PostgreSQL

Advanced Postgres

Surendra Panpaliya

Agenda

- Introduction to PostgreSQL
- History and development of PostgreSQL
- Key features and benefits of PostgreSQL
- PostgreSQL vs. Oracle database systems

Agenda

PostgreSQL vs Oracle: Key architectural differences

PostgreSQL installation & tools (pgAdmin, DBeaver)

Overview of pg_catalog, schemas, roles, and databases

Hands-On

Install PostgreSQL & pgAdmin

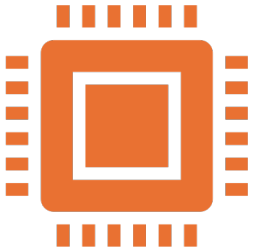
Connect to database, explore catalogs

Assignment

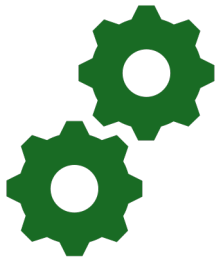
Install and configure PostgreSQL

Share screenshot of connected pgAdmin

Agenda

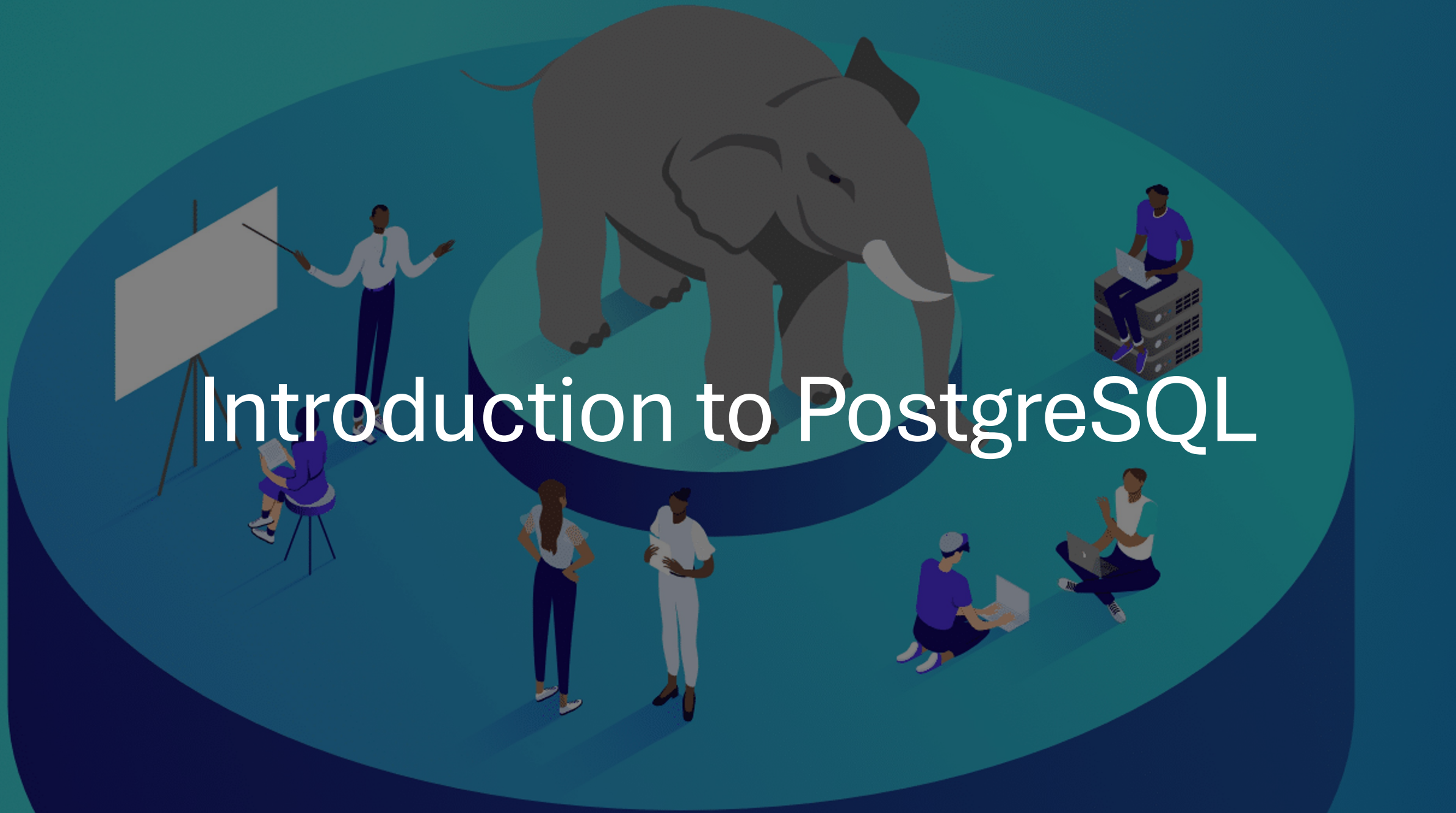


Installation (Windows, macOS, Linux)



Configuration basics

Connecting to the database



Introduction to PostgreSQL

What is PostgreSQL?

PostgreSQL (Postgres)

An advanced open-source

Relational Database Management System (RDBMS).

What is PostgreSQL?

- Known for
- Robustness
- Extensibility
- Standards compliance

What is PostgreSQL?

Designed to handle

Single-machine applications

Large-scale data warehousing

Web services

What is PostgreSQL?

Used for

Web

Mobile

Geospatial

Analytics applications

PostgreSQL 16

VARIETY OF
NEW FEATURES

IMPROVE
PERFORMANCE

USABILITY

SECURITY

Performance Improvements

Query Performance Enhancements

Partitioning Improvements

Parallelism Enhancements

Usability Enhancements

JSON and JSONB Enhancements

Enhanced Window Functions

Improved psql Command-Line Tool

Security Enhancements

Enhanced Authentication and Authorization

Advanced Data Encryption

Administrative Features

Enhanced Backup and Restore

Improved Monitoring and Logging

Developer Features

Enhanced PL/pgSQL

Advanced Indexing

Key Contributors

- **Michael Stonebraker**
- The primary Architect
- Original Postgres project
- Key figure in database research

PostgreSQL Global Development Group (PGDG)

A group of volunteers and Companies

Oversee the development

Maintenance of PostgreSQL

Community

Community-driven

Contributions from developers worldwide

Known for its collaborative spirit

Extensive documentation

Active mailing lists

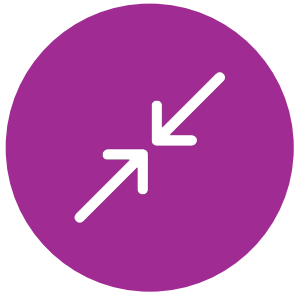
Design Philosophy

Standards Compliance

PostgreSQL adheres

To SQL standards

Ensuring compatibility



Ease of use

Extensibility

Designed to be highly extensible

Allowing users to define

New data types

Functions

Operators

Reliability and Stability

Focuses on delivering

Stable

Reliable database system

Suitable for

Mission-critical applications

Open Source

Released under the PostgreSQL License

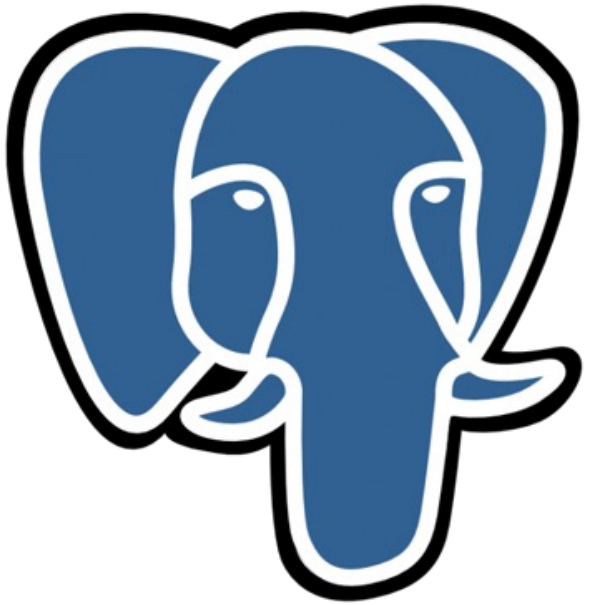
An open-source license

Allows for free use

Modification

Distribution

Major Milestones



PostgreSQL

MVCC (1997)

Multi-Version Concurrency Control

A crucial feature

for high Concurrency

Performance

SQL Compliance (1996)

Transition from

PostQUEL to SQL

Aligning with

Industry standards

Windows Support (2005)

Native support for

Windows

Broadening

PostgreSQL's user base

Streaming Replication (2010)

Hot standby

Improving

Data

Redundancy

Availability

Parallel Query Execution (2016)

Support for

Parallel Query Execution

Enhancing performance

for large queries.

Logical Replication (2017)

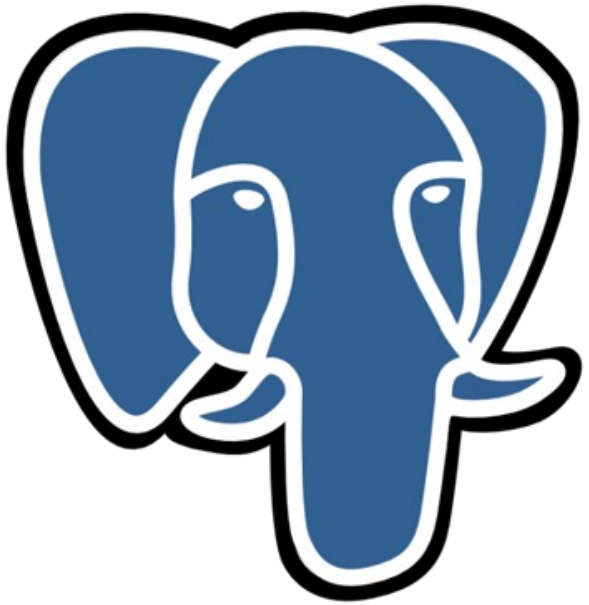
Introduction of

Logical replication,

Allowing selective

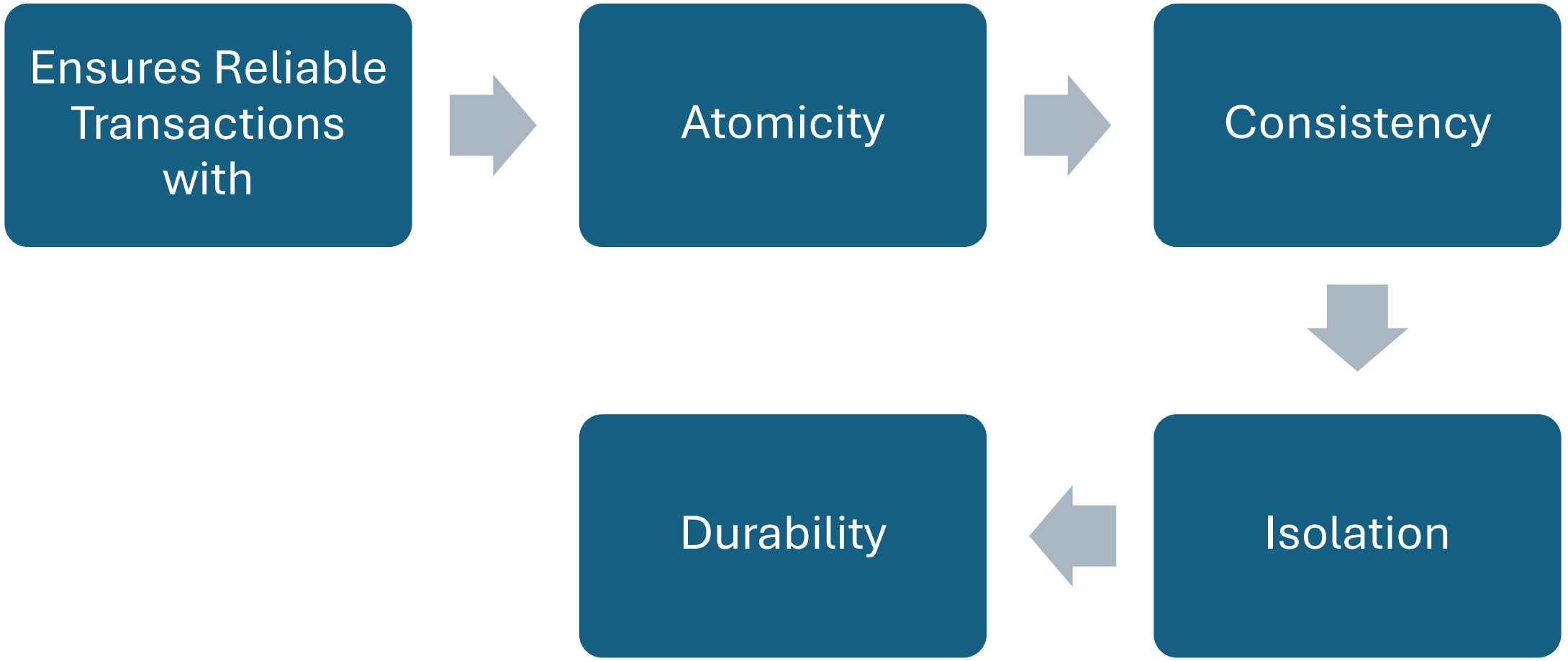
Replication of data

Key features and Benefits



PostgreSQL

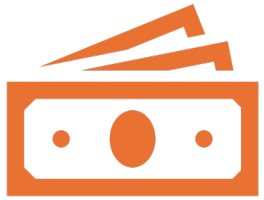
ACID Compliance



Atomicity

- Each transaction is treated
- Single ”Atomic" Unit
- Either completely succeeds or
- Completely fails.

Atomicity



If any part of the transaction fails

Entire transaction is rolled back

Database remains unchanged.

Example

If a transaction involves

transferring money from

Account A to Account B.

Example

Either both the debit from

Account A and

The credit to Account B

Happen or neither happens.

Example

If the debit succeeds

but the credit fails,

the transaction is rolled back.

Account A's balance is restored

to its original state

ACID Compliance

Provides robust support

For concurrent transactions

Without compromising data integrity

Consistency

Ensures that a transaction

From one valid state

To another valid state,

Maintain all predefined rules

Constraints, and triggers

Consistency

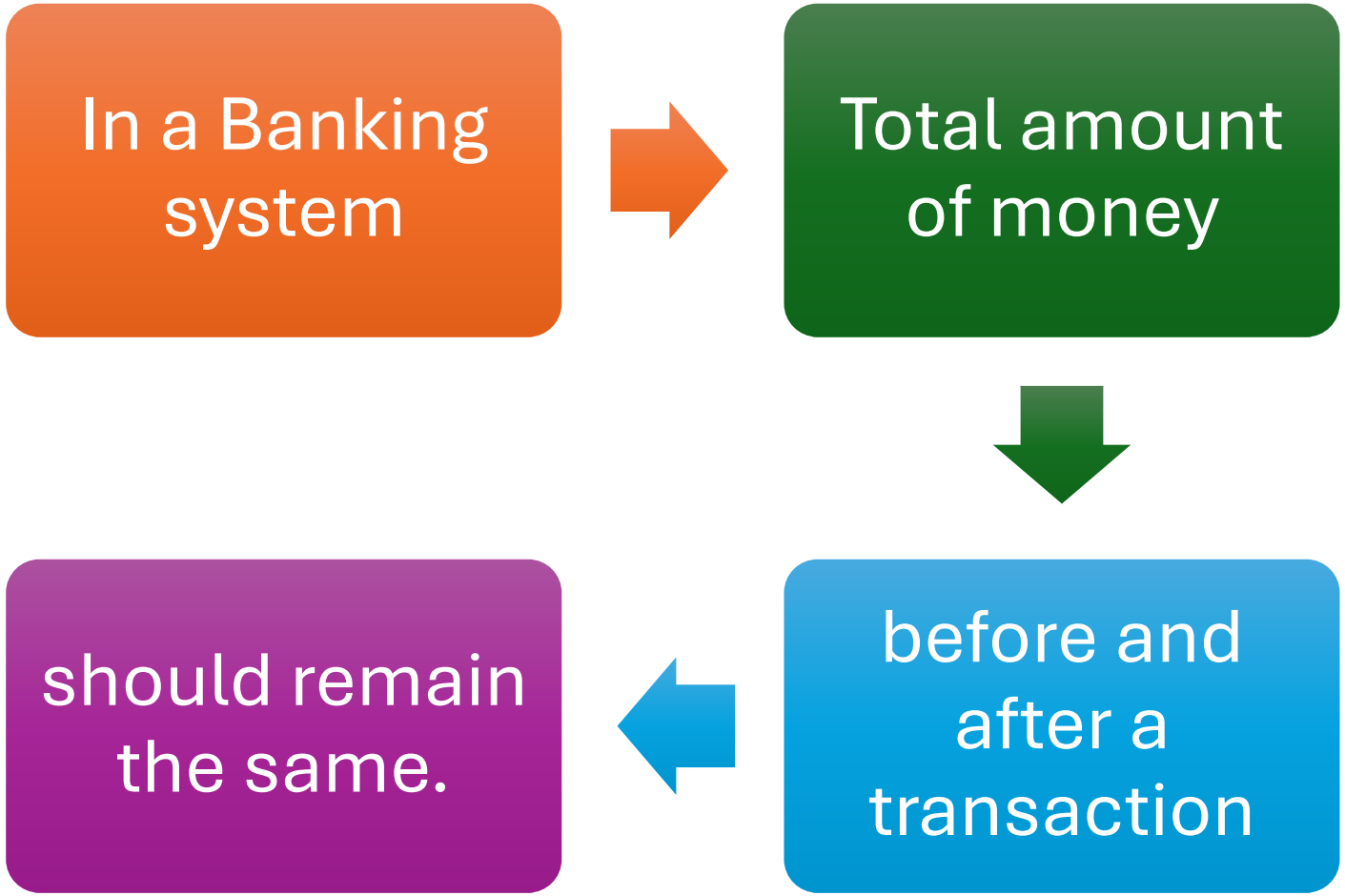
DATABASE MUST

REMAIN CONSISTENT

BEFORE AND AFTER

TRANSACTION

Consistency



Isolation

Transactions

Executed in

Isolation

From One another

Isolation

If two transactions

Running Concurrently

One transferring money from

Account A to Account B

Another from Account C to Account D

Isolation

Isolation ensures

Transactions

Do not see each other’s

Intermediate states

Durability

Once a transaction

Has been committed

Will remain

Even in the event of

a system failure

Durability

If a transaction to update

A bank balance is committed,

New balance is guaranteed

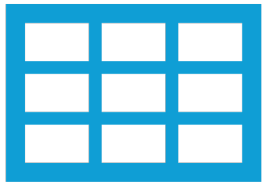
to be saved in the database.

Advanced Data Types

JSON/JSONB

Efficient storage

Querying of



JSON data.

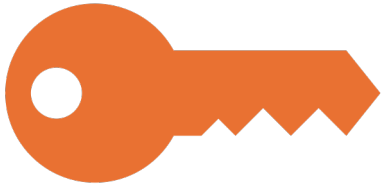
Arrays

Support for

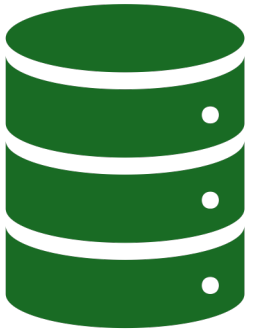
Multi-dimensional

Arrays

Hstore



Key-value pairs



Storage

Geometric Types

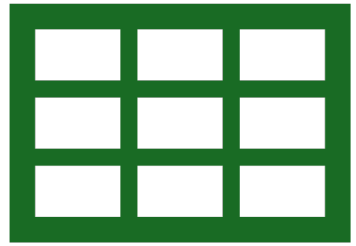
Points,

lines

polygons

Custom Types

Users can define



Own data types.

Extensibility

Users can create

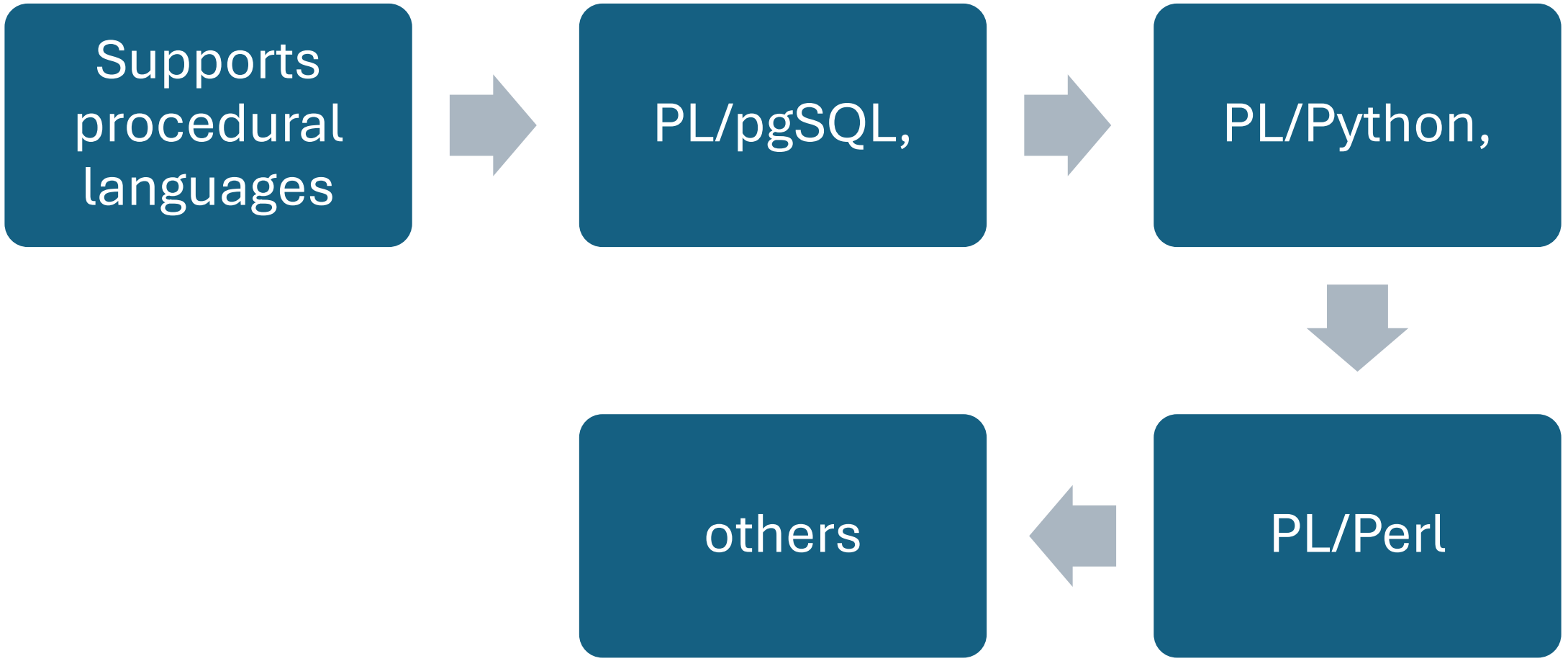
Custom functions

Operators,

Data types

index types.

Extensibility



Extensibility

Extensions like

PostGIS (for geospatial data)

Full-text search

Easily integrated

Multi-Version Concurrency Control (MVCC)

Allows multiple transactions

to occur simultaneously without locking.

Ensures data consistency

High performance

Indexing

Supports various index types

B-tree,

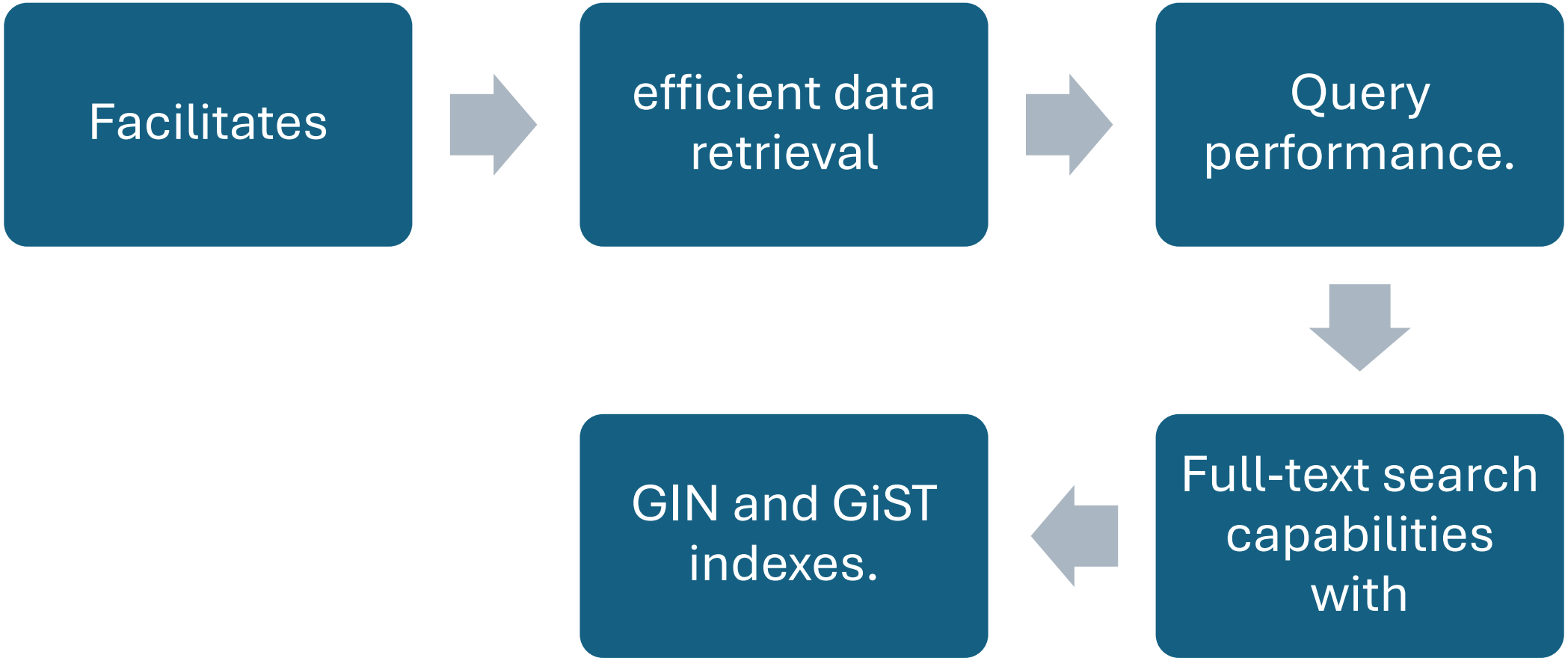
Hash,

GiST,

GIN

BRIN

Indexing


Facilitates
efficient data retrieval
Query performance.
Full-text search capabilities with
GIN and GiST indexes.

Foreign Data Wrappers (FDW)

Allows integration with

other databases and data sources.

Enables querying external data

as if it were a part of

the PostgreSQL database.

Full-Text Search

Built-in support for

full-text search capabilities.

Indexing and querying of

text data for fast search results

Replication and High Availability

- Supports both
- Synchronous
- Asynchronous
- Replication

Replication and High Availability

STREAMING REPLICATION

REAL-TIME DATA REDUNDANCY

LOGICAL REPLICATION

SELECTIVE DATA SYNCHRONIZATION

Partitioning

Declarative partitioning

for efficient management

of large tables.

Partitioning

Supports

Range,

List,

Hash

composite partitioning methods.

Security

Robust access control mechanisms

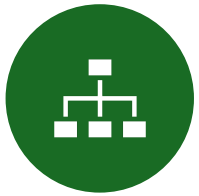
with roles and permissions.

SSL support for

secure client-server communication.

Security

Advanced features like



row-level security and

data encryption.

Performance Optimization

Parallel Query execution

for faster processing

of large datasets.

Performance Optimization

Query planner

Optimizer

for efficient

Query execution

Performance Optimization

CACHING MECHANISMS

TO IMPROVE PERFORMANCE.

Backup and Recovery

Comprehensive backup and

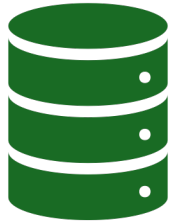
restore options with

tools like pg_dump and

pg_basebackup.

Backup and Recovery

Point-in-time recovery (PITR)



for restoring databases

to a specific state.

Benefits of PostgreSQL

Open Source and Free

PostgreSQL is open source,

allowing for free usage,

modification, and distribution.

No licensing costs,

making it cost-effective

for organizations

Flexibility

Highly extensible and

adaptable to various use cases.

Suitable for a wide range of applications,

from small-scale to

large-scale enterprise systems.

Community and Support

Strong, active community

providing extensive documentation,

support, and regular updates.

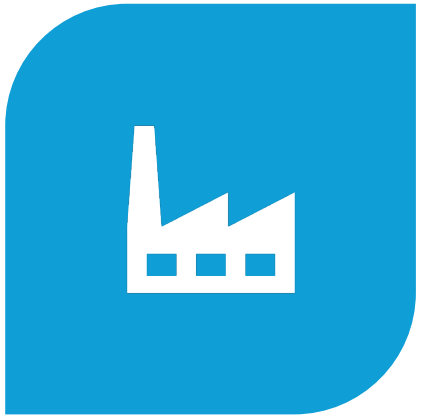
Numerous third-party tools and

extensions available.

Reliability and Stability

PROVEN TRACK RECORD OF

RELIABILITY AND STABILITY IN

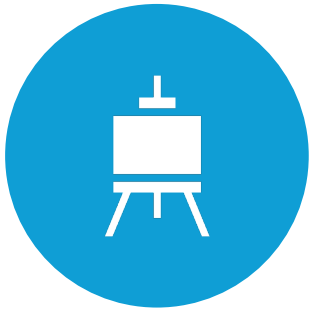


PRODUCTION ENVIRONMENTS.

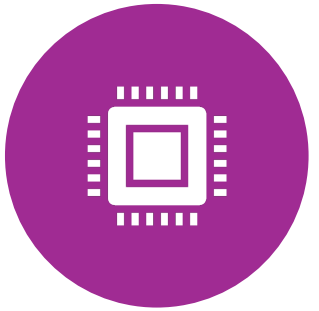
Standards Compliance

ADHERES TO SQL STANDARDS,

ENSURING COMPATIBILITY AND



EASE OF USE FOR DEVELOPERS



FAMILIAR WITH SQL.

Standards Compliance

REGULAR UPDATES

TO MAINTAIN COMPLIANCE

WITH EVOLVING STANDARDS.

Cross-Platform Compatibility

Runs on various operating systems

including Windows, macOS, and Linux.

Supports integration with

Various programming

languages and frameworks.

Performance

Advanced indexing and

query optimization techniques

ensure high performance.

Scalability to handle

large volumes of data and

high transaction loads.

Data Integrity

Strong support for data integrity

through constraints,

triggers, and foreign keys.

Ensures accurate and

consistent data storage and

retrieval.

Geospatial Capabilities

PostGIS extension

provides robust support

for geographic information systems (GIS).

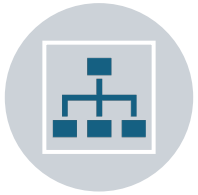
Ideal for applications

requiring spatial data management.

Enterprise Features

Comprehensive feature set

that meets the needs of



enterprise-level applications.

Support for complex queries,

transactions

data analytics.

Agenda

PostgreSQL vs Oracle: Key architectural differences

PostgreSQL installation & tools (pgAdmin, DBeaver)

Overview of pg_catalog, schemas, roles, and databases

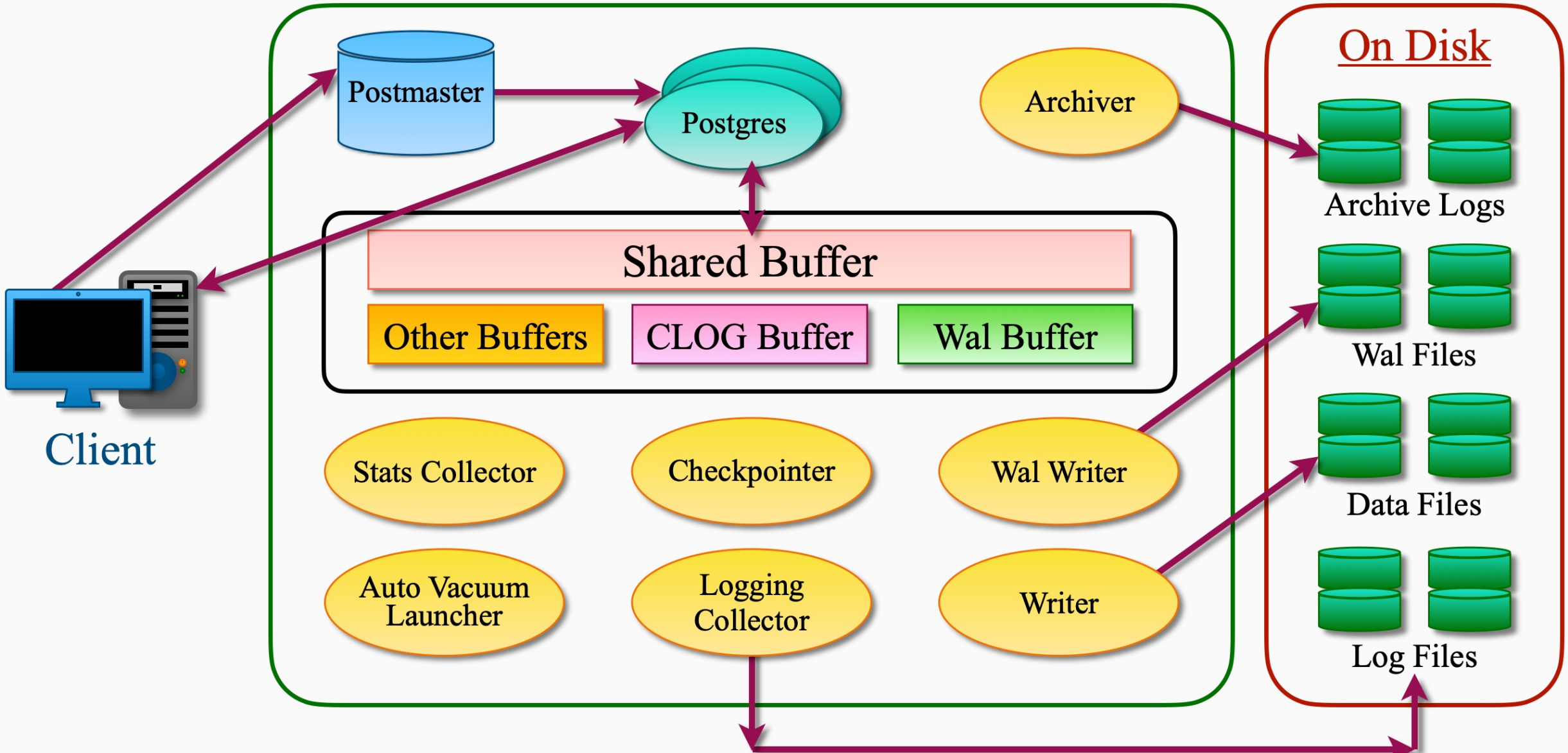
Hands-On

Install PostgreSQL & pgAdmin

Connect to database, explore catalogs

PostgreSQL vs Oracle: Key architectural differences

PostgreSQL Process and Memory Architecture



PostgreSQL Architecture Key components

1. Postmaster Process (Supervisor)
2. Shared Memory Segments
3. Background Processes
4. Physical File Structure

1. Postmaster Process (Supervisor)

The **first process**

Started when PostgreSQL launches

1. Postmaster Process (Supervisor)

Acts as a **listener**, handles:

Authentication

Authorization

Spawning new backend processes

Monitoring and restarting failed processes

2. Shared Memory Segments

Memory areas used for

query execution,

data caching, and

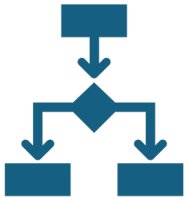
maintenance tasks.

Shared Buffers

Used for all **read/write operations**

Modified (dirty) pages are written to disk

by the **background writer**



Controlled by: shared_buffers

WAL Buffers (Write-Ahead Logging)

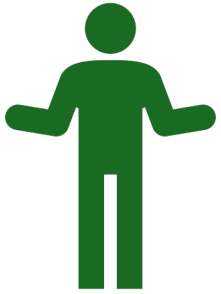
Temporarily store **metadata of changes**

Ensures **durability** and **crash recovery**

Controlled by: wal_buffers

CLOG Buffers (Commit Logs)

Track **transaction status**



(committed, aborted, etc.)

Shared across all sessions

Work Memory

Used for **sorting, hashing, joins**

Controlled by: work_mem

Separate allocation for each operation

Maintenance Work Memory

Used for **index creation,**

foreign key additions, etc.

Controlled by:

maintenance_work_mem

Temp Buffers

Used by **temporary tables**

Session-specific

3. Background Processes

Ensure

data consistency

help

maintain database health.

Background Writer

Writes

dirty pages

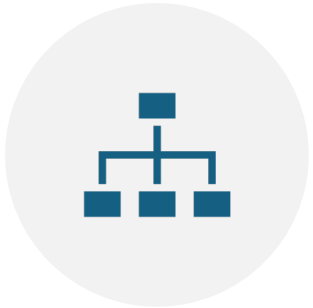
from shared buffers

to disk

Background Writer

CONTROLS:

BGWRITER_DELAY



BGWRITER_LRU_MAXPAGES

BGWRITER_LRU_MULTIPLIER

Checkpointer

Periodically **flushes**

dirty pages to disk

Essential for

crash recovery

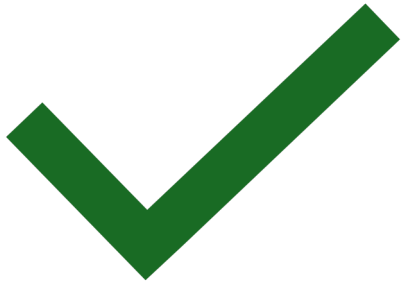
Checkpointer

Triggered by:

- Timeout (checkpoint_timeout)
- Manual CHECKPOINT command
- Backup processes
- max_wal_size limit

WAL Writer

Writes data from **WAL buffer to WAL files** on disk



Ensures write operations are **persistent before commit**

Autovacuum Launcher

Automates **VACUUM** and **ANALYZE**

Maintains **table statistics and bloat control**

Statistics Collector

Gathers usage metrics:

- Query counts
- Table scans
- Function usage

Statistics Collector

Controlled by:

- track_activities
- track_functions
- track_counts
- track_io_timing

Logging Collector

Captures server logs to files

Controlled by:

logging_collector = ON

log_directory = 'pg_log'

Archiver

Saves WAL segments to archive storage

for point-in-time recovery

Enabled during archive mode

4. Physical File Structure

PostgreSQL uses several file types

to store and manage data persistently:

Data Files

Store **actual user data**

Accessed via shared buffers

for reading/writing

WAL Files

Store **change logs** (Write-Ahead Logging)

Used for crash recovery

Log Files

Store **server logs** including

errors,

warnings, and

execution info

Archive Logs

Archived WAL segments

for **recovery purposes**

Workflow Summary

Client sends a query ➡️ received by Postmaster

Postmaster forks a backend (Postgres) process

Data retrieved from Shared Buffers

(or disk if not cached)

Workflow Summary

If **write operation**:

Change goes to Shared Buffer → WAL Buffer

WAL Writer commits to WAL file

Background Writer/Checkpointer writes to disk

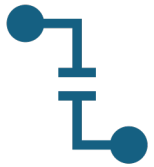
Statistics collected → Logging → Archiving if enabled

Conclusion

PostgreSQL's architecture is

robust, scalable, and

designed for high reliability,



fault-tolerance, and concurrency.

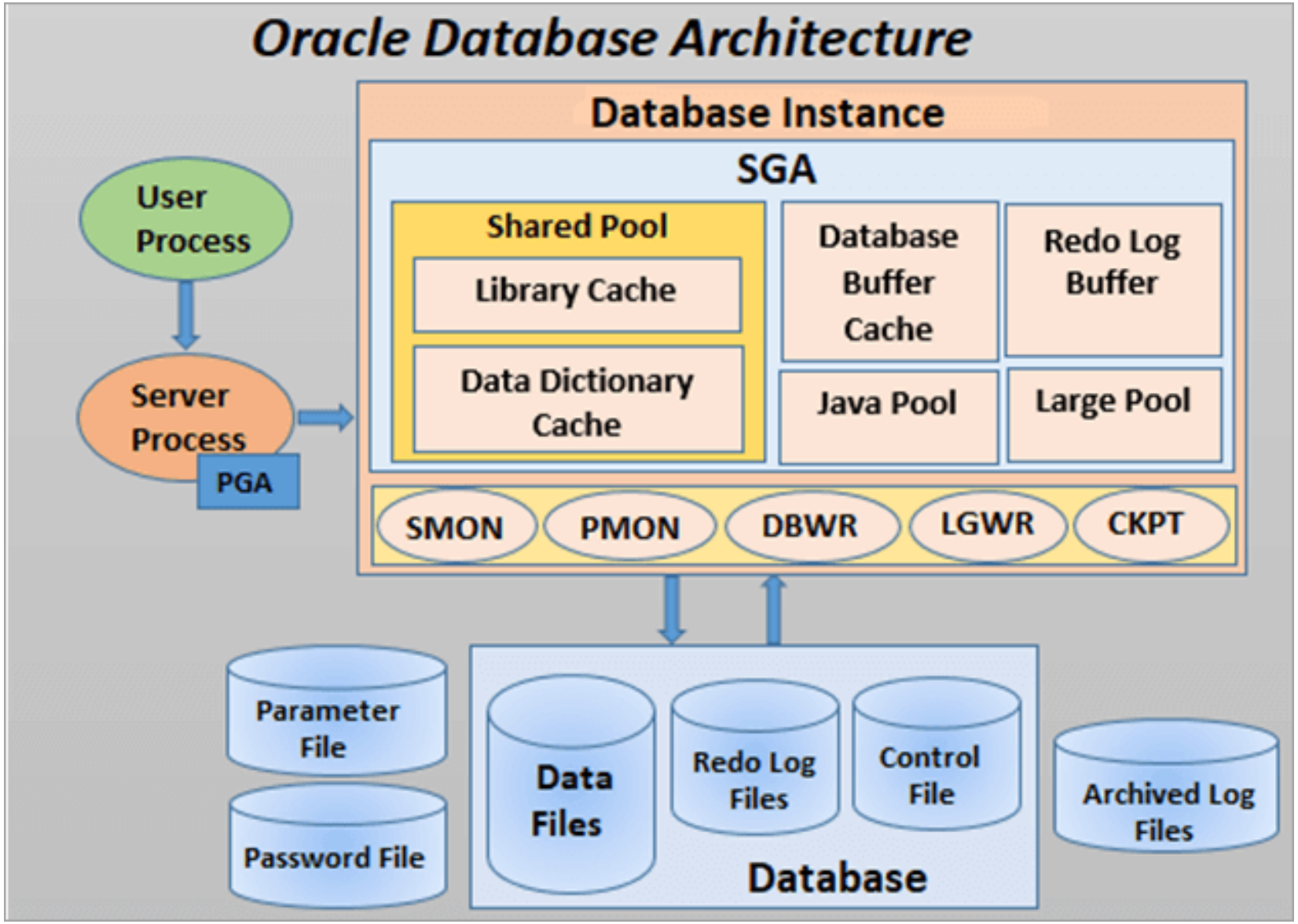
Conclusion

Its modular background processes and

memory components

ensure smooth operations and quick recovery,

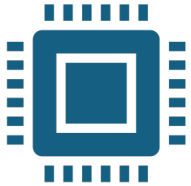
making it a strong choice for modern applications.


Oracle Database Architecture
Database Instance
SGA
User Process
Server Process
PGA
Shared Pool
Library Cache
Data Dictionary Cache
Database Buffer Cache
Redo Log Buffer
Java Pool
Large Pool
SMON
PMON
DBWR
LGWR
CKPT
Parameter File
Password File
Data Files
Redo Log Files
Control File
Database
Archived Log Files

Oracle Database Architecture

Oracle Database is a **multi-model**

relational database management system



known for its scalability, robustness, and

enterprise-grade features.

Oracle Database Architecture

Designed to manage

large amounts of data efficiently

using a combination of memory,

processes, files, and logical storage structures.

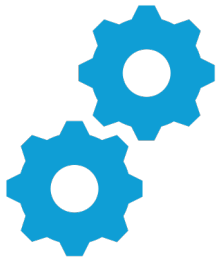
Oracle Database Key Components

- Oracle Instance (Memory + Processes)
- Physical Storage Structures
- Processes
- Logical Storage Structures

1. Oracle Instance (Memory + Processes)

Combination of:

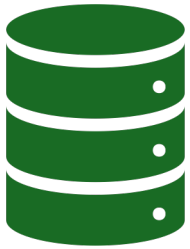
Memory Structures
(SGA & PGA)



Background Processes

1. Oracle Instance (Memory + Processes)

The instance manages



data stored in physical files and

serves client requests.

1.1 Memory Structures

A. SGA (System Global Area) – Shared memory

Allocated when the instance starts,

shared among all server/background processes.

Key components

Database Buffer Cache:

Caches recently used data blocks.

Redo Log Buffer:

Stores redo entries (change vectors) for data recovery.

Key components

Shared Pool:

Caches SQL statements, PL/SQL code, and metadata.

Library Cache:

Part of Shared Pool; stores parsed SQL and PL/SQL.

Key components

Data Dictionary Cache:

Stores definitions of tables, users, privileges.

Large Pool:

Used for large memory operations (e.g., parallel queries).

Key components

Java Pool: Used for Java-based stored procedures.

Controlled by SGA_MAX_SIZE and

init.ora/spfile.ora parameters.

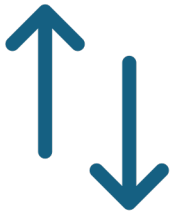
B. PGA (Program Global Area)

Private memory

Dedicated to each server process

B. PGA (Program Global Area)

Stores session-specific data like:



Sort areas

Cursor state

Session variables

Background Processes

Are essential daemons

that help manage memory,

I/O, logging, and

recovery

Mandatory Background Processes

SMON (System Monitor):

Performs crash recovery, temp space cleanup, etc.

PMON (Process Monitor):

Cleans up failed user processes and resources.

Mandatory Background Processes

DBWR (Database Writer):

Writes dirty buffers from memory to datafiles.

LGWR (Log Writer):

Writes redo log buffer to redo log files.

Mandatory Background Processes

CKPT (Checkpoint Process):

Signals DBWR and updates control & data files.

Optional Background Processes

ARCn (Archiver):

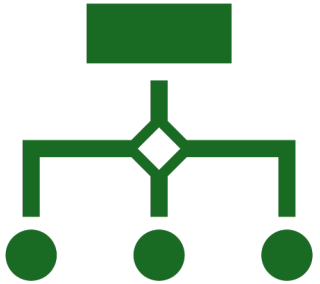
Archives redo logs for recovery.

CJQ0 (Job Queue Coordinator):

Manages scheduled jobs.

Optional Background Processes

SMCO (Space Management Coordinator):



Handles tasks like shrinking segments.

2. Physical Storage Structures

File Type	Description
Data Files	Store actual table/index data.
Redo Log Files	Store all changes made to the DB, used for recovery.
Control Files	Contain metadata about the database structure and state.
Archived Redo Logs	Backups of redo logs used for point-in-time recovery.
Parameter File (PFILE/SPFILE)	Configuration file for instance startup.
Password File	Stores authentication info for privileged users.

3. Processes

Processes manage

interaction between

users and the database engine.

3.1 User Process

Initiated when a user connects

Handles communication

between client and server

3.2 Server Process

Executes SQL queries

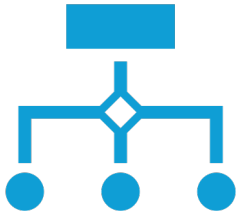
on behalf of the user process

Reads/writes data to memory or files

3.3 Background Processes

Run automatically

to handle tasks like recovery,



buffer management, etc.

4. Logical Storage Structures

Structure	Description
Tablespace	Logical storage unit composed of one or more data files
Segment	Set of extents used to store a specific database object
Extent	Group of contiguous data blocks
Data Block	Smallest unit of storage, defined by DB_BLOCK_SIZE (e.g., 8KB)

Oracle Workflow (Simplified)

User connects → User & Server processes are created

Query Execution:

SQL parsed and cached (Shared Pool)

Data fetched from **Buffer Cache** (or disk if not cached)

Query Execution

Modifications → written to **Buffer Cache** and **Redo Log Buffer**

LGWR writes Redo Buffer to Redo Logs

DBWR writes changes to Data Files

(on checkpoint or buffer pressure)

Recovery (if needed):

Redo logs and

archived logs

used to restore DB

Summary Table

Component	Details
Instance	SGA + PGA + Background Processes
Memory Structures	SGA (shared), PGA (private)
Mandatory Processes	SMON, PMON, DBWR, LGWR, CKPT
Physical Files	Data Files, Redo Logs, Control Files
Logical Units	Tablespaces → Segments → Extents → Data Blocks

Why Compare PostgreSQL & Oracle?

Understanding architectural differences helps in:

Migration planning

Cost-benefit analysis

Technology stack decisions

License & Philosophy

Feature	PostgreSQL	Oracle
License	Open Source (PostgreSQL License - similar to BSD)	Commercial (Proprietary License)
Philosophy	Extensibility, Standards Compliance, Community-Driven	Feature-Rich, Enterprise-Grade, Vendor Lock-in

Architecture Overview

Aspect	PostgreSQL	Oracle Database
Process Model	Multi-Process (One process per client connection + background processes)	Multi-Threaded (Single process with multiple threads + background processes)
Memory Architecture	Shared Buffers, Work Mem, WAL Buffers	SGA (System Global Area), PGA (Program Global Area), Redo Log Buffer

Architecture Overview

Aspect	PostgreSQL	Oracle Database
Storage Engine	Single storage engine	Multiple engines (Heap Organized Tables, Index Organized Tables, etc.)
Extensibility	Highly extensible (user-defined types, operators, functions, custom languages)	Limited extensibility (supports PL/SQL, but not new data types easily)

Transaction & Concurrency Control

Feature	PostgreSQL	Oracle
MVCC (Multi-Version Concurrency Control)	Native MVCC (Row-versioning, no read locks for SELECTs)	MVCC via Undo Segments (Undo data in rollback segments)
Locking	Row-level locks, advisory locks	Row-level locks, fine-grained locking, table-level partition locks
Isolation Levels	Read Committed, Repeatable Read, Serializable	Read Committed, Serializable, Read-Only Transactions

Replication & High Availability

Feature	PostgreSQL	Oracle
Replication	Streaming Replication, Logical Replication (built-in), 3rd party tools (e.g., Bucardo)	Oracle Data Guard (physical), Oracle GoldenGate (logical)
Sharding	External tools (Citus, pg_shard)	Native (Oracle Sharding)

Data Types & SQL Compliance

Feature	PostgreSQL	Oracle
Data Types	Rich: JSON/JSONB, Arrays, HSTORE, Range Types	Limited: JSON support added in 12c but less flexible
Standards	Very close to ANSI SQL compliance	SQL compliance + proprietary PL/SQL

Partitioning

Feature	PostgreSQL	Oracle
Partitioning	Declarative partitioning (since PostgreSQL 10)	Advanced partitioning (list, range, hash, composite)
Management	Manual in earlier versions; improved automation in recent versions	Fully automated, advanced features

Backup & Recovery

Feature	PostgreSQL	Oracle
Backup	pg_dump (logical), base backups + WAL archiving	RMAN (block-level backup), Data Pump (logical export/import)
Point-in-Time Recovery	WAL-based PITR	Redo/Undo Logs, Flashback Database

PL/SQL vs PL/pgSQL

Feature	PostgreSQL	Oracle
Procedural Language	PL/pgSQL (open), also supports Python, Perl, etc.	PL/SQL (proprietary, tightly integrated)
Triggers/Functions	Easily create functions in multiple languages	Primarily in PL/SQL

Performance & Optimization

Feature	PostgreSQL	Oracle
Optimizer	Cost-based, extensible, adaptive plans in progress	Highly advanced Cost-based Optimizer, adaptive plans, hints
Parallel Query	Supported but less mature	Advanced, with fine-grained control

Security & Auditing

Feature	PostgreSQL	Oracle
Authentication	SSL, GSSAPI, LDAP, Kerberos	Advanced (including fine-grained auditing, TDE, DB Vault)
Auditing	Basic logging and extensions (e.g., pgaudit)	Comprehensive built-in auditing

When to Use What?

PostgreSQL	Oracle
Startups, Open Source Projects	Large Enterprises, Legacy Systems
Cost-sensitive environments	High compliance, mission-critical
Customization & Extensions	Advanced features, support

Summary of Key Differences

PostgreSQL	Oracle
Open-source, community-driven	Proprietary, enterprise-focused
Process-based architecture	Thread-based shared memory architecture
Extensible with custom data types	Fixed feature set, but rich enterprise features
MVCC via row versioning	MVCC via undo segments
Free, with optional paid support	License-based, expensive but feature-rich

Final Thoughts

PostgreSQL: Innovation, Freedom, Extensibility 🚀

Oracle: Stability, Enterprise Support, Advanced Features 🏢

Choose based on your needs, not hype!

PostgreSQL 16 Installation

Surendra Panpaliya

Download PostgreSQL Installer

- Visit: https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
- Select: **Windows → PostgreSQL 16**

Run the Installer

Double-click the .exe file

Follow the Setup Wizard

Select components:

PostgreSQL Server ✅

pgAdmin 4 ✅

Stack Builder (optional)

Set Password & Configure Port

Enter	Enter a superuser password
Configure	Configure Port
Default	Default port: 5432
Change	You can change if needed.

Choose Data Directory

Default: C:\Program Files\PostgreSQL\16\data

Finish Installation

macOS Installation

- **Use EnterpriseDB Installer**
- Download from: https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
- Run the .dmg installer

Connect to PostgreSQL

psql postgres

Default User:

Username: **your system user**

No password by default (set manually if needed)

Install pgAdmin 4

Download pgAdmin

Visit: https://www.pgadmin.org/download/

Choose **Windows / macOS**

Run Installer

Install pgAdmin4

Set **email and master password** (for local access)

Add PostgreSQL Server

Open pgAdmin

Right-click → Register → Server

Input:

Name: Local PostgreSQL

Host: localhost

Port: 5432

Username: postgres

Password: (your password from install step)

Install DBeaver (Universal DB Tool)

Download DBeaver

Visit: https://dbeaver.io/download/

Step 2:

Install DBeaver

For **Windows**: Run .exe installer

For **macOS**: Run .dmg file, drag to Applications

Connect to PostgreSQL

Open DBeaver

Click: **Database → New Connection**

Select **PostgreSQL**

Fill in:

Host: localhost

Port: 5432

Connect to PostgreSQL

Database: postgres

Username: postgres

Password: your password

Test Connection & Finish

Summary

Tool	Purpose
PostgreSQL 16	Database Server
pgAdmin 4	Web-based GUI for PostgreSQL
DBeaver	Universal SQL Client & ER Diagram Tool

Useful Links

PostgreSQL Docs: https://www.postgresql.org/docs/16/

pgAdmin Docs: https://www.pgadmin.org/docs/

DBeaver Docs: https://dbeaver.io/docs/

Overview of pg_catalog, schemas, roles, and databases

Surendra Panpaliya

Databases in PostgreSQL

Self-contained unit of data.

Each database has its own

schemas, tables,

users (roles), and configurations

Key Points

PostgreSQL is a multi-database system.

Each database is isolated from others

(no cross-database queries directly).

Default databases

postgres → Default working database

template1 → Template for new databases

template0 → Clean template (minimal)

Schemas

Namespace inside a database.

Think of it as a folder to organize

tables, views, functions, etc.

Key Points

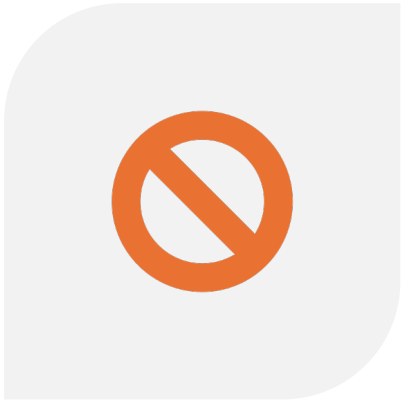
Each database can have **multiple schemas**.

Default schema: public

Objects are referenced as:

schema_name.object_name

Why Schemas?



AVOID NAME CONFLICTS

LOGICAL SEPARATION OF DATA

SUPPORT FOR MULTI-TENANT APPLICATIONS

Command Examples

```
-- Create a schema
CREATE SCHEMA sales;

-- Create table inside schema
CREATE TABLE sales.orders (id SERIAL PRIMARY KEY, item TEXT);

-- Access table
SELECT * FROM sales.orders;
```

pg_catalog

A system schema

Stores PostgreSQL

internal metadata.

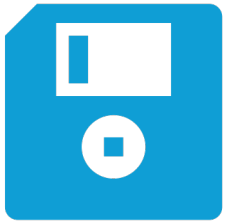
Contains

Object	Purpose
pg_class	Info about tables, indexes, sequences
pg_user	View of roles with login
pg_roles	Info about all roles
pg_namespace	Info about schemas
pg_tables	List of tables
pg_indexes	List of indexes

Use Case:

Query database metadata

Understand internal PostgreSQL structures



System operations, backups, monitoring

Example Queries:

```
-- List all tables using pg_catalog
SELECT tablename FROM pg_catalog.pg_tables WHERE schemaname = 'public';


-- List all roles
SELECT rolname FROM pg_catalog.pg_roles;
```

Roles

A **Role** in PostgreSQL = **User Account + Group**

Roles manage **authentication & authorization**.

Types of Roles:

Role	Description
Login Role	Can log in (CREATE ROLE username LOGIN;)
Group Role	Used for permissions, no login

Privileges Controlled By Roles:

CONNECT	CONNECT (to database)
CREATE	CREATE (schemas, tables)
SELECT	SELECT, INSERT, UPDATE, DELETE (on tables)
EXECUTE	EXECUTE (functions)

Role Management Examples

-- Create role with login

CREATE ROLE devuser LOGIN PASSWORD 'mypassword';

-- Grant role permissions

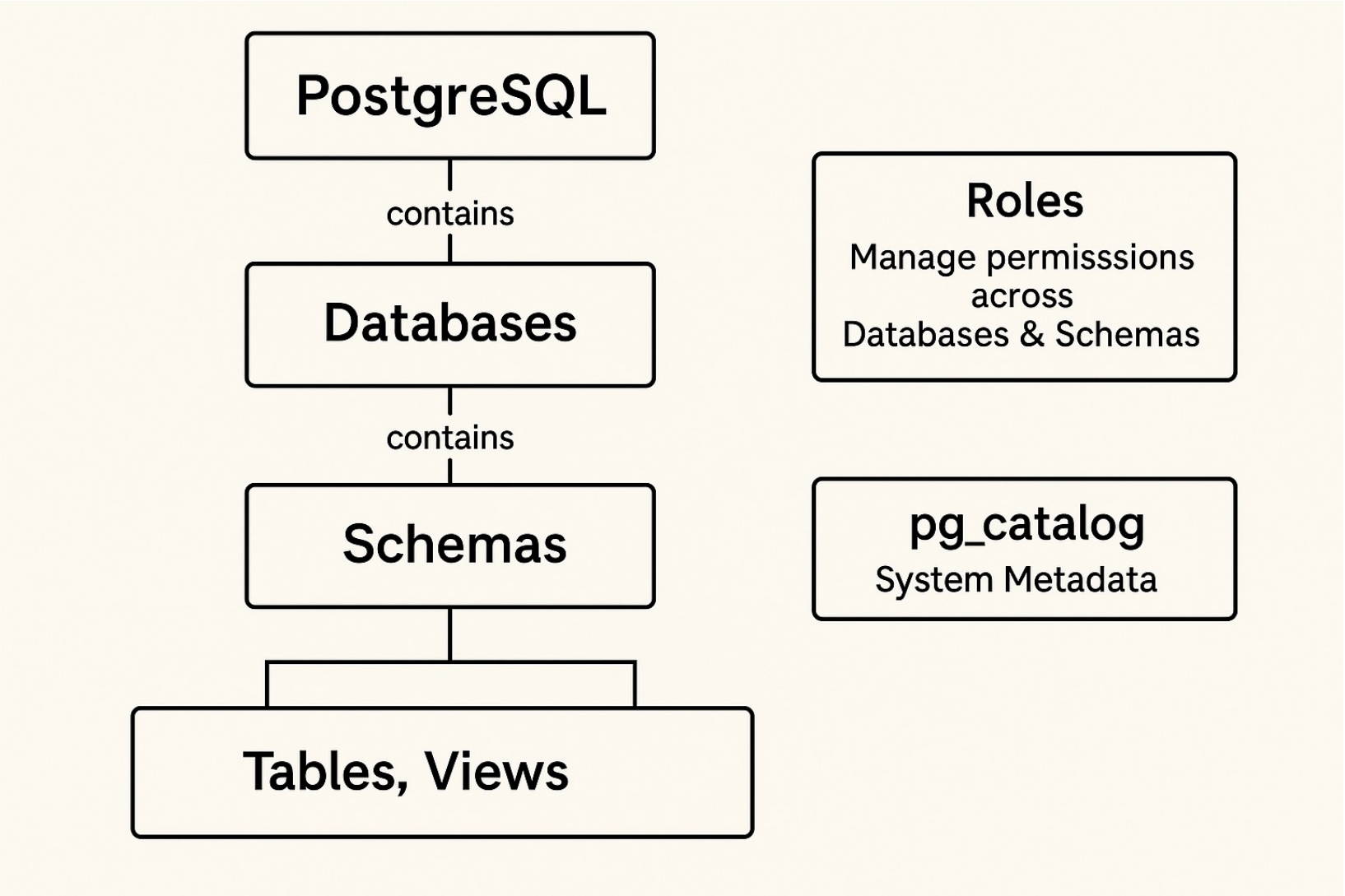
GRANT CREATE ON DATABASE mydb TO devuser;

Role Management Examples

```
-- Create group role
CREATE ROLE analyst;

-- Assign user to group
GRANT analyst TO devuser;
```

Relationship Diagram



Summary Table

Concept	Purpose
Database	Top-level data container
Schema	Namespace within a database
pg_catalog	Stores metadata about objects
Role	User and permission management

Real-World Analogy

PostgreSQL Concept	Analogy
Database	Apartment Building
Schema	Individual Apartment
Table/View	Furniture inside an apartment
Role	Security Guard controlling access
pg_catalog	Building Management Records

Surendra Panpaliya

Founder and CEO

GKTCS Innovations

https://www.gktcs.com