

★ Member-only story



Nidhi Gupta · [Follow](#)

4 min read · Oct 19, 2024



Listen



Share



More

Ways to Optimize and Improve PostgreSQL Performance



Photo by [Carl Heyerdahl](#) on [Unsplash](#)

PostgreSQL is a powerful and feature-rich open-source relational database system that can efficiently handle large volumes of data. However, like any system, it requires careful configuration and optimization to perform at its best, especially as data grows or workload increases. Here are the key strategies and techniques to optimize PostgreSQL performance.

1. Hardware and System Configuration

a. Memory Configuration

PostgreSQL relies heavily on memory for caching data and query execution. Make sure your system has enough memory to handle the workload.

shared_buffers : This is one of the most critical parameters. It controls how much memory PostgreSQL uses for its cache. Setting this too low will cause more disk access, slowing the database. The recommended starting point is 25% of the system's memory.

```
shared_buffers = 25% of total system memory
```

work_mem : This parameter controls memory for operations like sorting and hashing. It should be increased for complex queries involving sorting or joining large tables.

```
work_mem = 1-4 MB per connection
```

effective_cache_size : This estimates how much memory is available for caching data files by the OS and PostgreSQL. A higher value here helps PostgreSQL's planner make better choices for query execution.

```
effective_cache_size = 50-75% of system memory
```

b. Disk I/O

Disk I/O is a common bottleneck for database performance. Using faster storage, such as SSDs, can dramatically improve performance, especially for write-heavy workloads.

- **Use RAID with SSD: RAID 10 (striping with mirroring) is a great choice for balancing performance and reliability.**
- **Tune `checkpoint_segments` and `checkpoint_completion_target` :** Checkpoints in PostgreSQL are points at which all modified buffers are written to disk. Setting these too low can cause excessive I/O.

c. CPU Considerations

For CPU-bound queries (complex analytical queries), CPU speed and the number of cores are important. Parallel query execution, introduced in PostgreSQL 9.6, can leverage multiple cores for faster execution. Setting appropriate values for `max_parallel_workers_per_gather` and `max_parallel_workers` can improve performance.

2. Database Configuration

a. Autovacuum Tuning

PostgreSQL uses a background process called auto vacuum to reclaim storage from deleted rows and update statistics. However, improper auto vacuum settings can degrade performance.

Increase `autovacuum_vacuum_cost_limit` and `autovacuum_vacuum_scale_factor` : Larger databases may require more aggressive auto vacuum settings. Adjust these parameters based on table sizes and activity.

```
autovacuum_vacuum_cost_limit = 2000  
autovacuum_vacuum_scale_factor = 0.02
```

b. Connection Pooling

Too many active connections can lead to contention for resources, especially on a busy system. Using connection pooling tools like **PgBouncer** can significantly improve performance by limiting the number of open connections and reusing them efficiently.

c. WAL (Write-Ahead Logging) Tuning

The WAL log is critical to PostgreSQL's durability and recovery mechanism, but it can also affect performance.

wal_buffers : Increase this value to store more WAL data in memory before it is written to disk. For high-write environments, this can reduce I/O pressure.

```
wal_buffers = 16MB
```

synchronous_commit : For less critical transactions, setting this to **off** can improve performance by reducing the wait time for WAL data to be written to disk.

```
synchronous_commit = off
```

3. Indexing

a. Proper Indexing

Indexes are crucial for fast query execution, but too many indexes or poorly chosen indexes can degrade performance. Analyze your workload and create indexes where necessary, especially for frequently queried columns.

B-tree Index: The default index type, best for equality and range queries.

```
CREATE INDEX idx_example ON table_name (column_name);
```

```
CREATE INDEX gin_idx ON table_name USING gin(column_name);
```

b. Index Maintenance

Indexes need maintenance to remain effective, especially in a write-heavy environment. Running `VACUUM` and `ANALYZE` commands regularly help in cleaning up bloated indexes and keeping statistics updated.

4. Query Optimization

a. Use `EXPLAIN` and `EXPLAIN ANALYZE`

Always use `EXPLAIN` or `EXPLAIN ANALYZE` to analyze query performance and execution plans. Look for potential issues like sequential scans on large tables, which indicate missing indexes.

```
EXPLAIN ANALYZE SELECT * FROM table WHERE condition;
```

b. Avoid N+1 Queries

Instead of running multiple queries inside loops, use joins and subqueries to fetch data in fewer queries. This reduces the round-trips between the application and the database.

c. Partitioning Large Tables

Partitioning can improve performance by splitting large tables into smaller, more manageable pieces. This is particularly useful for time-series data or large datasets where certain columns can be used to split the data.

```
CREATE TABLE partitioned_table (id SERIAL, data TEXT, created_at DATE) PARTITIONED BY (created_at);
```

d. Caching Results

If the same queries are being executed frequently, consider using caching mechanisms such as `pg_stat_statements` or application-level caching.

5. Monitoring and Maintenance

a. Track Slow Queries

PostgreSQL logs slow queries that exceed a certain threshold, which you can set using `log_min_duration_statement`. Use this log to identify and optimize problematic queries.

```
log_min_duration_statement = 1000 # Logs queries taking longer than 1 second
```

b. Monitoring Tools

Use tools like `pg_stat_activity`, `pg_stat_statements`, and external tools like `pgAdmin`, `pgHero`, or `pg_top` to monitor system performance and query execution.

6. Upgrading PostgreSQL

Lastly, consider upgrading to the latest version of PostgreSQL. Each new release includes performance improvements and new features that can optimize your workload without additional configuration.

Optimizing PostgreSQL performance requires a combination of proper hardware configuration, fine-tuning database settings, intelligent indexing, query optimization, and monitoring. By applying these strategies, you can significantly improve the speed, scalability, and efficiency of your PostgreSQL database.

Thanks for the read 🙏🙏. Do clap 🙌🙌 if find it useful 😊.

“Keep learning and keep sharing knowledge”

[Postgres](#)[Postgresql](#)[Database](#)[Women In Tech](#)[Women](#)[Follow](#)

Written by Nidhi Gupta

1.8K Followers · 574 Following

Azure Data Engineer 🧑‍💻.Heading towards cloud technologies expertise 🙌.

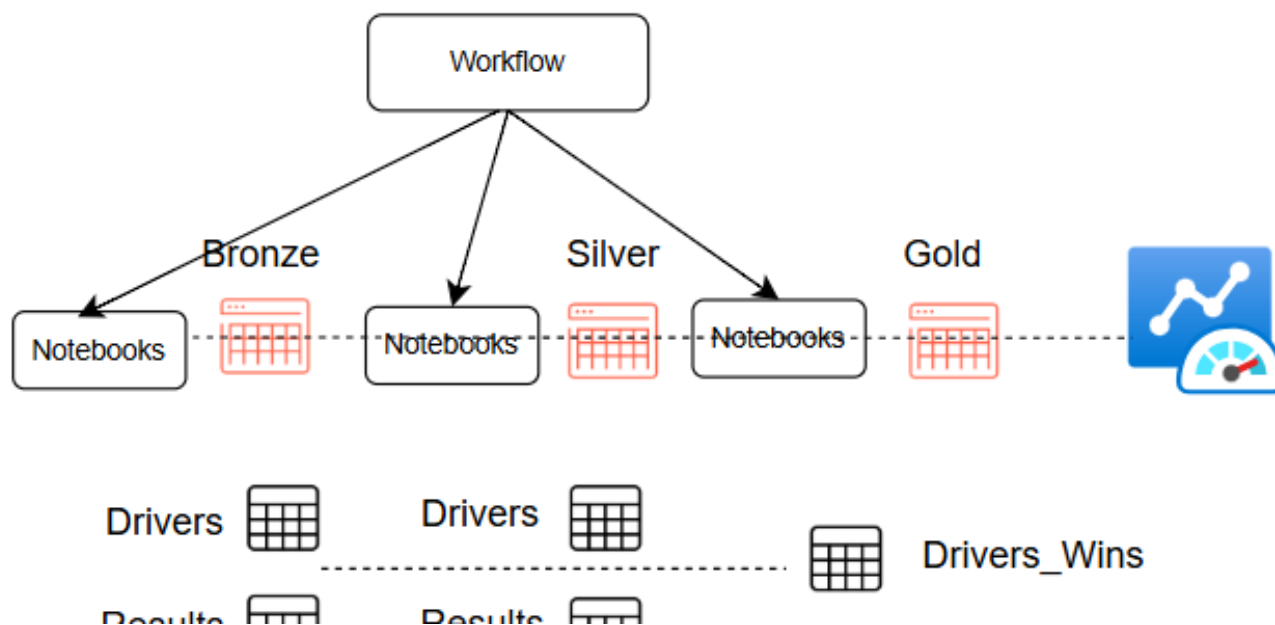
No responses yet



Gvadakte

What are your thoughts?

More from Nidhi Gupta



Nidhi Gupta

Implementing Unity Catalog with Medallion Architecture: A Mini Project

Project Description: Enable a Databricks workspace with Unity Catalog for centralized data governance and access control. Implement a...

★ Feb 16 🖱 71

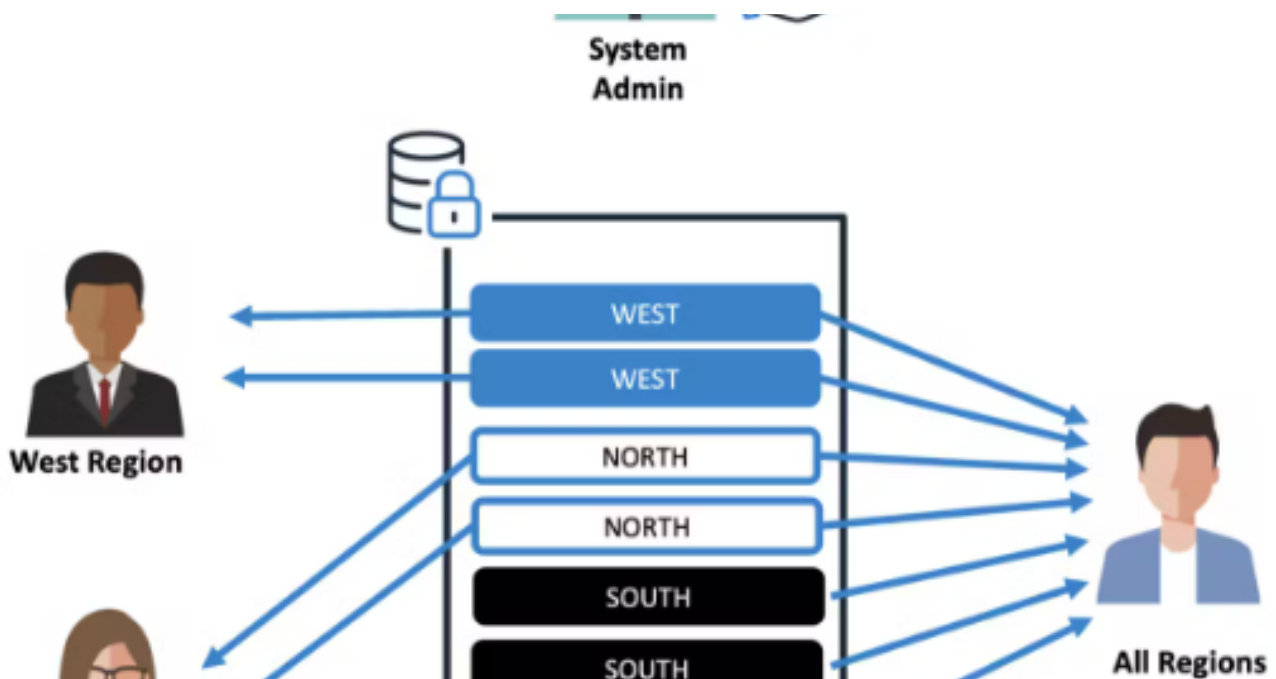


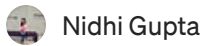
Nidhi Gupta

DECODE vs CASE IN ORACLE

While working on one of the projects got a chance to explore DECODE and CASE STATEMENT in Oracle Database.

★ Oct 8, 2022 🖱 106





Nidhi Gupta

Row Level Security(RLS) with Unity Catalog in Databricks

Row-level security (RLS) with Unity Catalog is a powerful feature designed to enhance data governance and security in a multi-tenant...

★ Jan 24 🖱 18

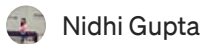


Open in app ↗

Medium



Search



Nidhi Gupta

“LIKE” vs “ILIKE” operator in PostgreSQL

In this article, we will walk through the “LIKE” and “ILIKE” operators used in PostgreSQL.

★ Sep 14, 2021 🖱 31



See all from Nidhi Gupta

Recommended from Medium

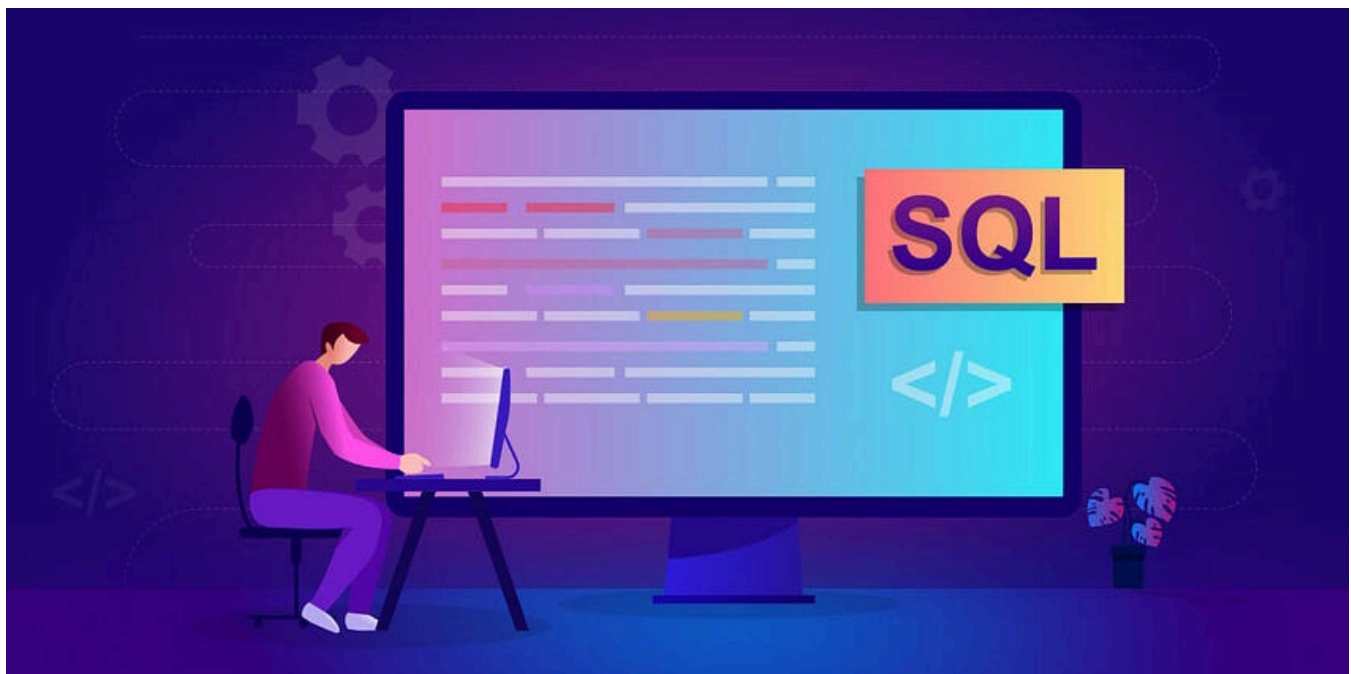


 In Databases by Sergey Egorenkov

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  234  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 🖱 186 💬 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

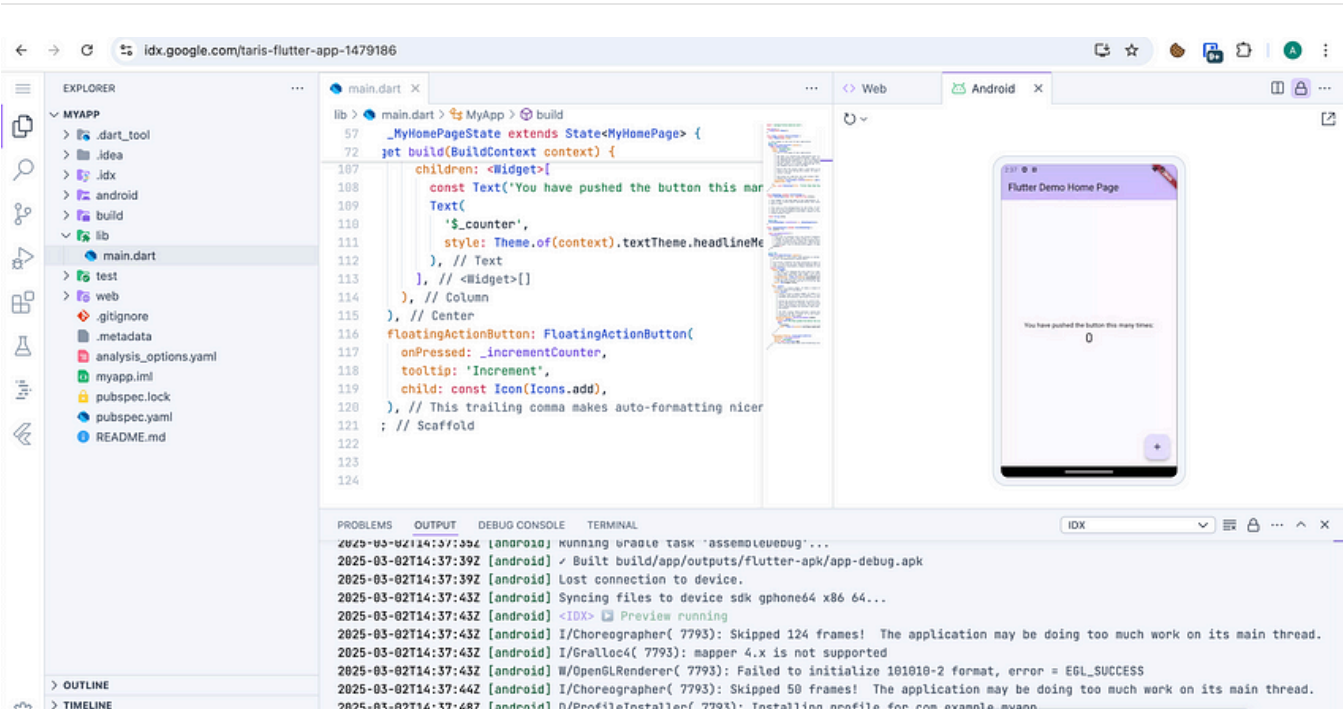
★ Mar 27 🖱 2K 💬 27


 Mateus Trentz

7 Amazing DBeaver Tips and Tricks to Improve Your SQL Workflow

Straight-to-the-point tips for the best SQL IDE

Mar 18 45 2



 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 201



at it Means	Best Used For
e directly in the row	Simple data like INT
e in the row (unless large)	Larger types, but tri
1press + store out-of-row	Long texts, large ob
e 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