the root cause of slow queries, optimizing query design, tuning system settings, and leveraging PostgreSQL extensions, you can ensure that your database performs optimally.

Remember, optimizing queries is not a one-time task. Regular monitoring, maintenance, and continuous improvement are key to keeping your PostgreSQL database running smoothly.

If I could go back to that frustrating Tuesday morning, I'd be much better prepared now. Hopefully, with these strategies, you'll be ready to face your own slow query challenges head-on.

[Postgresql](#)[Sql](#)[Data](#)[Database](#)



[Follow](#)

## Written by howtouselinux

1.1K Followers · 17 Following

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

### No responses yet



Gvadakte

What are your thoughts?

### 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

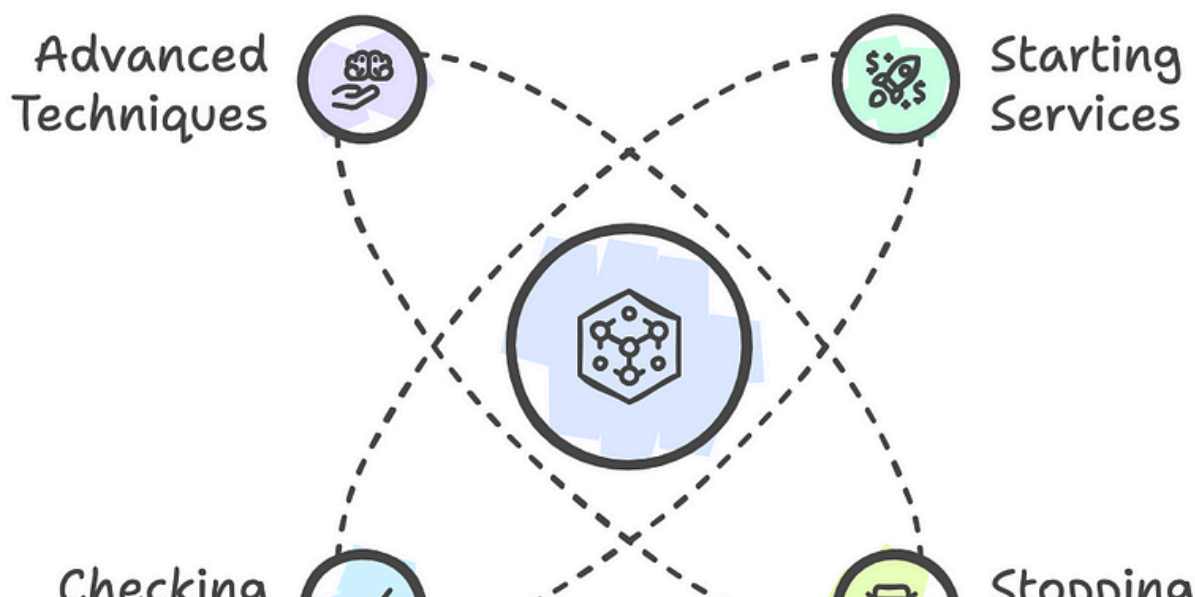


151



1





 In Level Up Coding by howtouselinux

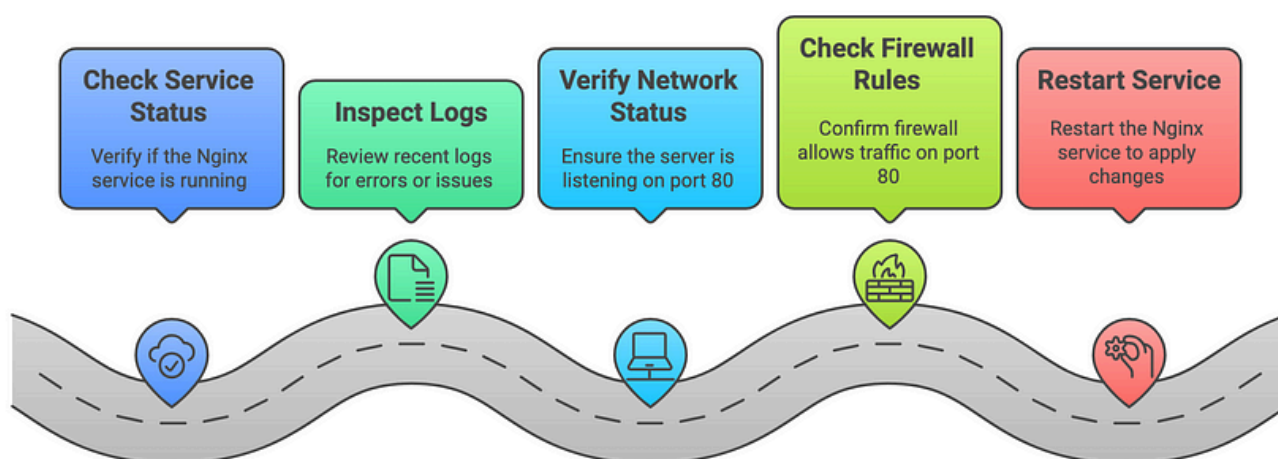
## Stop Using systemctl Blindly: Master Advanced Service Management Techniques!


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



# SaaS Killer

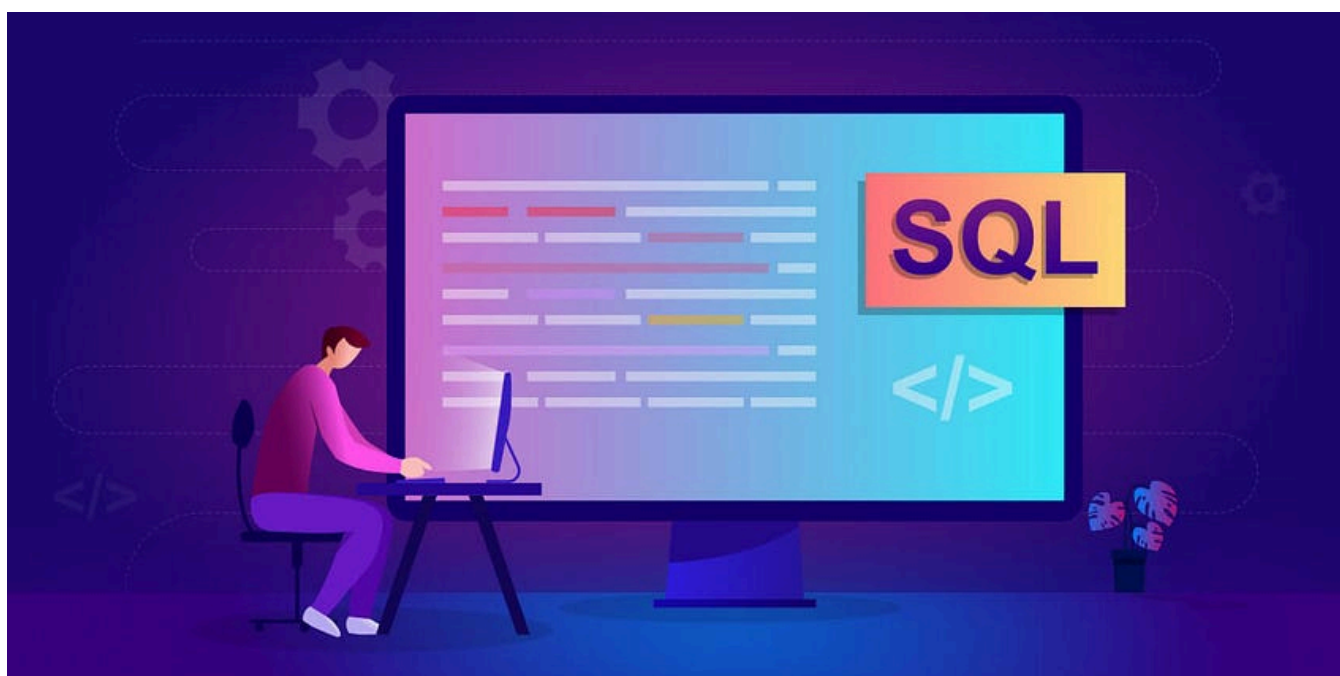


 Dipanshu

## 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 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

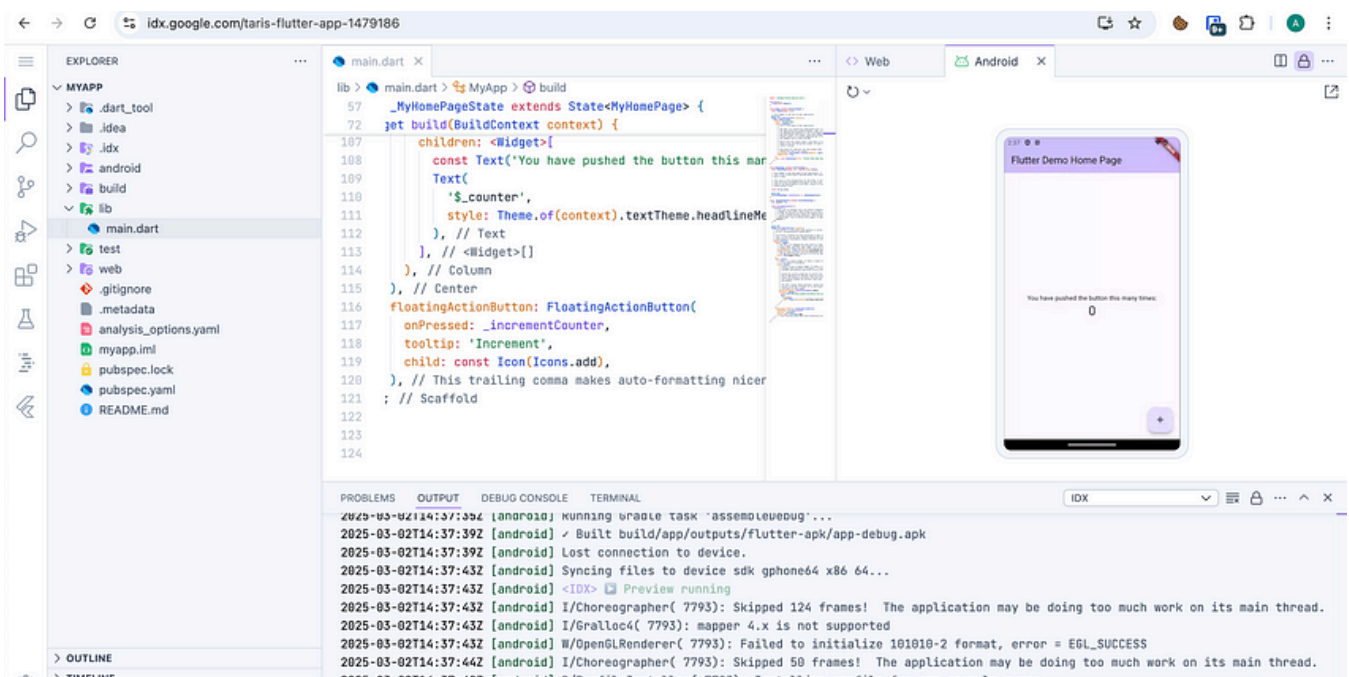


Utsav Madaan

## 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





In Coding Beauty by Tari Ibaba

## This new IDE from Google is an absolute game changer

This new IDE from Google is seriously revolutionary.



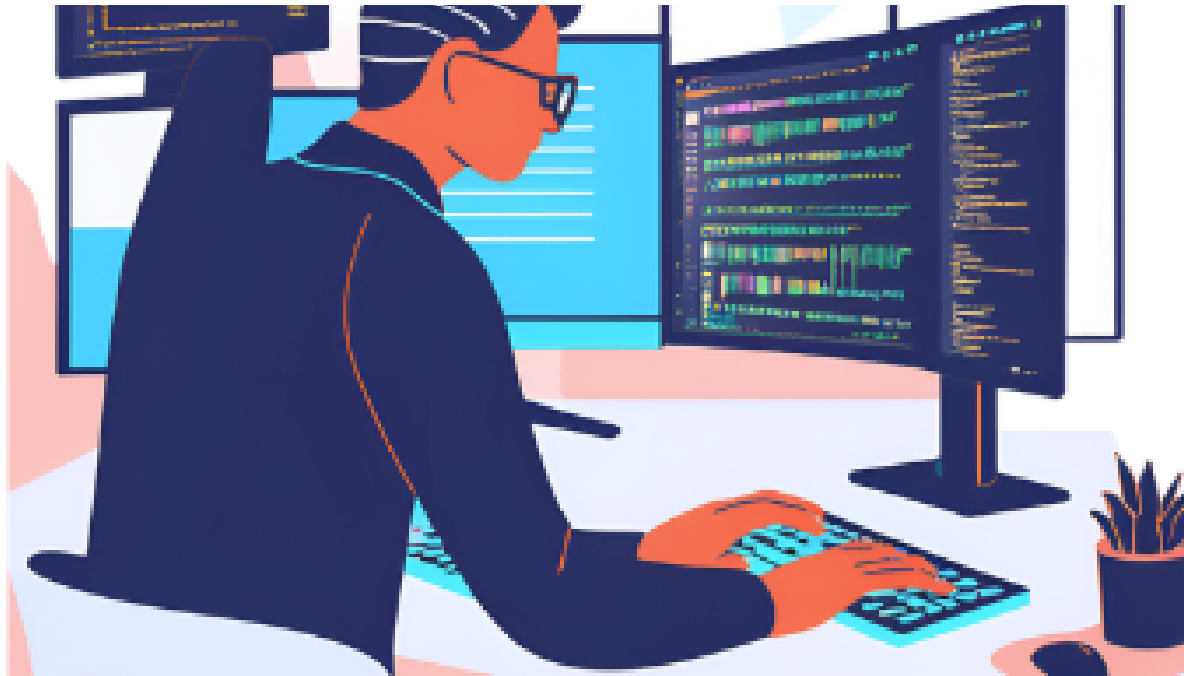
Mar 12



3.7K



200



Anil R

## Advanced Java Performance Optimization and Concurrency Techniques: A Comprehensive Guide

Introduction



Mar 26



31



2

[See more recommendations](#)