PostgreSQL VACUUM vs ANALYZE

Sheikh Wasiu Al Hasib · Following
3 min read · Aug 2, 2024

Listen Share More

`**VACUUM**` and `**ANALYZE**` are two essential maintenance operations in PostgreSQL that help manage data storage efficiency and query performance. They are crucial for maintaining a healthy and performant database system.

VACUUM

Purpose:
`**VACUUM**` reclaims storage occupied by dead tuples. In PostgreSQL, when data is updated or deleted, the old data is not immediately removed; instead, it is marked as obsolete. This allows transactions that started before the data was updated to still see the old data. These obsolete rows are known as "dead tuples."

How it Works:
- `**VACUUM**` scans the database tables and removes these dead tuples, making the space available for future data inserts and updates.
- It also updates the visibility map, which helps optimize future queries and vacuum operations.

- There are two main types of `VACUUM`:
— Standard **VACUUM:** Removes dead tuples and updates statistics.
— **VACUUM FULL:** Rewrites the entire table, compacting it by removing dead tuples and reclaiming the space. This can be resource-intensive and requires a table lock, making it less suitable for use in production environments without careful planning.

Advantages:
- **Space Reclamation:** Frees up disk space, making it available for new data.

- **Prevents Transaction ID Wraparound:** Regular vacuuming is critical to prevent transaction ID wraparound issues, which can lead to database corruption if not managed properly.
- **Performance Improvement:** Reduces bloat, which can significantly improve performance by reducing the amount of data that needs to be scanned during queries.

ANALYZE

Purpose:
`**ANALYZE**` collects statistics about the contents of tables in the database, particularly the distribution of data within columns. This information is used by the PostgreSQL query planner to create efficient query execution plans.

How it Works:
- `**ANALYZE**` samples rows from the tables and gathers statistics on column data distributions, such as the most common values and the number of distinct values.
- These statistics are stored in the system catalog and are crucial for the query planner to estimate the cost of different query execution plans accurately.

Advantages:
- **Improved Query Performance:** By providing the query planner with accurate data distribution statistics, `**ANALYZE**` helps the planner choose the most efficient execution plans, leading to faster query performance.
- **Optimal Index Usage:** Helps in the effective use of indexes, as the planner can better understand the selectivity of indexed columns.

When is VACUUM and ANALYZE Important?

1. Regular Maintenance:
— Regularly running `**VACUUM**` and `**ANALYZE**` is crucial for maintaining database performance and preventing data bloat. Many PostgreSQL installations use `autovacuum`, an automated process that periodically runs these operations.
— Autovacuum: Autovacuum automatically performs `**VACUUM**` and `**ANALYZE**` on tables that need maintenance, based on thresholds related to the number of tuples updated or deleted. It's essential to ensure that `**autovacuum**` settings are appropriately configured for your workload.

2. After Large Data Changes:
— After bulk inserts, updates, or deletes, it is often necessary to run `**VACUUM**` and `**ANALYZE**` to reclaim space and update statistics. This ensures that the database

remains efficient and queries continue to perform well.
— For instance, after a batch update that affects many rows, running **`VACUUM ANALYZE`** can help reclaim space and provide fresh statistics to the query planner.

3. Before Query Optimization:
— Before optimizing queries, it is advisable to run `ANALYZE` to ensure that the planner has up-to-date statistics. This can significantly impact the planner's ability to choose the most efficient query execution plan.

4. Preventing Transaction ID Wraparound:
— PostgreSQL uses a 32-bit counter for transaction IDs, and if this counter wraps around, it can lead to data corruption. Regular **`VACUUM`**ing of tables is necessary to prevent this by advancing the "oldest transaction ID" in the system.

Conclusion

`VACUUM` and **`ANALYZE`** are critical for maintaining PostgreSQL performance and stability. **`VACUUM`** reclaims space and prevents transaction ID wraparound, while **`ANALYZE`** updates statistics crucial for query planning. Regularly running these operations, either manually or through **`autovacuum`**, helps ensure that the database remains efficient and performant, especially after significant data modifications. Proper maintenance using **`VACUUM`** and **`ANALYZE`** is essential for any production PostgreSQL environment.

Postgresql Vacuum Postgresql Analyze Postgresql Autovacuum Postgresql

Following

Written by Sheikh Wasiu Al Hasib

80 Followers · 3 Following

I am a PostgreSQL DBA with RHCE,RHCSA, Hardening ,CKA and CKS certified. I enjoy to work with database, automation tools like ansible ,terraform, bash etc. .

No responses yet

Gvadakte

What are your thoughts?

More from Sheikh Wasiu Al Hasib

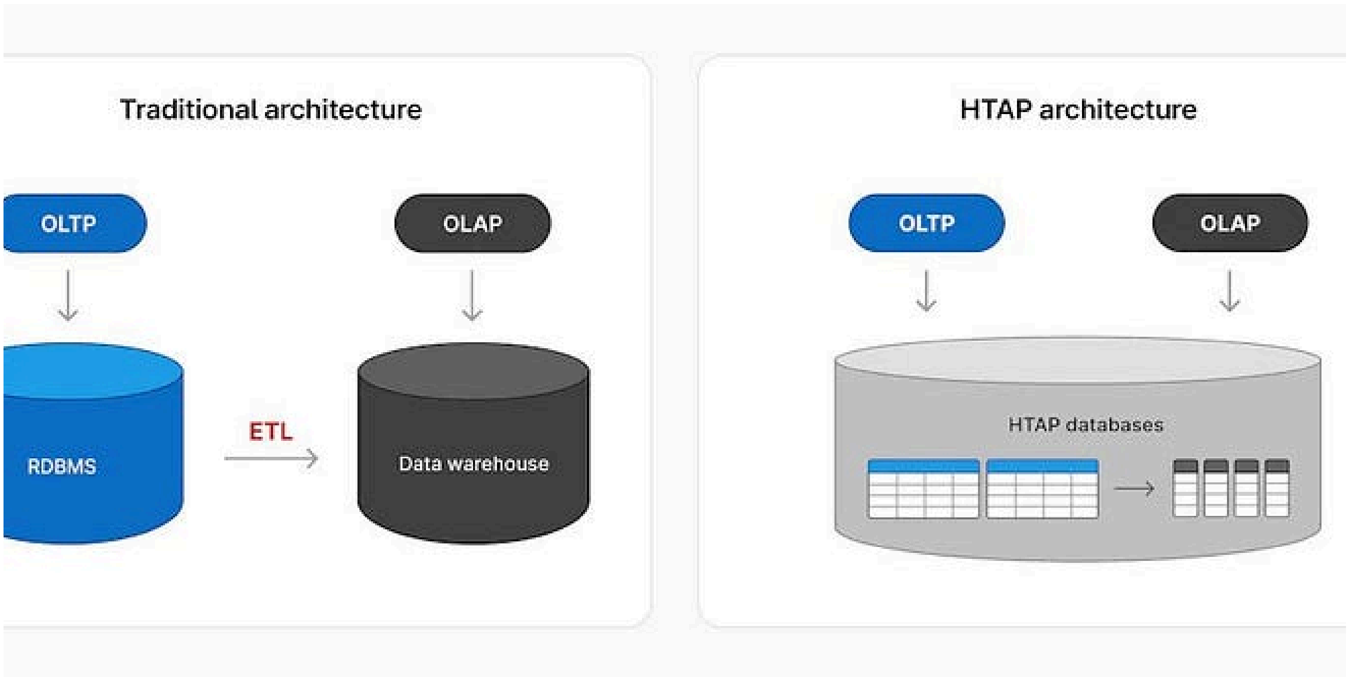
Open in app ↗

Medium Search

Effect of multiple update on same row in PostgreSQL tables

In PostgreSQL, updates to the same row in a table do not lock the whole table; instead, they lock the specific row being updated. Here's...

Aug 2, 2024 6



Sheikh Wasiu Al Hasib

PostgreSQL Hybrid Transactional/Analytical Processing using

What is HTAP?

Jul 16, 2024 6

Sheikh Wasiu Al Hasib

Balancing Disk Space and Reliability with PostgreSQL's max_slot_wal_keep_size

The `max_slot_wal_keep_size` parameter in PostgreSQL is a configuration setting that helps manage the retention of WAL (Write-Ahead…

Aug 4, 2024 1

Sheikh Wasiu Al Hasib

Different method of replicating data from PostgreSQL to Clickhouse as a Reporting Database

Choosing between ClickHouse and PostgreSQL depends largely on the specific use cases and the nature of the workloads they need to support…

Aug 26, 2024 1

See all from Sheikh Wasiu Al Hasib

Recommended from Medium

In Databases by Sergey Egorenkov

Making SQL query 40x faster for 10 million rows table

Make your SQL query really fast using this approach

Mar 17 10

Ajaymaurya

How Often Should You Reindex Your PostgreSQL Database? A Data-Driven Approach

PostgreSQL is one of the most powerful and flexible relational databases available today. However, like any database system, it requires…

Mar 24

In Dev Genius by Doran Gao

Efficient Large Table Cleanup in PostgreSQL

“We can only see a short distance ahead, but we can see plenty there that needs to be done.” — Alan Turing

Oct 31, 2024 1

In Towards Dev by Nakul Mitra

PostgreSQL Performance Optimization — Cleaning Dead Tuples & Reindexing

Performance optimization is crucial in PostgreSQL to ensure efficient query execution and minimal resource consumption.

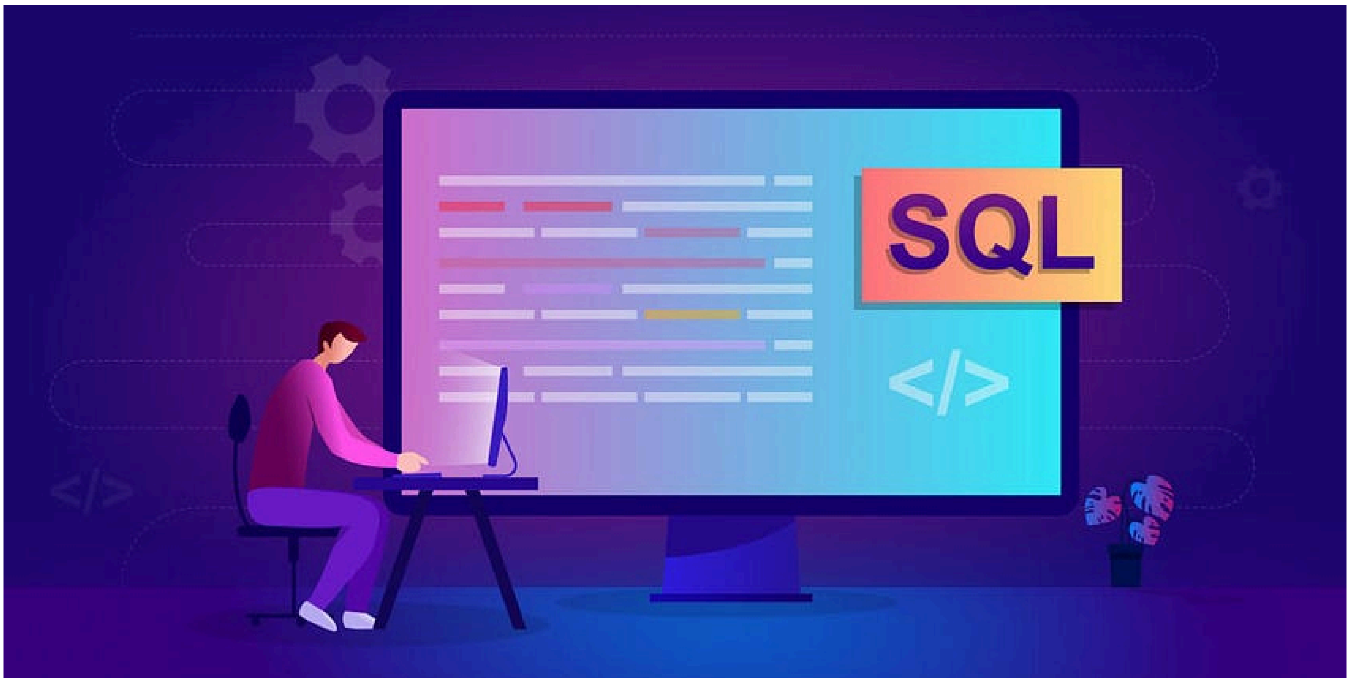
Mar 28 1

Dumindu Jayasekara

PostgreSQL: How data is stored in your drive

PostgreSQL is an open-source Relational Database Management System that uses and extends the SQL language to provide a powerful, robust…

Dec 28, 2024 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 27 208 5

See more recommendations