9. work_mem = 27306kB

- **Description**: Defines the amount of memory used for internal query operations like sorting and hashing (e.g., `ORDER BY`, `GROUP BY`).
- **Default**: 4MB

What is `work_mem` in PostgreSQL?

`work_mem` is a setting in PostgreSQL that determines how much **memory** each query operation (e.g., sorting, joining) can use **in RAM** before it switches to disk. It directly affects the performance of queries that process large amounts of data.

Simple Explanation:

- If a query operation (like a sort) needs more memory than `work_mem`, PostgreSQL will use temporary disk space, which is much slower than RAM.
- By increasing `work_mem`, you can allow queries to perform these operations in RAM, making them faster.

How it Works with an Example:

Scenario:

You are running this query on a table with **10 million rows**:

sql

Copy code

```
SELECT * FROM orders ORDER BY order_date;
```

This query sorts a large number of rows.

Case 1: `work_mem = 4MB`

- PostgreSQL allocates **4 MB of memory** for the sorting operation.
- Since sorting requires more than 4 MB (e.g., 50 MB is needed), PostgreSQL uses temporary files on disk for the excess data.
- **Result:** Sorting takes longer because disk access is slower than memory.

Case 2: `work_mem = 64MB`

- PostgreSQL allocates **64 MB of memory** for the sorting operation.
- The entire sorting can now happen in RAM without needing disk space.
- **Result:** Sorting is much faster because everything happens in memory.

Key Points:

1.Per Operation, Not Per Query:

1. `work_mem` applies to **each operation** in a query.
2. If a query has multiple operations (e.g., sorting and joining), each one gets its own `work_mem`.

2. Danger of Setting It Too High:

1. If too many queries run at the same time, and each uses a large `work_mem`, the server can run out of RAM, causing crashes or slowdowns.

Recommended Values:

- **Small databases or low activity:** Use the default value (4 MB).
- **High-performance systems or large queries:** Increase it to `32 MB` or more but monitor memory usage.
- For heavy analytic queries, you can temporarily set a higher value like `128 MB` for that session.

- **Impact**: A larger value allows more memory for complex queries, improving query performance by avoiding disk-based operations. However, setting it too high can cause excessive memory usage when many queries run concurrently.

Work_mem explaination:-

"Per Operation, Not Per Query" - Meaning in Simple Terms

The `work_mem` setting applies to **each operation** within a query, not the entire query as a whole.

What Does This Mean?

When you run a query in PostgreSQL, the query can have multiple steps (or operations), such as:

1. Sorting
2. Joining tables
3. Grouping data
4. Creating hash tables

Each of these operations gets its **own memory allocation** based on the `work_mem` setting.

Example to Understand:

Query:

```
SELECT customers.name, SUM(orders.amount)FROM customersJOIN orders ON
customers.id = orders.customer_idGROUP BY customers.nameORDER BY
SUM(orders.amount) DESC;
```

What Happens Here:

This query has **three operations**:

1. **Joining:** Combine `customers` and `orders` tables.
2. **Grouping:** Group data by `customers.name` and calculate the total amount (`SUM`).
3. **Sorting:** Sort the results by the total amount in descending order.

If `work_mem = 4MB`:

- **Join Operation:** PostgreSQL gets **4 MB** to create a hash table in memory for joining the tables. If this exceeds 4 MB, it uses the disk.
- **Grouping Operation:** Another **4 MB** is allocated to group data and calculate `SUM`.
- **Sorting Operation:** A separate **4 MB** is allocated to sort the results.

Total Memory Used:
If all operations happen simultaneously, PostgreSQL will use **4 MB × 3 = 12 MB** for this query.

If `work_mem = 64MB`:

- Each operation gets **64 MB** instead of 4 MB.

- PostgreSQL is less likely to use the disk, making the query much faster.

Total Memory Used:
If all operations happen simultaneously, PostgreSQL will use **64 MB × 3 = 192 MB** for this query.

Why is This Important?

- If you increase `work_mem` too much and many queries run simultaneously, the total memory usage can exceed the server's available RAM, causing performance issues or crashes.
- Example: If 50 queries each have 3 operations and `work_mem = 64MB`:
  - Total memory used = `50 × 3 × 64 MB = 9.6 GB`.

Summary:

- **Each operation** in a query gets its own `work_mem` allocation.
- For complex queries with many operations, the total memory usage will be **work_mem × number of operations**.
- Be cautious when setting a high `work_mem`, especially on systems with many concurrent queries.

The work_mem setting in PostgreSQL defines how much memory a single operation in a query can use before switching to disk. If you have multiple SELECT queries running, each query (and each operation in those queries) will use its own work_mem allocation. Let's break it down step by step.

Key Points About `work_mem`:

Per Operation, Not Per Query:

1. `work_mem` applies to each operation in a query (e.g., sorting, hashing, joining), not the query as a whole.
2. Complex queries with multiple operations (e.g., multiple joins or sorts) can use `work_mem` multiple times.

Multiple Queries:

1. If multiple `SELECT` queries run at the same time, each query (and its operations) gets its own `work_mem`.

Total Memory Usage

1. Total memory consumed depends on: Total Memory Usage=

   Total Memory Usage=work_mem×Number of Operations per Query)×(Number of Queries Running Simultaneously)

Example Scenario:

Database Setup:

- `work_mem = 4MB` (default value).
- You are running 3 `SELECT` queries simultaneously.

  Query 1: Simple Query

  ```
  SELECT * FROM orders ORDER BY order_date;
  ```

- **Operation:** Sorting (`ORDER BY`).
- Memory Usage:
  - Sorting uses **4 MB** of memory (1 operation × 4 MB).

Query 2: Complex Query

```
SELECT o.customer_id, SUM(o.total)

FROM orders o

JOIN customers c ON o.customer_id = c.id

GROUP BY o.customer_id

ORDER BY SUM(o.total) DESC;
```

Operations in Query 2:

1. **Hash Join:** Joins `orders` and `customers`.
2. **Group By:** Groups data by `customer_id` and calculates the `SUM`.
3. **Sort:** Orders the results by `SUM(total)`.

Memory Usage: 3operations×4MB=12MB per query

What Happens If More Queries Run?

If more queries are executed, PostgreSQL will allocate more memory for each query's operations. If the total memory usage exceeds the server's physical RAM, the system may:

1. Start swapping (using disk instead of RAM), which slows everything down.
2. Fail if the system runs out of memory entirely.

Best Practices for `work_mem`:

1.

Estimate Total Usage:
Set `work_mem` based on your system's available RAM and expected query concurrency:

$$\text{Max Work Memory} = \frac{\text{Available RAM}}{\text{Max Concurrent Queries} \times \text{Operations per Query}}$$

Example:

- Server has 16 GB RAM.
- Max 50 concurrent queries.
- Average 2 operations per query.

$$work\_mem = \frac{16\,\text{GB}}{50 \times 2} = 160\,\text{MB per operation.}$$