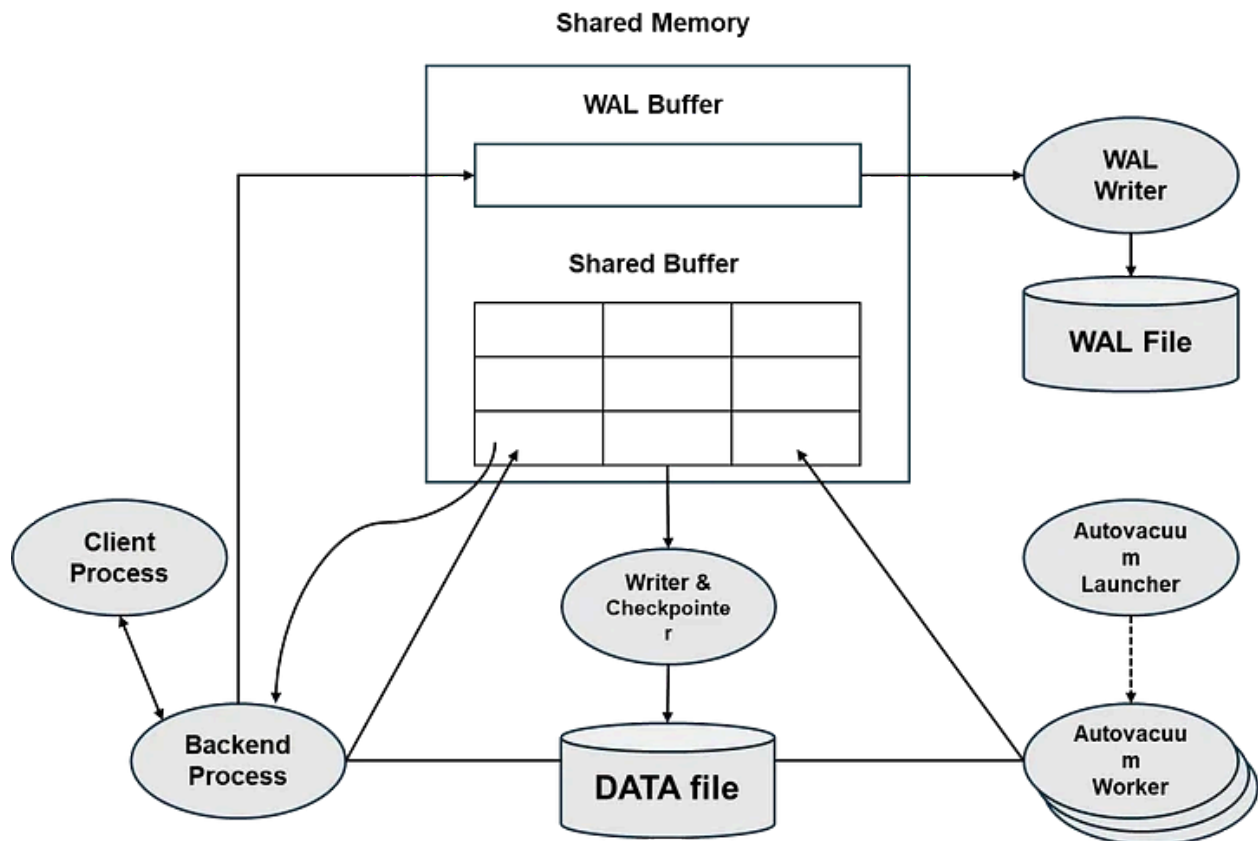


## Estrutura Interna do PostgreSQL

Compartilhamento de memória e processos executando no backend

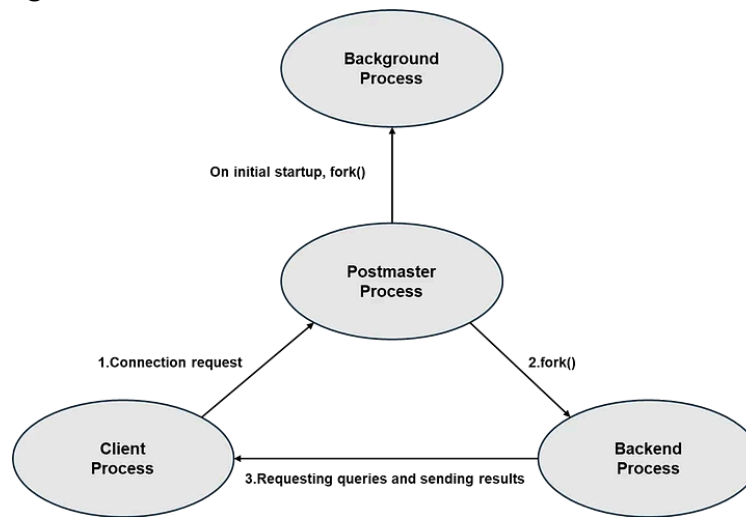


A memória compartilhada (shared memory) é formada por duas partes:

**Shared Buffer:** é o *buffer pool* cujo objetivo é minimizar o acesso ao disco (I/O). Segue o conceito que grandes porções do buffer tem que serem acessados de forma rápida; minimizar a competição quando acessado por várias transações e páginas mais utilizadas devem ficar mais tempo no buffer.

**WAL Buffer:** Veremos com detalhes no estudo dos gerenciadores de transação e de recuperação após falha. Nele são armazenadas as atualizações feitas pelas transações, mas que ainda não foram para o disco.

## Processos do PostgreSQL



**Postmaster process:** é o processo executado no boot do SGBD. Ele inicializa as estruturas do postgres (buffers, executa os processos internos, entre outros) e cria os processos de backend das conexões.

**Background process** (processos internos): todos os processos utilizados pelo PG para garantir o funcionamento do SGBD. Como exemplos desse tipo de processo: coletor de estatísticas, escritor do WAL Buffer e escritor das páginas “suja” no disco.

**Backend process** (processos de retaguarda): configurado pelo parâmetro **max\_connections** (default 100). Executam as consultas e devolvem os resultados para os clientes. Utilizam basicamente as memórias locais para executarem suas tarefas:

**work\_mem:** espaço utilizado para operações de ordenação, geração de índices. O padrão é 4 MB

**maintenance\_work\_mem:** usado para o processo de *vacuum* e criação de índices. Default 64 Mb.

**temp\_buffers:** espaço para armazenar tabelas temporárias. Default 8 Mb.

## Organização dos Dados PostgreSQL

### Banco de dados:

Composto de múltiplos banco de dados, chamado de **database cluster**.

Após a inicialização do SGBD, três banco de dados são criados: **template0**, **template1** e **postgres**.

**template0** e **1** são utilizados como templates para criar os bancos dos usuários (incluindo os catálogos do sistema - dicionário de dados). Após a inicialização do SGBD (programa **initdb**, ambos os templates têm uma lista idêntica de tabelas. Mas o **template1** é o que permite criar os objetos necessários para os usuários. O create database copia o **template1** para o banco a ser criado pelo usuário.

Se acessar o banco **template1** e criar objetos nele, todos os bancos criados herdarão tais objetos.

## Tablespaces:

Da mesma forma que o banco de dados, após a inicialização duas *tablespaces* são criadas: **pg\_default** e **pg\_global**. Como o nome diz, a **pg\_default** é a utilizada quando da criação de um objeto em que o usuário não especifica uma tablespace. A **pg\_global** é utilizada pelos objetos do PG.

A localização física da **pg\_default** é \$pastaDadosPG\base (ex.: /var/lib/postgresql/16/base)

A localização física da **pg\_global** é \$pastaDadosPG\global (ex.: /var/lib/postgresql/16/global)

A pasta /etc/postgresql/<version>/main contém o **postgresql.conf** que é o arquivo de configurações do PG. Ou executar o comando **show config\_file**; no prompt do **psql**.

**ALTER SYSTEM SET** <parâmetro> { **TO** | = } { **value** | 'value' | **DEFAULT** }, altera algumas das configurações do PG.

## Estrutura Tabelas

Uma tabela gera três arquivos no disco (sob a tablespace que está associada): (i) um com os dados sendo o nome formado pelo **OID** na tabela, (ii) outro para armazenar o espaço livre da tabela **OID\_fsm**, e (iii) para gerenciar os blocos livres **OID\_vm**. Para índices, apenas dois arquivos são criados: **OID** e **OID\_fsm**.

Existem um grupo de função com o prefixo **pg\_relation\_<tipo>(<table>)** que retornam dados sobre uma tabela. **<tipo>**: **filepath** (pasta onde se encontra o arquivo de dados da tabela, o ponto de entrada é a pasta da tablespace da tabela), **filenode** (retorna OID da tabela), **size** (tamanho da tabela - **pg\_total\_relation\_size** retorna o tamanho total ocupado pela tabela e seus agregados). Obs: a função **pg\_size\_pretty** converte os bytes em um formato mais legível: **pg\_size\_pretty(pg\_relation\_size(tabela))** retorna o tamanho da tabela em um formato mais amigável.

## Views do Sistema

**pg\_stat\_activity** apresenta dados dos processos rodando.

**pg\_stat\_statements** (criar a extensão **create extension** e atualizar parâmetro **shared\_preload\_libraries** no postgresql.conf): auxilia a rastrear as estatísticas de planejamento e execução de todos comandos SQL executados no servidor. Alguns dados: oid do usuário e do banco (userid e bdid), a consulta realizada (query)

**pg\_database** (parecido com **VI** no **plsql**)

**pg\_class**: lista todos os objetos do BD: oid, nome, tipo dono, tablespace, entre outros ([aqui](#) tem a relação completa). Atributo **relkind** pode ser r = ordinary table, i = index, S = sequence, t = TOAST table, v = view, m = materialized view, c = composite type, f = foreign table, p = partitioned table, l = partitioned index

Ex.:

```
select oid, relname, reltype, relowner, relfilenode, reltablespace,
       relpages, reltuples, relhasindex, relkind
from pg_class;
```

**pg\_tablespace** - informações sobre as tablespaces

**pg\_config** - configurações de sistema do PostgreSQL

**pg\_file\_settings** - apresenta um sumário dos parâmetros do servidor incluindo: a pasta dos dados, as configurações de língua, data e moeda, max\_connections, entre outros.

**pg\_indexes**: informações sobre os índices existentes no BD. Incluindo o nome da tabela do índice e sua definição.

```
select indexname, indexdef from pg_indexes where tablename=<tabela>;
```

**pg\_locks**: lista de objetos bloqueados (atributos database e relation - é retornado o OID, para acessar o nome: `select * from pg_class where oid=OID;`), **locktype** define o tipo de bloqueio utilizado ([aqui](#) pode ser encontrado a relação de tipos de bloqueio)

**pg\_matviews**: lista de visões materializadas

**pg\_user** e **pg\_roles**: listam os usuários e os papéis existentes

**pg\_sequences**: as sequências criadas

**pg\_tables**: lista alguns dados das tabelas (tablespace, schema, proprietário, se tem índices, triggers ...)

### Esquema com o dicionário:

information\_schema: é um esquema que armazena o catálogo (dicionário) de um banco de dados. É usado para pesquisar informações sobre os objetos do banco de dados conectado.

**information\_schema**. ->

**tables** (table\_type 'VIEW' ou 'BASE TABLE' - table\_schema='public')

**columns** (table\_name = <tabela>)

**table\_constraints** (restrições das tabelas)

**constraint\_column\_usage** (colunas das restrições das tabelas)

### Consultar stored procedures

```
select prosrc from pg_proc where proname ilike '<procedure|function name>';
```

-- ilike ignora maiúscula/minúscula (mostra o fonte da stored procedure)

Atributo **prokind** pode ser **p**, **f** ou **a**. Úteis p -> procedure e f -> function

### Consultas úteis:

### Uso das tabelas e índices com as tuplas retornadas:

```
SELECT
    t.schemaname, t.tablename, c.reltuples::bigint      AS num_rows,
    pg_size_pretty(pg_relation_size(c.oid))            AS table_size,
    psai.indexrelname                                   AS index_name,
    pg_size_pretty(pg_relation_size(i.indexrelid))     AS index_size,
    CASE WHEN i.indisunique THEN 'Y' ELSE 'N' END      AS "unique",
    psai.idx_scan AS number_scans, psai.idx_tup_read AS tuples_read,
    psai.idx_tup_fetch                                   AS tuples_fetched
FROM pg_tables t
    LEFT JOIN pg_class c ON t.tablename = c.relname
```

```

LEFT JOIN pg_index i ON c.oid = i.indrelid
LEFT JOIN pg_stat_all_indexes psai ON i.indexrelid =
                                psai.indexrelid
WHERE t.schemaname NOT IN ('pg_catalog', 'information_schema')
ORDER BY 1, 2;

```

**Missing indexes (unused indexes):**

```

SELECT      relname,  seq_scan - idx_scan AS too_much_seq,
               CASE WHEN seq_scan - coalesce(idx_scan, 0) > 0
                   THEN 'Missing Index?'
                   ELSE 'OK'
               END,
               pg_relation_size(relname::regclass) AS rel_size,
               seq_scan, idx_scan
FROM        pg_stat_all_tables
WHERE       schemaname = 'public' -- set the right schema
AND         pg_relation_size(relname::regclass) > 80000 -- threshold?
ORDER BY    too_much_seq DESC;

```

**Outros:**

**pg\_settings:** lista todas as configurações do postgresql.conf (ou show all;) - **pending\_restart** indica se o SGBD está esperando reboot para o novo valor do parâmetro ser utilizado.

```
select version(); -- PG version
```

Restart progress:     sudo systemctl restart postgresql  
 Locate **postgresql.conf** :    /etc/postgresql/<version>/main  
                               /etc/postgresql/17/main/

ps aux | grep postgres (shows postgres's processes)

How to track execution statistics of SQL statements:

```
select * from pg_stat_statements order by total_exec_time desc;
```

If **pg\_stat\_statements** does not exist, run **create extension pg\_stat\_statements;**

Add **pg\_stat\_statements** to "shared\_preload\_libraries" parameter in postgresql.conf