**PostgreSQL - System Architecture**

**PostgreSQL**, often simply called Postgres, is an advanced open-source **Object-Relational Database Management System** (**ORDBMS**). It stands out due to its robust feature set, extensibility, and compliance with SQL standards. Originating as the successor to the **POSTGRES system**, one of the earliest database systems, PostgreSQL has evolved to become a highly preferred choice for a wide range of applications and developers worldwide.

**System Architecture of PostgreSQL**

The architecture of **PostgreSQL** is built around a client-server model, which is pivotal for handling multiple database functions such as managing connections, operations, and both static and dynamic assets. In the simplest terms, a PostgreSQL service has 2 processes:

**1. Server-side process**

This is the "**Postgres**" application that manages connections, operations, and static & dynamic assets. It efficiently handles multiple tasks including session management, query processing, and transaction management. Additionally, it ensures data integrity and concurrency control by maintaining locks and writing transaction logs. This process is pivotal in managing database connections and executing the queries received from client applications, making it essential for the robust performance of the system.

**2. Client-side Process(Front-end applications)**

These are the applications that users use to interact with the database. It generally has a simple UI and is used to communicate between the user and the database generally through APIs.

When the user runs queries on PostgreSQL, the Client Application can connect to the PostgreSQL server (Postmaster Daemon Process) and submit queries through one of many**Database Client Application** program interface supported by PostgreSQL like JDBC, Perl DBD, ODBC, etc. that helps to provide client-side libraries. In the Client Process, the Communication between Client Application and Client Application library occurs with the help of Library API as shown in the figure below:

A diagram of a computer process

AI-generated content may be incorrect.PostgreSQL System Architecture

**Core Components of PostgreSQL**

**1. Postmaster Daemon Process**

The system architecture of PostgreSQL is based on **Process-Per-Transaction Model(Client/Server Model)**. A running PostgreSQL site is managed by **Postmaster**which is a central coordinating process. It is also known as Server Process.

The postmaster daemon process is responsible for :

* Initializing the server
* Shutting Down the server
* Handling Connection requests from new clients.
* Perform Recovery.
* Run Background Processes.

**2. Shared Memory**

Shared memory is the memory that is simultaneously accessed by multiple programs in order to provide fast and efficient results with less redundancy. This is the memory that is reserved for Database Caching and transactional log caching. In PostgreSQL shared Disk Buffer and Shared Tables are Used whose working is explained below.

**Shared Disk Buffer:**The purpose of the shared disk buffer is to minimize the disk Input/Output.If it is not used then the Disk Input/Output takes more time which causes redundancy and an inefficient system. The advantages of using a shared buffer are:

* Reduce time.
* Can Access a very large amount of Data Easily.
* Minimize heating when multiple users is accessing at the same time.

**Shared Tables:**This approach involves using the same set of tables to host multiple client data. The main advantages of using this approach are:

* The Lowest Hardware Cost
* The Lowest Backup Cost
* It allows working with large data in a single database.

**3. Back-end process**

The Postmaster is responsible for handling initial client connections. For this, it constantly listens for new connections as a known port. After Performing an initialization process such as authentication of the user, the postmaster will give rise to a new backend server process to handle the new client. The client interacts only with the Backend server process like submitting queries and receiving queries result. This will show that PostgreSQL actually uses Process-per-transaction model.

The Backend Server is responsible for Executing queries submitted by the client by performing specific operations. Each backend server will handle only a single query at a time. At a time multiple clients are connected to the system hence multiple backend servers executing queries Concurrently. The back-end server access data from the main-memory buffer pool which is placed in shared memory.

After that, the result obtained is provided to the Client Process by Back-end Process.

| **WAL**  **(Write Ahead Log)Writer** | This process Write and flushes WAL Data on WAL buffer |
| --- | --- |
| **logging collector** | This process is also called logger. It will write an error message to the log file. |
| **Auto vacuum launcher** | When auto vacuum is enabled, this process has the responsibility of the auto vacuum daemon to carry vacuum operations on bloated tables. This process relies on the stats collector process for perfect table analysis. |
| **Archiver** | When  Achiever is enabled, the process has the responsibility to copy WAL log file to the specified directory. |
| **stats collector** | In this statistics information like *pg\_stat\_activity*  and *pg\_stat\_all\_tables* is collected |
| **checkpointer** | When a checkpoint occurs, the dirty buffer written to the file. |
| **writer** | It will periodically write the dirty buffer to the file. |

**4. Shared Pool**

The Shared pool is a RAM area within the RAM Heap that is created during the starting time. A shared pool is a component of **SGA (System Global Area)**. If Shared Pool is not available in RAM or it is not used then it results in high library cache reloads, high row cache reloads.

**Why PostgreSQL didn't use Shared Pool?**

PostgreSQL doesn't provide a shared pool although most of the Database Systems like Oracle, the Shared pool is an important component of its structure. It doesn't have because PostgreSQL will provide a feature to share SQL information at the process level as compared to Shared Pool.

Simply, if the user will execute the same SQL query several times in one process, it will**hard-parse**only once which is advantageous over other Database systems because, in another database system that uses a shared pool, the **hard-parse**occurs for a single SQL statement that is loaded from Shared pool. If the user executes simultaneously a single SQL query several times then it will cause more load.