PostgreSQL Vacuum and Vacuum full are not two different processes

PostgreSQL Vacuum and Vacuum full are not two different processes

---

PostgreSQL's `VACUUM` and `VACUUM FULL` are not separate processes but rather different operational modes of the same maintenance command. Here's why:

Core Implementation

Both commands share the same underlying codebase and are executed through the **`vacuum_rel()`** function in PostgreSQL's source code (`src/backend/commands/vacuum.c`). The key distinction lies in the **`FULL`** option, which triggers additional steps:

- **Standard `VACUUM`**:
  - Removes dead tuples (obsolete rows) and marks space reusable *within PostgreSQL*
  - Updates the visibility map to optimize future queries
  - Runs concurrently with read/write operations
- **`VACUUM FULL`**:
  - Rewrites the entire table into a new disk file, compressing it and reclaiming space for the *operating system*
  - Rebuilds all indexes and requires an `ACCESS EXCLUSIVE` lock, blocking other operations

Key Differences in Behavior

Aspect	Standard VACUUM	VACUUM FULL
Space Reclamation	Internal reuse only	OS-level space release
Locking	Non-blocking	Full table lock

Aspect	Standard VACUUM	VACUUM FULL
Performance Impact	Lightweight, incremental	Heavy, resource-intensive
Use Case	Routine maintenance	Severe table bloat remediation

Why They Aren't Separate Processes

- **Shared Code Path**: Both use the same core logic for dead-tuple identification and cleanup. `VACUUM FULL` adds a table-rewrite step by calling `cluster_rel()`
- **Configuration Integration**: Parameters like `autovacuum_vacuum_scale_factor` apply to both, and autovacuum workers handle standard `VACUUM` by default
- **Unified Command Structure**: The `FULL` option is a modifier rather than a standalone tool, as seen in the SQL syntax:

```pgsql
VACUUM (FULL, ANALYZE) table_name;  -- vs.
VACUUM table_name;
```

When to Use Each

- **Standard `VACUUM`**: Daily maintenance to prevent bloat from MVCC dead tuples
- **`VACUUM FULL`**: Rarely, for extreme cases where table size has grown uncontrollably due to long-unvacuumed updates/deletes

In summary, while their outcomes differ significantly, `VACUUM` and `VACUUM FULL` are part of a single maintenance framework, differentiated primarily by the aggressiveness of space reclamation and locking behavior.

Comprehensive Guide to Aggregate Functions in PostgreSQL

PostgreSQL is equipped with a robust suite of statistical functions that are essential for performing detailed data analysis directly within the database. These functions allow users to calculate various statistical measures, such as averages, variances, … Continue reading

The WebScale Database Infrastructure Operations Experts in PostgreSQL, MySQL, MariaDB, MongoDB and ClickHouse

PostgreSQL Vacuum Internals – What happens during PostgreSQL Vacuum Processing?

PostgreSQL Vacuum Internals – What happens during PostgreSQL Vacuum Processing? During PostgreSQL vacuum processing, the database system scans through the table or index being vacuumed, removing any dead or outdated rows. This process is necessary … Continue reading

The WebScale Database Infrastructure Operations Experts in PostgreSQL, MySQL, MariaDB, MongoDB and ClickHouse