Database Development with PL/SQL INSY 8311

PLSQL
PL/SQL Language

ADVENTIST UNIVERSITY OF CENTRAL AFRICA

Instructor:

- Eric Maniraguha | eric.maniraguha@auca.ac.rw | LinkedIn Profile

6h00 pm – 9h50 pm

- Monday A –G209
- Tuesday B-G207
- Wednesday C-G207
- Thursday D-G310

September 2025

Database Development with PL/SQL

PLSQL
PL/SQL Language

Reference reading

- What is PL/SQL?
- Oracle PL/SQL Articles
- Difference Between SQL and PLSQL
- Real Application Clusters (RAC) Articles
- Oracle SQL Queries: Basic Exercises with Solution
- Autonomicity in Oracle Database Management System
- The Toughest Challenges and Problems with Oracle Database

What is PL/SQL?

PLSQL
PL/SQL Language

What?

PL/SQL is a procedural relational database language developed by **Oracle Corporation** in earlier 90's.

PL/SQL – Procedural Language Structured Query Language.

"PL/SQL allows the execution of a block of code at a time which increases it performance."

Procedural language?

Procedural programming means writing instructions in a **step-by-step (procedure) way**.
It tells the computer *how* to do something, not just *what* you want.

It's the opposite of **declarative** style (like plain SQL), where you just describe the result you want.

Course Overview

PLSQL
PL/SQL Language

Starting out with Oracle

This course introduces Oracle's database environment**, covering SQL, PL/SQL, and SQL*Plus.** You'll learn their fundamentals, interactions, and applications for managing and manipulating relational databases.

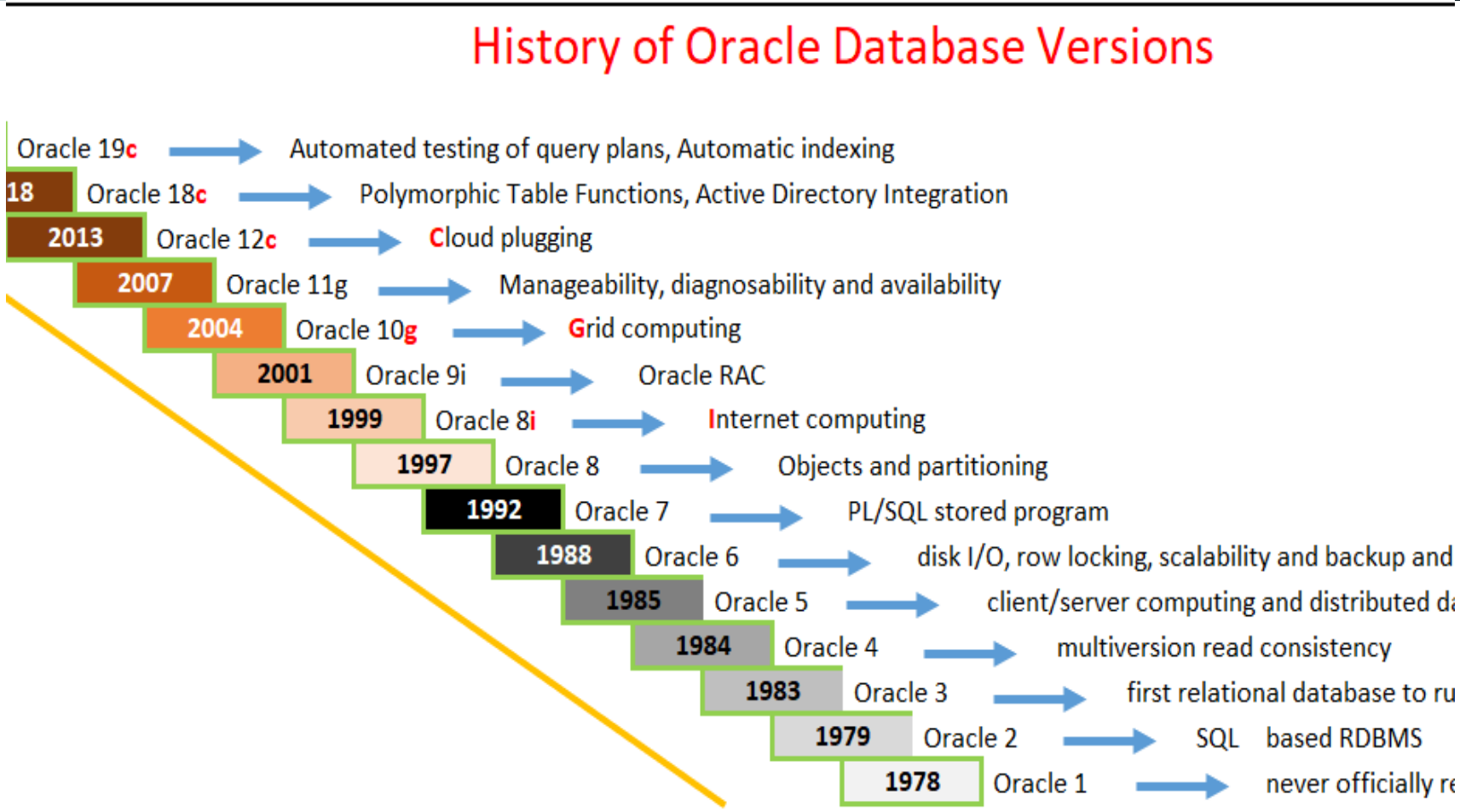
Covering Databases, SQL, PL/SQL, Developer Tools and DBA

Feature	SQL	PL/SQL	SQL*Plus
Type	Query language	Procedural extension	Command-line interface
Purpose	Data manipulation	Logic and automation	Executing SQL/PL/SQL commands
Execution	Directly on database	Runs as blocks/procs	Executes SQL/PLSQL commands
Portability	Cross-platform	Oracle-specific	Oracle-specific
Use Case	Query data	Implement logic	Manage and execute database tasks

Brief History of Oracle Database

Oracle Database is a relational database management system (RDBMS) developed by Oracle Corporation, one of the world's leading providers of database solutions. Founded in **1977** by **Larry Ellison**, **Bob Miner**, and **Ed Oates**, Oracle Corporation began as Software Development Laboratories (SDL). The company revolutionized the database industry by introducing the first commercially available **SQL-based RDBMS** with Oracle V2 in 1979.

Over the years, Oracle Database has become synonymous with innovation, scalability, and reliability, evolving to meet the needs of businesses and modern computing environments.



Source Image: https://ittutorial.org/oracle-version-history-oracle-database-release-versions/

Oracle Database Latest Versions and Features

PLSQL
PL/SQL Language

Oracle Database 19c (2019)

- **Significance**: Advanced hybrid partitioning and real-time statistics.
- **Features**:
  - Improved **Autonomous Database** capabilities.
  - Enhanced PL/SQL functionality.

Oracle Database 21c (2021)

- **Significance**: Emerging technologies and modern data management.
- **Features**:
  - Introduced **Blockchain Tables** for immutable records.
  - Added **Native JSON Datatypes** for better NoSQL integration.

Oracle Database 23c AI (2023)

Significance: Integration of Artificial Intelligence and developer-friendly tools.

Features:

- Transpiler for converting Python and JavaScript into PL/SQL.
- AI-driven automation for query tuning and anomaly detection.
- In-database AI/ML model training and execution.

Oracle Database 24c AI (2024)

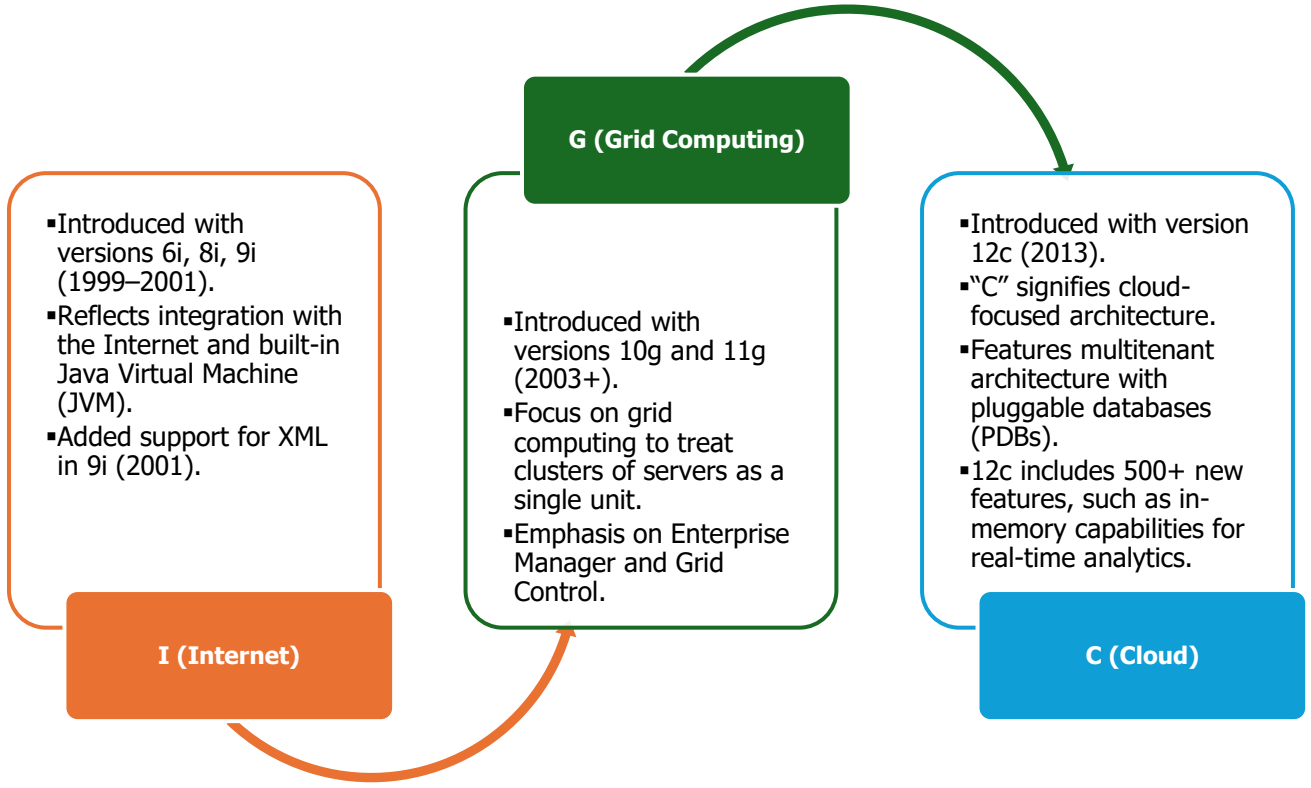
Significance: Advanced real-time AI capabilities and enhanced productivity.

Features:

- Improved real-time AI-driven insights and decision-making.
- Expanded Transpiler support for more programming languages.
- Enhanced AI algorithms for predictive modeling and analytics.

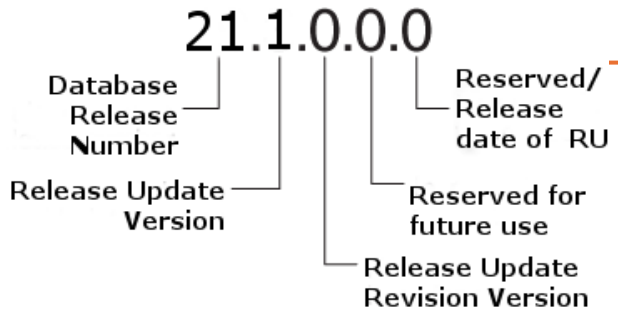
Meaning of "I," "G," and "C" in Oracle Versions & Oracle Version Numbers

PLSQL PL/SQL Language



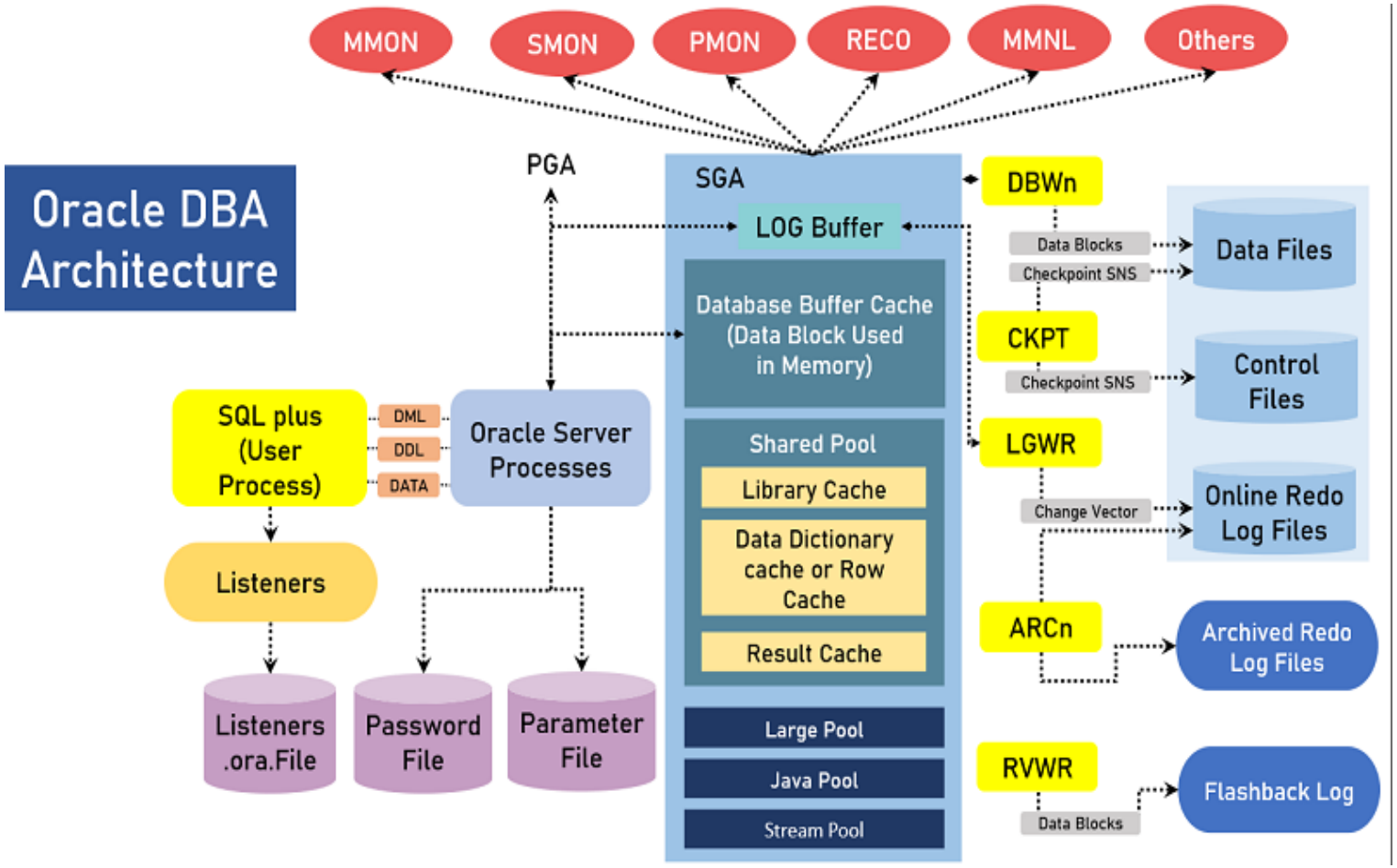
Oracle version numbers follow the format: **X.Y.Z.P.Q**

1. **X** – **Major Release**:
   Indicates significant new features and architectural changes.
2. **Y** – **Maintenance Release**:
   Includes enhancements and optimizations.
3. **Z** – **Application Server Release Number**:
   Minor updates or bug fixes for the application server.
4. **P** – **Patch Release Number**:
   Fixes specific bugs and security vulnerabilities.
5. **Q** – **Platform-Specific Patch Level**:
   Updates for specific hardware or OS platforms.



Source Image: Description of "Figure 1-2 Example of an Oracle Database Release Number"

Oracle Architecture workflow



How it all works together (Flow)

1. A user sends a SQL request → goes through **Listener** → reaches **Oracle server process**.
2. Data request is checked in **SGA** (cache). If not found, it's read from **data files** on disk.
3. Changes made are first written to **Log Buffer**, then persisted in **Redo Log Files**.
4. Background processes (DBWn, LGWR, CKPT, ARCn, RVWR) ensure data is written, safe, and recoverable.
5. The result is returned to the user.

Source Image: https://learnomate.org/oracle-dba-architecture/

Oracle Architecture workflow - Explanation

PLSQL
PL/SQL Language

1. User Interaction Layer

- **SQL Plus (User Process):**
  Users (DBAs, developers, applications) send requests in the form of SQL statements (DML, DDL, or queries).
- **Listeners:**
  These are like *gatekeepers*. They listen for client requests and pass them to the database server.

2. Oracle Server Processes

- Once a request reaches Oracle, **server processes** are started.
- They manage user sessions, execute SQL, fetch data, and return results.
- There are also **Oracle background processes** (shown at the top: MMON, SMON, PMON, RECO, MMNL, etc.) that run automatically to keep the database healthy:
  - **PMON (Process Monitor):** Cleans up failed processes.
  - **SMON (System Monitor):** Recovers the database after a crash.
  - **MMON (Memory Monitor):** Monitors memory and statistics.
  - **RECO (Recoverer):** Handles distributed transactions.
  - **Others:** Do housekeeping jobs like logging, writing data, and checkpointing.

3. Memory Structures

Oracle uses two main memory areas:

a) PGA (Program Global Area):

- Dedicated to a single server process.
- Stores things like session info, sorting operations, etc.

b) SGA (System Global Area):

Shared across all processes, it's the main memory area of Oracle DB. Key components:

- **Log Buffer:** Temporary storage of redo (change) information before writing to redo log files.
- **Database Buffer Cache:** Holds copies of data blocks read from disk. Speeds up access by avoiding repeated disk I/O.
- **Shared Pool:** Stores parsed SQL, PL/SQL code, dictionary cache, and execution plans for reuse.
- **Result Cache:** Caches query results to avoid recomputation.
- **Large Pool, Java Pool, Stream Pool:** Support backup, recovery, Java execution, and streaming.

4. Background Processes and File Interaction

Oracle coordinates memory with physical files through background processes:

- **DBWn (Database Writer):** Writes modified data blocks from buffer cache to data files.
- **LGWR (Log Writer):** Writes redo log entries from log buffer to online redo log files.
- **CKPT (Checkpoint):** Signals DBWn to write dirty buffers, updates control files and data file headers.
- **ARCn (Archiver):** Copies redo log files to archive storage (important for recovery).
- **RVWR (Recovery Writer):** Writes data to flashback logs for fast recovery.

5. Physical Storage

Oracle stores data on disk in different types of files:

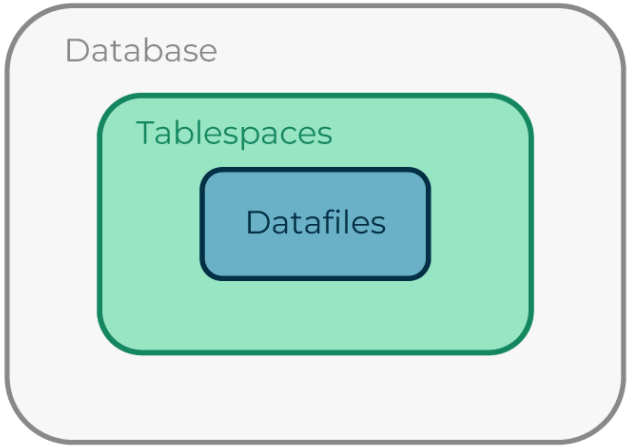
- **Data Files:** Store actual user/application data.
- **Control Files:** Store database structure and metadata.
- **Redo Log Files (Online):** Store records of changes for recovery.
- **Archived Redo Logs:** Backup copies of redo logs.
- **Flashback Logs:** Enable point-in-time recovery.

Oracle DB Environments: Tablespaces, Datafiles, and Control Files

PLSQL PL/SQL Language

Tablespaces are logical **storage units in Oracle Database used to store and organize data.** They enable efficient data retrieval by providing logical structures for managing and locating specific information.

A database, as defined by Oracle, is an organized collection of data, typically stored electronically. However, managing this data efficiently requires structured storage, which is where tablespaces come into play.

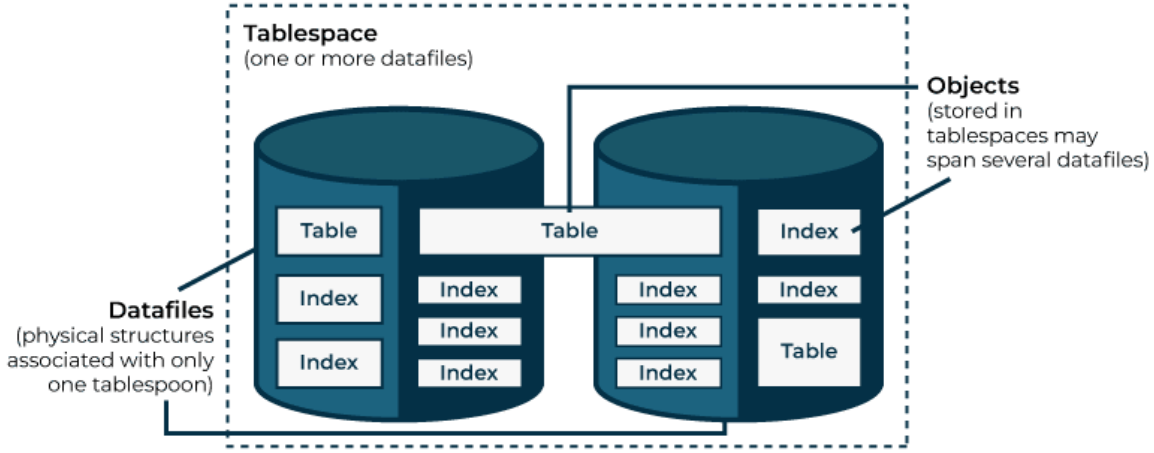


Source Image: https://www.geeksforgeeks.org/working-on-oracle-tablespaces-for-developers/

Type of Tablespace

SYSTEM Tablespace
- Automatically created during database creation.
- Stores the data dictionary tables for the entire database.

Undo Tablespaces
- Dedicated to storing undo information exclusively.



Database → Tablespaces → Datafiles
(Logical → Physical storage).

Key Points on Oracle Tablespaces

PLSQL PL/SQL Language

1. Logical vs Physical

- **Tablespace = Logical container** → groups related database objects (tables, indexes, etc.).
- **Datafiles = Physical files** on disk that actually store the data.
  👉 **Think**: Tablespace is like a *folder*, while datafiles are the *hard drive files* inside it.

2. Types of Tablespaces

- **SYSTEM**
  - Mandatory tablespace.
  - Stores the **data dictionary** (metadata about database objects).
- **SYSAUX**
  - Auxiliary to SYSTEM.
  - Reduces load on SYSTEM by holding performance data, tools, and features.
- **USER**
  - Stores objects created by users (tables, indexes, etc.).
- **TEMP**
  - Used for **temporary operations** like sorting, joins, or creating indexes.
  - Does not hold permanent data.
- **UNDO**
  - Stores undo information for **rollback, read-consistency, and recovery**.

3. Default Tablespace

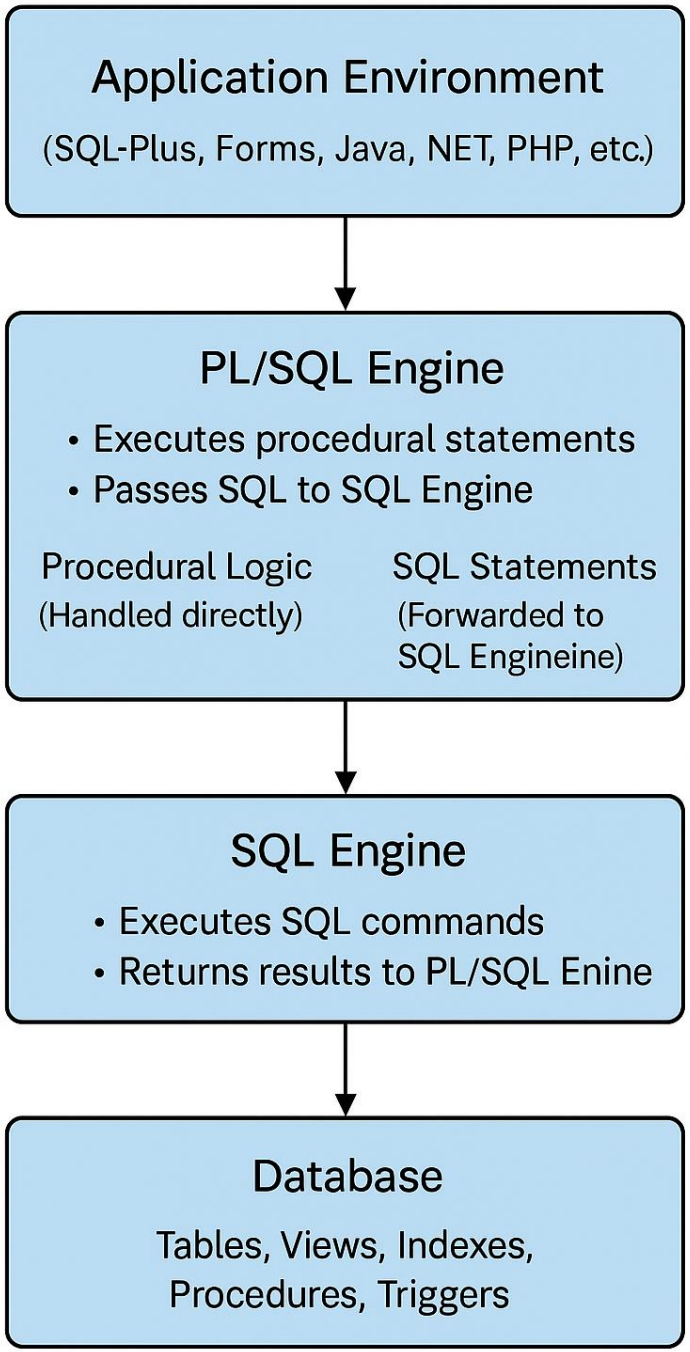
- If you don't specify a tablespace when creating an object → it automatically goes into the **user's default tablespace**.
- Best practice: assign users a specific default tablespace to avoid cluttering SYSTEM.

4. Why Important for Students

- Separates **logical management** from **physical storage**.
- Helps in **performance tuning** (TEMP, UNDO).
- Ensures **database stability** (SYSTEM, SYSAUX).
- Organizes **user data** (USER).

PLSQL
PL/SQL Language

PL/SQL Architecture Overview



PL/SQL sits between the **application** and the **Oracle database engine**, allowing SQL and procedural code to work together efficiently.
Main Components:

1. Application Environment
- User interfaces (like SQL*Plus, Oracle Forms, Java/PHP applications) that send PL/SQL blocks.

2. PL/SQL Engine
- Executes procedural statements (loops, conditions, exceptions).
- Passes SQL statements to the SQL engine.

3. SQL Engine
- Processes standard SQL statements (SELECT, INSERT, UPDATE, DELETE).
- Works together with the PL/SQL engine.

4. Database
- Stores tables, views, indexes, and PL/SQL objects (procedures, functions, packages, triggers).

Oracle Architecture workflow – Simple Terms

PLSQL
PL/SQL Language

Oracle architecture ensures **performance (fast memory caching)**, **reliability (redo logs & checkpoints)**, and **recoverability (archives & flashback logs)**.

1. **Performance – Fast Memory Caching**
   - **What it means:** Oracle keeps frequently used data in memory (SGA, especially the buffer cache and shared pool) so it doesn't always have to read from the slower disk.
   - **Example for students:** Imagine you're studying – instead of going to the library every time you need a book, you keep the important ones on your desk. You'll finish faster because you're not wasting time walking back and forth.
   - **In Oracle:** Data buffer cache and shared pool help SQL queries run faster.

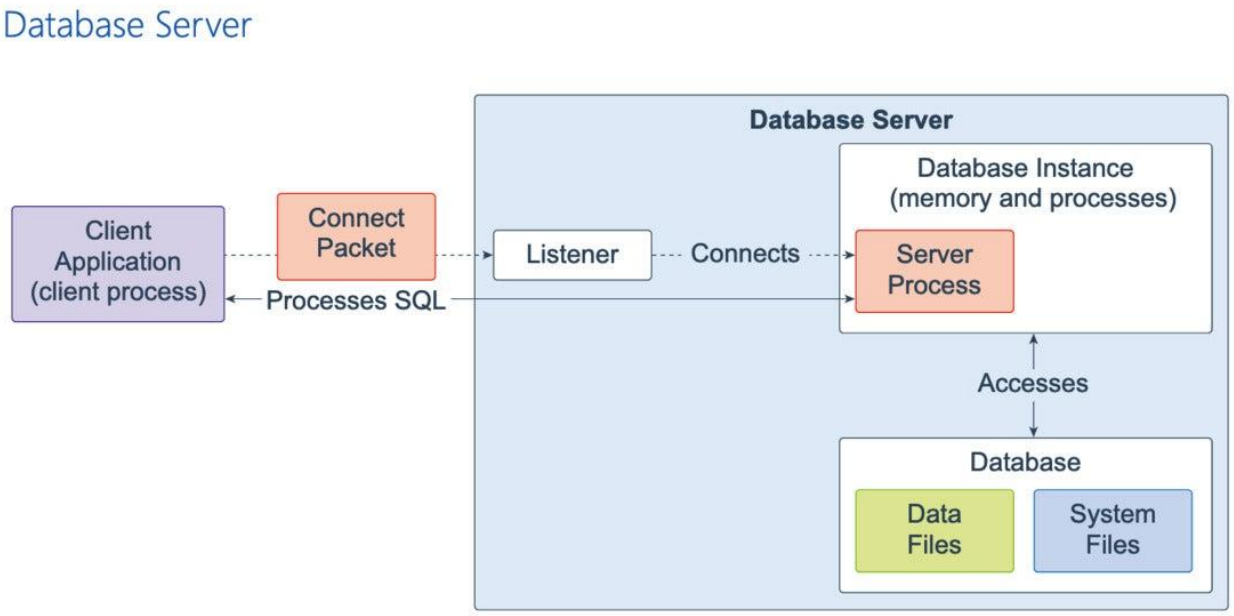
2. **Reliability – Redo Logs & Checkpoints**
   - **What it means:** Every change is first written into *redo logs* and checkpoints are used to mark safe positions. This ensures that even if the system crashes, no committed transaction is lost.
   - **Example for students:** Think of typing an assignment in Word. Even if your laptop suddenly shuts down, the *auto-save* (redo logs) ensures you don't lose your last edits. Checkpoints are like the "Save" button that marks a consistent point.
   - **In Oracle:** The **LGWR** writes redo logs, and **CKPT** ensures data files are updated safely.

3. **Recoverability – Archives & Flashback Logs**
   - **What it means:** Oracle keeps archived redo logs and flashback logs so you can restore the database to any past state if needed.
   - **Example for students:** Imagine Google Docs – you can open "Version History" and go back to how your file looked yesterday or last week. That's recoverability.
   - **In Oracle:**
     - **Archived redo logs** let DBAs restore the database after failure.
     - **Flashback logs** let you rewind to a previous moment (undo mistakes).

Oracle Database 21c Server – Simplified Explanation

PLSQL
PL/SQL Language

This Is The Highest-level Diagram. Note That The Listener Process Often Runs Outside The Server, Especially In Clustered Environments.



Source Image: https://www.infoworld.com/article/2264510/oracle-database-21c-review-the-old-rdbms-is-new-again.html

1. Client Application (Client Process):
The program used by end users to send SQL queries.
Sends a **connect packet** to request access to the database.

2. Connect Packet:
A small message that carries the connection request from the client to the server.

3. Listener:
A special process on the server that waits for incoming client requests.
Once a valid request arrives, it connects the client to a **server process**.
(Note: In clustered environments, the listener often runs outside the server.)

4. Database Server:
Contains two major parts:
Database Instance (memory + processes).
Database (physical storage).

5. Database Instance:
Manages interactions between the client and the database.
Includes the **server process**, which executes SQL queries.

6. Database (Physical Storage):
- **Data Files:** Store user data (tables, indexes, etc.).
- **System Files:** Store metadata, configurations, and database structure information.

What is an Oracle Instance?

PLSQL
PL/SQL Language

- An **Instance** is what runs when you start Oracle.
- An **Instance** works with the **Database** (the physical files such as datafiles, control files, redo logs).

You can have:
- **Single Instance Database** → one instance manages one database.
- **RAC (Real Application Clusters)** → multiple instances manage the same database for high availability.

👉 **In simple terms for students:**
- The **Database** is the data stored on disk.
- The **Instance** is the *active brain (memory + processes)* that manages and uses that data.

Oracle Database 21c Instance Diagram

PLSQL
PL/SQL Language

The Instance Handles Memory And Processes.

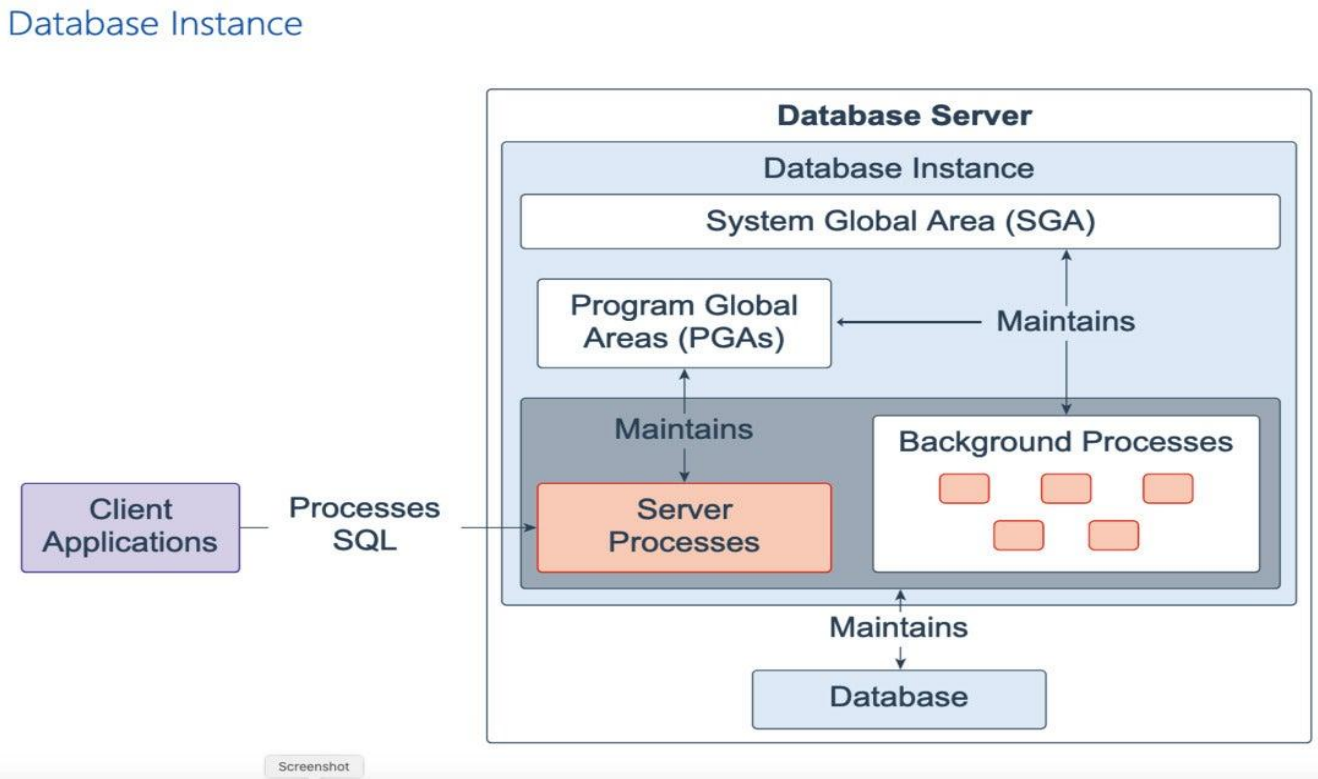


Diagram 2: Database Instance

1.Client Applications:
- Send SQL queries to the database instance.

2.Database Server:
- Includes the **Database Instance** and the physical **Database**.

3.Database Instance:
- Comprises memory structures and processes used to interact with the database.
- Key Components:
  1. **System Global Area (SGA)**:
     - Shared memory area that contains information shared across multiple server processes.
     - Stores data such as SQL query results, cached data, and execution plans.
  2. **Program Global Areas (PGAs)**:
     - Memory areas allocated for individual server processes.
     - Contains data specific to a single user or process.
  3. **Server Processes**:
     - Handle client requests by executing SQL queries and interacting with the database.
  4. **Background Processes**:
     - Perform maintenance tasks like managing data files, logs, and memory cleanup.

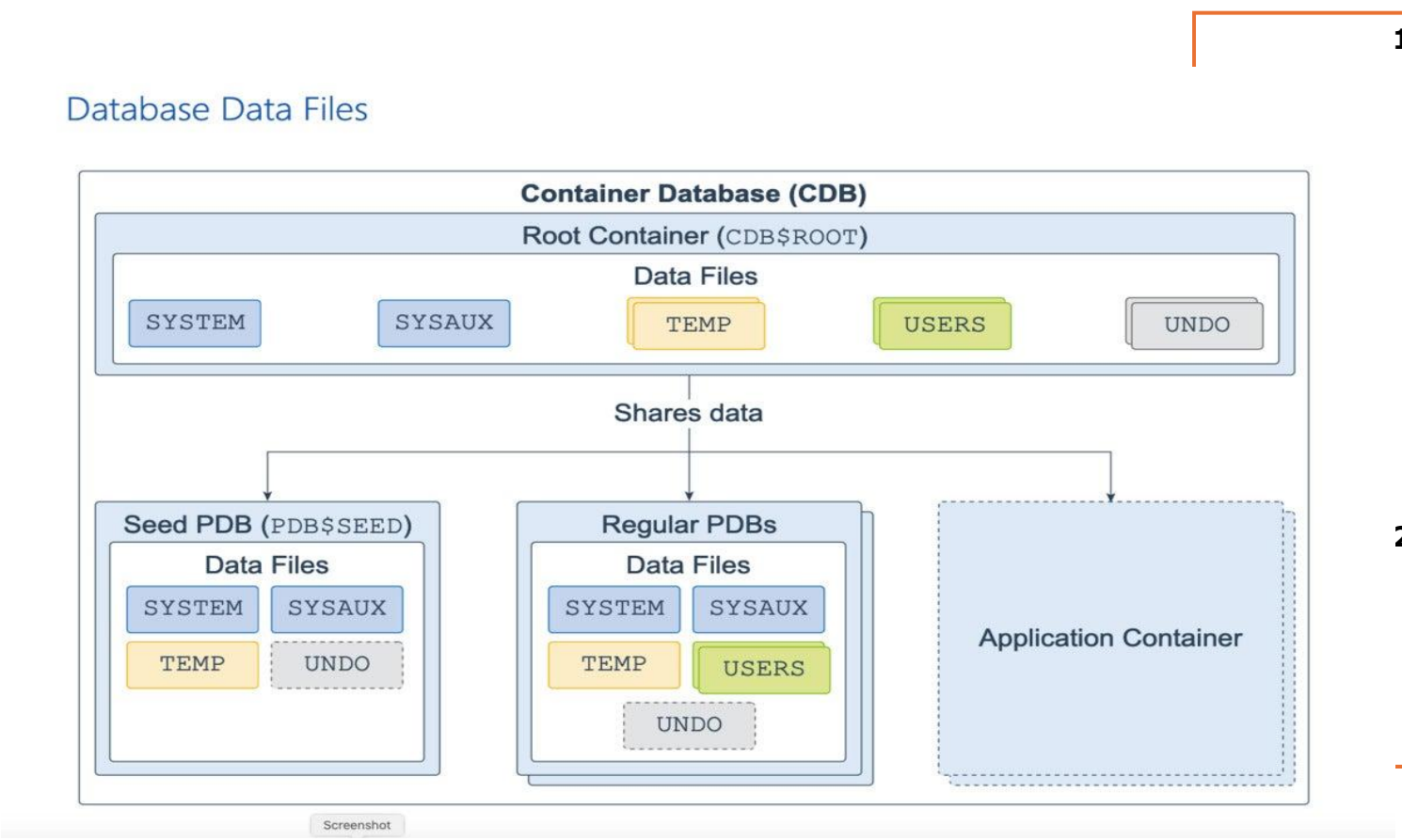
4.Database:
- The storage component that holds the physical data files and system files.

Source Image: https://www.infoworld.com/article/2264510/oracle-database-21c-review-the-old-rdbms-is-new-again.html

Multi-tenant Container Database (CDB) & Pluggable Databases (PDB)

PLSQL PL/SQL Language

An Oracle Database Is A **Multi-tenant Container Database (CDB) With Multiple User-created Pluggable Databases (PDB).**



1. Container Database (CDB):

1. The **CDB** is the main database that houses multiple pluggable databases (PDBs).
2. It has a **Root Container (CDB$ROOT)**, which contains metadata and shared data used across all PDBs.
3. The root container includes the following data files:
   - **SYSTEM**: Contains core database metadata, objects, and code required by Oracle Database.
   - **SYSAUX**: Holds auxiliary metadata, such as performance and administrative data.
   - **TEMP**: Temporary data storage used for sorting and other transient tasks.
   - **USERS**: A tablespace where user data can be stored.
   - **UNDO**: Handles rollback operations (shown here as a shared resource).

2. Seed PDB (PDB$SEED):

1. The **Seed PDB** is a template for creating new PDBs. It has a set of predefined data files:
   - **SYSTEM** and **SYSAUX** (as in the root container).
   - **TEMP** for temporary data.
   - **UNDO** may not always be unique to the Seed PDB, depending on configuration.

Source Image: https://www.infoworld.com/article/2264510/oracle-database-21c-review-the-old-rdbms-is-new-again.html

Multi-tenant Container Database (CDB) & Pluggable Databases (PDB)

PLSQL PL/SQL Language

3. Regular Pluggable Databases (Regular PDBs):

1. Each **Regular PDB** is a self-contained database created from the Seed PDB.
2. Regular PDBs include their own data files for:
   - **SYSTEM** and **SYSAUX** for metadata specific to the PDB.
   - **TEMP** and **USERS** for temporary and user data storage.
   - **UNDO**, if not shared, for rollback operations.

4. Application Container:

- This is an optional container for managing shared data or metadata across multiple application PDBs.
- It can host application-specific metadata and configuration while maintaining separate PDBs for different applications.

Key Insights:

- The **CDB$ROOT** shares its resources with the PDBs to optimize management and performance.
- The **Seed PDB** serves as a standardized starting point for creating PDBs.
- Regular PDBs are isolated from each other in terms of their data but share the root container's infrastructure.
- This architecture supports scalability, allowing multiple PDBs to run within a single CDB, minimizing resource overhead.

What is SQL*Plus?

SQL*Plus is an Oracle command-line tool used to interact with Oracle databases. It acts as a **query interface** for running SQL commands, PL/SQL blocks, and managing database objects. SQL*Plus is widely used by database administrators (DBAs) and developers for various database operations.

Key Features of SQL*Plus

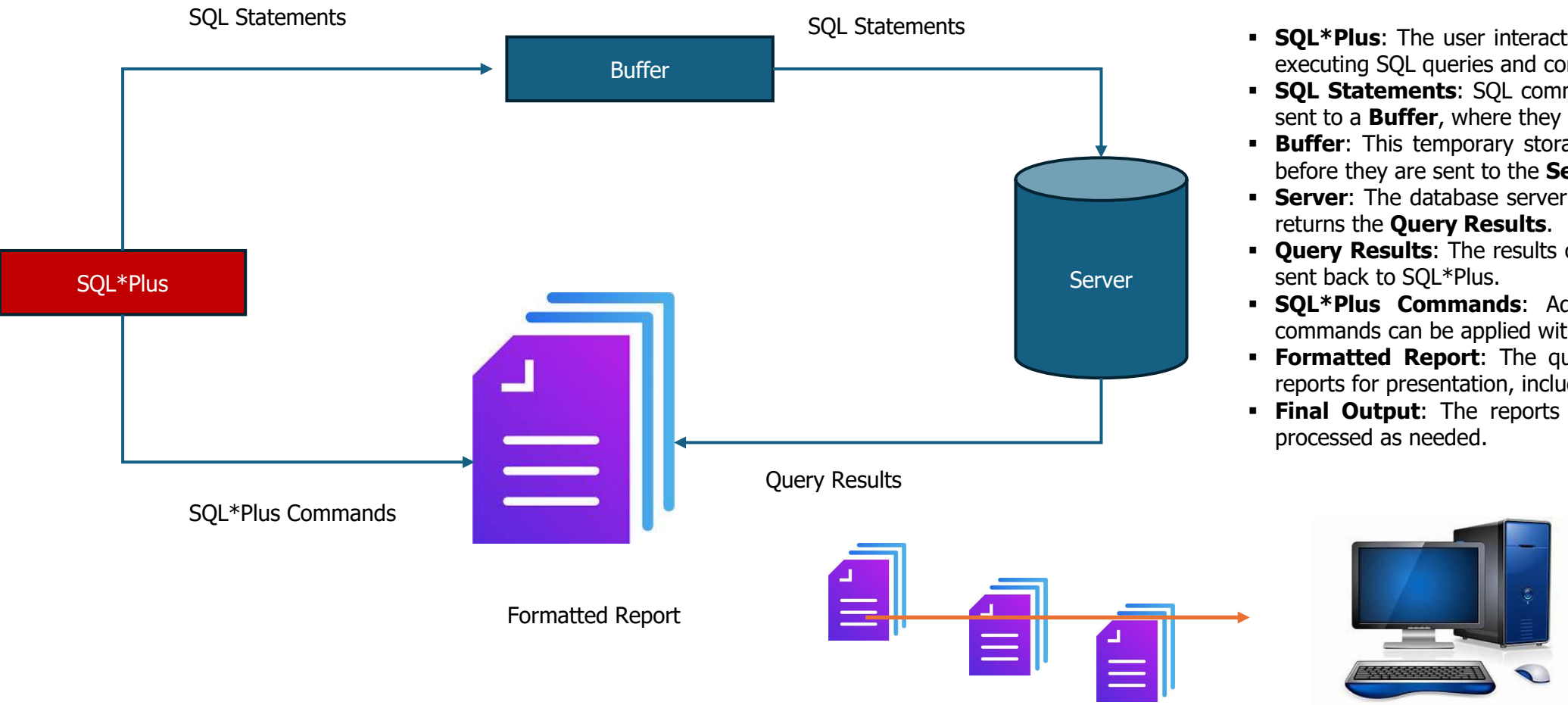
- **Interactive Query Execution**: Run SQL queries and view results directly in the command-line interface.
- **Support for PL/SQL Blocks**: Execute PL/SQL scripts for procedural programming.
- **Database Administration**: Perform tasks like creating users, managing roles, and monitoring database health.
- **Scripting and Automation**: Use SQL*Plus scripts (.sql files) to automate routine tasks.
- **Customization**: Customize the environment with SQL*Plus commands like SET and DEFINE to format outputs.

Key SQL*Plus Commands

- **CONNECT**: Connect to an Oracle database.
  Example: CONNECT username/password@database
- **SELECT**: Retrieve data from tables.
  Example: SELECT * FROM employees;
- **DESC**: Describe the structure of a table or view.
  Example: DESC employees;
- **SET**: Customize session settings (e.g., format, linesize).
  Example: SET LINESIZE 100
- **SPOOL**: Save query results to a file.
  Example: SPOOL output.txt

SQLPlus *is an essential tool for database interaction, particularly for running SQL queries and managing Oracle databases. While it provides an interface to execute PL/SQL, the two serve distinct roles:* ***SQL*****Plus is a client tool**, whereas **PL/SQL is a language for database programming.**

SQL and SQL*Plus Interaction



Process Flow:

- **SQL*Plus**: The user interacts with SQL*Plus, an Oracle tool for executing SQL queries and commands.
- **SQL Statements**: SQL commands entered by the user are first sent to a **Buffer**, where they are temporarily stored.
- **Buffer**: This temporary storage area holds the SQL statements before they are sent to the **Server** for processing.
- **Server**: The database server processes the SQL statements and returns the **Query Results**.
- **Query Results**: The results of the executed SQL statements are sent back to SQL*Plus.
- **SQL*Plus Commands**: Additional formatting or processing commands can be applied within SQL*Plus.
- **Formatted Report**: The query results can be formatted into reports for presentation, including structured outputs.
- **Final Output**: The reports can be printed, saved, or further processed as needed.

SQL Statements vs SQL*Plus Commands

PLSQL
PL/SQL Language

Aspect	SQL Statements	SQL*Plus Commands
Definition	Instructions written in **SQL** to interact directly with the database.	Commands specific to **SQL*Plus** used to manage the environment and improve user interaction.
Purpose	Perform database operations such as querying, updating, or managing data.	Control the behavior, formatting, and output of the SQL*Plus session.
Execution	Can be executed in **any SQL environment** (e.g., SQL*Plus, SQL Developer, JDBC, etc.).	Only recognized and executed within **SQL*Plus**.
Examples	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP	SET, SPOOL, DESCRIBE, PROMPT, SHOW
Output	Produces results from database operations (e.g., rows returned from a query).	Changes how output is displayed or saved (e.g., line width, page size, saving results to a file).
Scope	Affects the **database itself** (data manipulation, schema definition, control).	Affects only the **SQL*Plus session environment**, not the database.

Difference between SQL Statements and SQL*Plus Commands.

PLSQL PL/SQL Language

Category	Command / Statement	Description
SQL Statement	SELECT * FROM employees WHERE department_id = 10;	Retrieves all columns from the **employees** table where department_id = 10.
SQL Statement	UPDATE employees SET salary = salary * 1.1 WHERE employee_id = 101;	Increases salary by **10%** for the employee with employee_id = 101.
SQL Statement	CREATE TABLE departments (department_id NUMBER PRIMARY KEY, department_name VARCHAR2(50));	Creates a new table **departments** with department_id as the primary key.
SQL*Plus Command	SET LINESIZE 100;	Adjusts the line width of the SQL*Plus output to **100 characters** per line.
SQL*Plus Command	SET PAGESIZE 50;	Limits the output to **50 rows per page** in SQL*Plus.
SQL*Plus Command	SPOOL results.txt;	Starts saving query results to a file named **results.txt**.
SQL*Plus Command	SELECT * FROM employees;	Executes a query to fetch all rows from the **employees** table.
SQL*Plus Command	SPOOL OFF;	Stops saving results to the file (results.txt).
SQL*Plus Command	DESCRIBE employees;	Displays the structure of the **employees** table (columns + data types).
SQL*Plus Command	SHOW USER;	Displays the **currently logged-in user** in SQL*Plus.

Key Points

- **SQL Statements** directly interact with the database (e.g., querying, inserting, updating data).
- **SQL*Plus Commands** help manage the SQL*Plus environment (e.g., controlling output format, saving results to files, and showing session details).

SQL*Plus - Usage

PLSQL
PL/SQL Language

Windows PowerShell | Windows PowerShell | Windows PowerShell

```
PS C:\Users\ericm> sqlplus sys/admin@localhost:1521/FREE as sysdba

SQL*Plus: Release 23.0.0.0.0 - Production on Sun Sep 28 13:00:18 2025
Version 23.9.0.25.07

Copyright (c) 1982, 2025, Oracle.  All rights reserved.


Connected to:
Oracle Database 23ai Free Release 23.0.0.0.0 - Develop, Learn, and Run for Free
Version 23.9.0.25.07

SQL> SHOW PDBS;

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 FREEPDB1                       READ WRITE NO
         6 ECOMMERCE_PDB                  READ WRITE NO
SQL> SHOW CON_NAME;

CON_NAME
------------------------------
CDB$ROOT
SQL>
```

Windows PowerShell | Windows PowerShell | Windows PowerShell | Windows PowerShe

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ericm> sqlplus / as sysdba

SQL*Plus: Release 23.0.0.0.0 - Production on Sun Sep 28 13:00:26 2025
Version 23.9.0.25.07

Copyright (c) 1982, 2025, Oracle.  All rights reserved.


Connected to:
Oracle Database 23ai Free Release 23.0.0.0.0 - Develop, Learn, and Run for Free
Version 23.9.0.25.07

SQL> SHOW PDBS;

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 FREEPDB1                       READ WRITE NO
         6 ECOMMERCE_PDB                  READ WRITE NO
SQL> SHOW CON_NAME;

CON_NAME
------------------------------
CDB$ROOT
SQL>
```

Why sqlplus / as sysdba Works Directly

SQL*Plus - Usage

PLSQL
PL/SQL Language

```
SQL>
SQL> show con_name;

CON_NAME
------------------------------
CDB$ROOT
SQL>
SQL> SELECT name FROM v$pdbs;

NAME
--------------------------------------------------------------------------------
PDB$SEED
ORACLE21DBCPDB
PLSQL2024

SQL>
SQL> SET SERVEROUTPUT ON;
SQL>
SQL>  BEGIN
  2   DBMS_OUTPUT.PUT_LINE('Hello, World!');
  3   END;
  4  /
Hello, World!

PL/SQL procedure successfully completed.
```

Oracle SQL (Structured Query Language) Features 1/2

1. Data Querying and Manipulation

- **SELECT Statements**: Retrieve data from one or more tables.
- **Joins**: Combine data from multiple tables using INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN.
- **Subqueries**: Use nested queries within the main query.
- **Aggregating Data**: Functions like COUNT(), SUM(), AVG(), MAX(), and MIN() for summarizing data.
- **Filtering and Sorting**: Use WHERE, ORDER BY, HAVING, and GROUP BY to filter and sort data.

2. Data Definition

- **CREATE**: Define new tables, views, indexes, etc.
- **ALTER**: Modify the structure of an existing table, column, or index.
- **DROP**: Remove objects like tables, views, or indexes from the database.
- **RENAME**: Change the name of a database object.

3. Data Integrity and Constraints

- **Primary Key**: Ensure that each record in a table is unique and identifiable.
- **Foreign Key**: Enforce referential integrity between tables.
- **Unique**: Ensure all values in a column are unique.
- **Check Constraints**: Ensure values in a column satisfy a specified condition.
- **Not Null**: Ensure that a column does not contain null values.

4. Transactional Control

- **COMMIT**: Save changes made to the database.
- **ROLLBACK**: Undo changes made in the current transaction.
- **SAVEPOINT**: Create a point in a transaction to roll back to.
- **SET TRANSACTION**: Control the properties of a transaction, such as isolation level.

5. Views and Indexes

- **Views**: Virtual tables based on the result of a query, simplifying complex queries and providing security by limiting access to certain columns.
- **Indexes**: Improve the performance of queries by providing quick access to rows in a table.

6. Stored Procedures and Functions

- **PL/SQL**: Oracle's extension of SQL for procedural programming, allowing you to define reusable stored procedures, functions, and triggers.
- **Triggers**: Automatic actions that occur in response to certain events, such as before or after an insert, update, or delete operation.

Oracle SQL (Structured Query Language) Features 2/2

PLSQL
PL/SQL Language

7. Advanced Querying Capabilities

- **Hierarchical Queries**: Use CONNECT BY for querying hierarchical data (e.g., organization charts, bill-of-materials).
- **Analytic Functions**: Functions like RANK(), ROW_NUMBER(), LEAD(), and LAG() for advanced reporting and windowed analytics.
- **Full-Text Search**: Oracle's Full-Text Search allows efficient querying of text-based data.

8. Security Features

- **User Management**: Create and manage user roles and privileges.
- **Granting and Revoking Permissions**: Control access to database objects using the GRANT and REVOKE commands.
- **Audit**: Track and log database activity for security auditing purposes.

9. Performance Tuning

- **EXPLAIN PLAN**: Analyze query execution plans to optimize performance.
- **Indexes and Optimizers**: Help speed up query processing by reducing the amount of data the database engine needs to scan.
- **Partitioning**: Improve query performance by dividing large tables into smaller, more manageable parts.

10. Concurrency Control

- **Locking**: Mechanisms to ensure data consistency and prevent conflicts in multi-user environments.
- **Isolation Levels**: Control the visibility of data changes made by other transactions.

11. Integration with Other Languages

- **Java**: Oracle supports embedding Java within SQL queries and using Java Stored Procedures.
- **XML Support**: Oracle SQL has robust features for working with XML data, such as XMLTYPE and XQuery.

12. Error Handling

- **Exception Handling in PL/SQL**: Catch and handle exceptions (errors) in PL/SQL blocks to prevent program crashes and improve robustness.

13. Oracle Extensions to SQL

- **Dual Table**: A special table (DUAL) that returns a single row with a single column, typically used to select a value or perform a calculation.
- **Sequences**: Automatically generate unique numbers for primary keys and other purposes.
- **Synonyms**: Create alternative names for database objects for easier referencing.

Oracle PL/SQL (Procedural Language/SQL)

Feature	Description
Block Structure	PL/SQL programs are organized into blocks (anonymous or named).
Procedures and Functions	Create reusable subprograms for modular programming and improved maintainability.
Control Structures	Use loops (FOR, WHILE), conditional statements (IF-THEN-ELSE), and case expressions.
Exception Handling	Manage runtime errors using EXCEPTION blocks.
Cursors	Work with explicit and implicit cursors to process query results row-by-row.
Triggers	Automatically execute code in response to specific events (e.g., BEFORE or AFTER DML operations).
Packages	Group related procedures, functions, variables, and cursors into modules.
Data Types	Support scalar, composite, reference, and LOB data types for flexible programming.
Collections	Use associative arrays, nested tables, and VARRAYs for advanced data handling.
Dynamic SQL	Execute SQL dynamically at runtime using EXECUTE IMMEDIATE and DBMS_SQL.
PL/SQL Libraries	Use built-in packages like DBMS_OUTPUT, UTL_FILE, and DBMS_SCHEDULER.
Integration with SQL	Seamlessly embed SQL statements within PL/SQL code for database interaction.
Performance Features	Leverage bulk processing (FORALL, BULK COLLECT) for efficient data operations.
Portability	Write code that runs across multiple Oracle database platforms.
Object-Oriented Programming	Support for user-defined types, methods, and object-oriented programming features.

Usage: PL/SQL is used for:

- **Batch Processing**: Running multiple SQL statements in sequence, using variables, cursors, and control statements.
- **Improved Performance**: Avoids multiple round-trips between the application and the database by bundling SQL statements into a single block of code.

PL/SQL vs. Its Counterparts in Different Database Platforms

PL/SQL (Procedural Language/Structured Query Language) is specifically designed for Oracle databases. It is Oracle Corporation's procedural extension to SQL and is tightly integrated with Oracle's SQL engine.

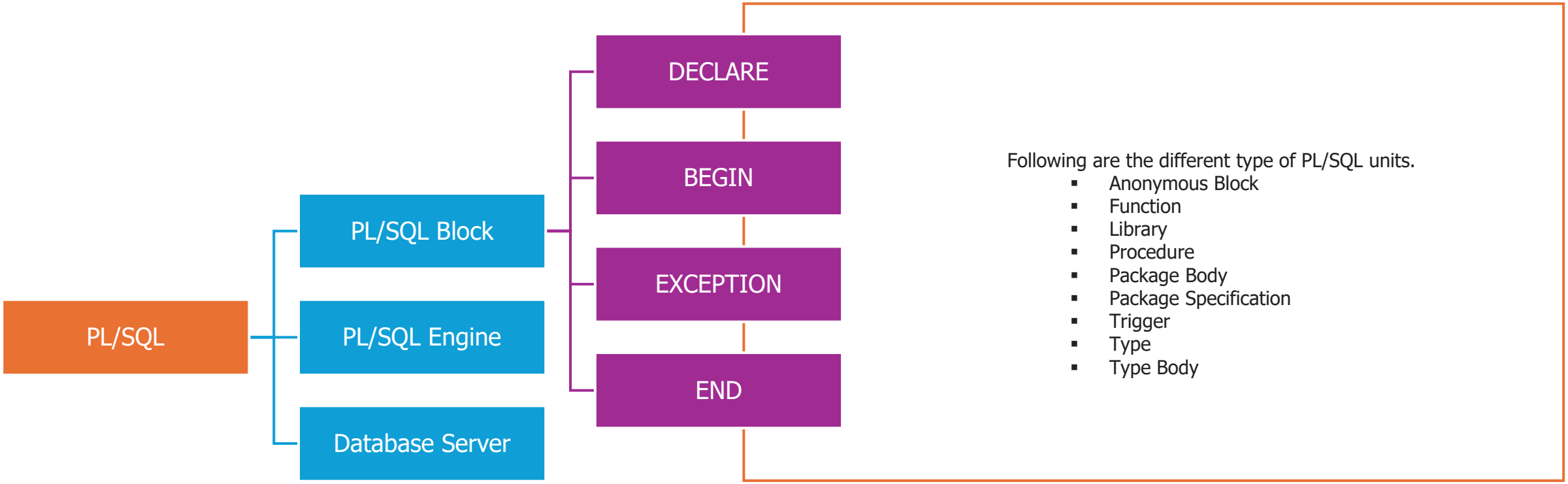
Database System	Procedural Language	Description
Oracle	PL/SQL	Oracle's proprietary procedural extension to SQL.
PostgreSQL (pgAdmin)	PL/pgSQL	Procedural language designed for PostgreSQL.
Microsoft SQL Server	T-SQL (Transact-SQL)	Procedural extension used in Microsoft SQL Server.
MySQL	MySQL SQL/PSM	MySQL's procedural extension for SQL.

In summary, **PL/SQL is exclusive to Oracle databases**, and each database management system (DBMS) typically has its own procedural language.

- **PL/SQL (Oracle)** → Rich, enterprise-grade, very mature for large-scale business logic.
- **PL/pgSQL (PostgreSQL)** → Flexible, very similar to PL/SQL, with open-source extensibility.
- **T-SQL (SQL Server)** → Strong in integration, system functions, and transactional programming.
- **SQL/PSM (MySQL)** → Lighter, standard-based, suitable for procedural routines but less feature-rich.

The Architecture of PL/SQL (1/2)

PL/SQL (Procedural Language/Structured Query Language) is Oracle Corporation's procedural extension to SQL, designed specifically for **managing and manipulating data in Oracle databases**. It combines the data manipulation capabilities of SQL with the procedural programming features of languages like C or Java, allowing developers to write complex scripts that can execute multiple **SQL statements, control logic, handle exceptions,** and more **within a single program unit**.

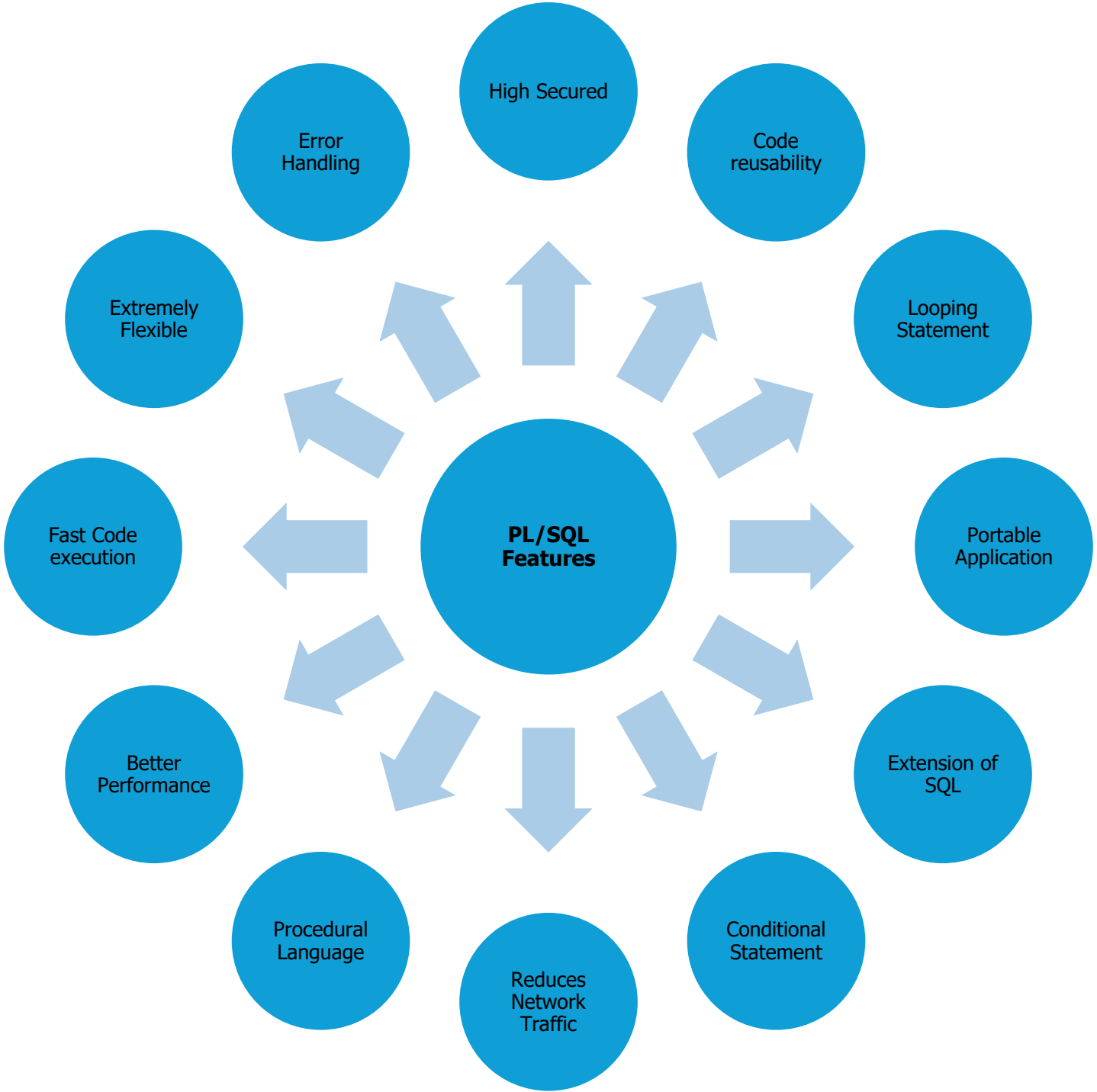


Following are the different type of PL/SQL units.

- Anonymous Block
- Function
- Library
- Procedure
- Package Body
- Package Specification
- Trigger
- Type
- Type Body

PLSQL
PL/SQL Language

Why Use PL/SQL? Key Features & Benefits



PL/SQL Key Features and Benefits - Examples

PLSQL
PL/SQL Language

Feature	Description	Example
High Performance	PL/SQL is designed for high-performance transaction processing, optimizing database operations.	A PL/SQL block that retrieves and processes thousands of records faster than a standard SQL query.
Code Reusability	Supports code reusability, allowing you to write modular code that can be reused across different programs.	Creating a procedure to calculate tax, which can be called whenever needed, instead of rewriting the tax calculation logic.
Error Handling	Provides robust error handling by returning user-friendly error messages when errors occur.	Using EXCEPTION to catch a NO_DATA_FOUND error and return a custom message.
Procedural Language Capability	Includes procedural language constructs like conditional statements (IF...ELSE) and loops (FOR loop).	Using an IF...ELSE statement to check a condition and a FOR loop to iterate over a range of values.
Block Structure	Consists of nested blocks of code, where each block is a unit of task or logical module.	A block containing variable declarations, SQL statements, and an exception-handling section, all within a single logical unit.
Better Performance	Oracle engine processes multiple SQL statements in a single block, reducing network traffic.	A PL/SQL block that executes multiple SQL operations (e.g., INSERT, UPDATE, DELETE) in a single transaction.
Variable Declaration	Allows declaring variables that can be used and manipulated within the block.	Declaring a variable to store an employee's salary and then updating it within the same block.
Looping and Conditional Statements	Supports loops (FOR, WHILE) and conditional statements (IF...ELSE) for complex control flows.	A FOR loop to iterate over a list of employee IDs and apply a bonus if the salary exceeds a certain amount.
Reduce Network Traffic	Multiple SQL statements are processed as a single block, minimizing network interactions with the database.	Combining several INSERT operations in one PL/SQL block instead of executing them individually.
Portable Applications	PL/SQL applications are portable across different operating systems without modification.	A PL/SQL procedure created on an Oracle database in Windows will work on Linux without changes.

Anonymous PL/SQL Block

```
-- Declare the PL/SQL block
DECLARE
  -- Declare a variable `v_salary` of type NUMBER with a precision of 8 digits and 2 decimal places
  v_salary NUMBER(8,2);

-- Start the executable section of the PL/SQL block
BEGIN
  -- Execute a SELECT statement to retrieve the salary of the employee with employee_id = 1001
  -- Store the retrieved salary into the variable `v_salary`
  SELECT salary INTO v_salary FROM employees WHERE employee_id = 1001;

  -- Check if the value of `v_salary` is greater than 5000
  IF v_salary > 5000 THEN
    -- If the condition is true (i.e., `v_salary` is greater than 5000), output 'High salary'
    DBMS_OUTPUT.PUT_LINE('High salary');
  ELSE
    -- If the condition is false (i.e., `v_salary` is 5000 or less), output 'Low salary'
    DBMS_OUTPUT.PUT_LINE('Low salary');
  END IF;

-- End of the executable section
END;
```

Anonymous PL/SQL Block:

- **Not Named:** It is not stored with a name in the database.
- **Temporary:** Executed once and then discarded.

Key Points

- **Variable Declaration**: v_salary is defined to store the salary of an employee.
- **Data Retrieval**: The SELECT INTO statement retrieves data from the database and stores it in the variable.
- **Conditional Logic**: The IF-THEN-ELSE construct is used to perform different actions based on the value of v_salary.
- **Output**: DBMS_OUTPUT.PUT_LINE is used to display messages based on the condition.

Features Higher Performance & Code Reusability

PLSQL
PL/SQL Language

High Performance

```
BEGIN
   -- Loop through each record from the 'employees' table where department_id is 10
   FOR rec IN (SELECT * FROM employees WHERE department_id = 10) LOOP
      -- Output the first name of each employee in the result set
      DBMS_OUTPUT.PUT_LINE('Employee: ' || rec.first_name);
   END LOOP;
END;
```

- **FOR employee_record IN (SELECT first_name FROM employees WHERE department_id = 10):**
  This loop iterates through the result set of the query, which retrieves the first_name of employees in department 10.

- **DBMS_OUTPUT.PUT_LINE(employee_record.first_name);:**
  This line prints the first_name of each employee to the output.

Make sure that DBMS_OUTPUT is enabled in your environment to see the output. You can enable it in SQL*Plus or SQL Developer by executing:

- **SET SERVEROUTPUT ON;**

Code Reusability

```
CREATE OR REPLACE PROCEDURE calculate_tax (p_salary IN NUMBER) IS
   -- Declare a variable to store calculated tax
   v_tax NUMBER;
BEGIN
   -- Calculate 10% tax on the input salary
   v_tax := p_salary * 0.10;

   -- Output the calculated tax value
   DBMS_OUTPUT.PUT_LINE('Tax: ' || v_tax);
END;
```

- **Procedure Name**: calculate_tax
- **Input Parameter**: p_salary (of type NUMBER)
- **Variable**: v_tax is declared to store the calculated tax.
- **Main Logic**: It calculates 10% of the salary (p_salary * 0.10) and assigns it to v_tax.
- **Output**: The calculated tax is printed using DBMS_OUTPUT.PUT_LINE.

Error Handling & Procedural Language Capability

Error Handling

```
BEGIN
   -- Retrieve salary of employee with employee_id 999 and store in v_salary
   SELECT salary INTO v_salary FROM employees WHERE employee_id = 999;

EXCEPTION
   -- Handle case when no record is found for the given employee_id
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No employee found with the given ID.');
END;
```

- **Main Logic**: The block tries to retrieve the salary of an employee with employee_id = 999 and store it in v_salary.
- **Exception Handling**: If no employee is found with that ID, the NO_DATA_FOUND exception is caught, and a message is printed using DBMS_OUTPUT.PUT_LINE.

Procedural language Capability

```
BEGIN
   -- Check if the salary is greater than 5000
   IF v_salary > 5000 THEN
      -- If salary is greater than 5000, print "High salary"
      DBMS_OUTPUT.PUT_LINE('High salary');
   ELSE
      -- If salary is 5000 or less, print "Low salary"
      DBMS_OUTPUT.PUT_LINE('Low salary');
   END IF;
END;
```

- **IF v_salary > 5000 THEN**:
- This line checks if the variable v_salary (employee salary) is greater than 5000.
- **DBMS_OUTPUT.PUT_LINE('High salary');**:
- If the condition is true (salary > 5000), it prints "High salary" to the output.
- **ELSE**:
- If the salary is less than or equal to 5000, the code executes the block in the ELSE statement.
- **DBMS_OUTPUT.PUT_LINE('Low salary');**:
- This line prints "Low salary" if the salary is 5000 or less.
- **END IF;**:
- Ends the conditional block.

Block Structure & Better Performance

PLSQL
PL/SQL Language

Block Structure

```
DECLARE
   -- Declare a variable to store the salary
   v_salary NUMBER;
BEGIN
   -- Select the salary of the employee with ID 100 into the v_salary variable
   SELECT salary INTO v_salary FROM employees WHERE employee_id = 100;

   -- Check if the salary is greater than 5000 and print 'High salary' if true
   IF v_salary > 5000 THEN
      DBMS_OUTPUT.PUT_LINE('High salary');
   END IF;

EXCEPTION
   -- Handle the case where no employee with the specified ID is found
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No employee found');
END;
```

- **Main Logic**: The block tries to retrieve the salary of an employee with employee_id = 999 and store it in v_salary.
- **Exception Handling**: If no employee is found with that ID, the NO_DATA_FOUND exception is caught, and a message is printed using DBMS_OUTPUT.PUT_LINE.

- **Update**: Increases salaries by 10% for all employees in department 10.
- **Delete**: Removes the employee with employee_id = 999.
- **Commit**: Saves both changes in a single transaction to the database.

This block efficiently handles both operations together and ensures data consistency.

Better Performance

```
BEGIN
   -- Update salaries in department 10 by 10%
   UPDATE employees SET salary = salary * 1.10 WHERE department_id = 10;

   -- Delete employee with employee_id = 999
   DELETE FROM employees WHERE employee_id = 999;

   -- Commit both operations in one transaction
   COMMIT;
END;
```

Variable & Looping and Condition Statement

PLSQL
PL/SQL Language

SQL*Plus

Variable Declaration

```
DECLARE
  -- Declare and initialize bonus variable
  v_bonus NUMBER := 1000;
  -- Declare salary variable
  v_salary NUMBER;
BEGIN
  -- Retrieve the salary of the employee with ID 100
  SELECT salary INTO v_salary FROM employees WHERE employee_id = 100;

  -- Add bonus to the retrieved salary
  v_salary := v_salary + v_bonus;

  -- Print the new salary
  DBMS_OUTPUT.PUT_LINE('New salary: ' || v_salary);
```

- **Variable Declaration:**
  - **v_bonus:** Initialized to 1000.
  - **v_salary:** Declared to hold the employee's salary.
- **Retrieve Salary:**
  - Fetches the salary of the employee with employee_id = 100 into v_salary.
- **Calculate New Salary:**
  - Adds the bonus to the retrieved salary.
- **Output:**
  - Displays the updated salary with the bonus added.

Looping and Conditional Statements

```
BEGIN
  -- Loop through numbers 1 to 5
  FOR i IN 1..5 LOOP
    -- Check if the current number is 3
    IF i = 3 THEN
      -- Print a special message for the number 3
      DBMS_OUTPUT.PUT_LINE('Special case for 3');
    ELSE
      -- Print the current number
      DBMS_OUTPUT.PUT_LINE('Number: ' || i);
    END IF;
  END LOOP;
END;
```

- **Loop**:
  - Iterates from 1 to 5, with i taking each value in this range.
- **Condition**:
  - If i is 3, prints "Special case for 3".
  - For other values of i, prints "Number: " followed by the value of i.

Reduce Network Traffic & Portable Application

Reduce Network Traffic

```
BEGIN
  -- Insert a new employee with ID 101 and name 'John'
  INSERT INTO employees (employee_id, first_name) VALUES (101, 'John');

  -- Insert another employee with ID 102 and name 'Jane'
  INSERT INTO employees (employee_id, first_name) VALUES (102, 'Jane');

  -- Commit the transaction to save the changes
  COMMIT;
END;
```

- **Insert Statements:**
  - Adds two employees to the employees table: one with ID 101 and name 'John', and another with ID 102 and name 'Jane'.
- **Commit:**
  - Saves both insertions in a single transaction, making the changes permanent.
- **Batch Inserts in PL/SQL Block:**
  - Both INSERT statements are sent as a single request.
  - A single commit operation finalizes all changes.

By consolidating operations into a single PL/SQL block, network traffic is optimized, leading to better performance and reduced latency.

Portable Application

```
-- This procedure will run on any Oracle Database regardless of OS
CREATE OR REPLACE PROCEDURE hello_world IS
BEGIN
  -- Output a simple greeting message
  DBMS_OUTPUT.PUT_LINE('Hello, World!');
END;
/
```

- **Procedure**:
  - **CREATE OR REPLACE PROCEDURE hello_world IS**: Defines a new procedure named hello_world.
  - **BEGIN**: Starts the executable part of the procedure.
  - **DBMS_OUTPUT.PUT_LINE('Hello, World!');**: Outputs the string "Hello, World!" the console.
  - **END;**: Ends the procedure.
- **Execution**: This procedure can be executed in any environment where Oracle Database is installed, making it highly portable across different operating systems and versions of Oracle Database.

Procedure Implementation & Solution

PLSQL
PL/SQL Language

```
CREATE OR REPLACE PROCEDURE Add_Numbers_With_Increment(
    num1      IN  NUMBER,
    num2      IN  NUMBER,
    increment IN  NUMBER,
    result    OUT NUMBER
) AS
BEGIN
    -- Add the two numbers and the increment
    result := (num1 + num2) + increment;

    -- Output the result
    DBMS_OUTPUT.PUT_LINE('The sum of ' || num1 || ' and ' || num2 ||
                         ' with an increment of ' || increment ||
                         ' is: ' || result);
END;
/
```

- num1 and num2 are the numbers to be added.
- increment is the value to be added to the sum of the two numbers.
- result is an output parameter that will hold the final result.
- DBMS_OUTPUT.PUT_LINE is used to print the result in the console.

Call the procedure

```
DECLARE
    sum_result NUMBER;
BEGIN
    -- Call the procedure
    Add_Numbers_With_Increment(5, 10, 2, sum_result);

    -- Display the final result
    DBMS_OUTPUT.PUT_LINE('Final Result: ' || sum_result);
END;
/
```

- The DECLARE block defines sum_result to store the output.
- The procedure Add_Numbers_With_Increment(5, 10, 2, sum_result) is called with:
  - 5 and 10 as the two numbers to add.
  - 2 as the increment.
  - sum_result to hold the result (17).
- The procedure computes (5 + 10) + 2 = 17 and stores it in sum_result.
- DBMS_OUTPUT.PUT_LINE prints the final result: "Final Result: 17".

- **The sum of 5 and 10 with an increment of 2 is: 17**
- **Final Result: 17**

Oracle SQL vs PL/SQL

Feature	Oracle SQL	Oracle PL/SQL
Type	Declarative (Non-Procedural)	Procedural Language (Extension of SQL)
Purpose	Data querying and manipulation	Procedural logic and complex operations
Execution	Executes single statements	Executes blocks of code (multiple SQL statements and logic)
Control Structures	No control structures (loops, conditions)	Supports loops, conditions, exceptions
Reusability	No reusable logic	Supports reusable logic (procedures, functions)
Transaction Handling	Manages transactions (COMMIT, ROLLBACK)	Handles transactions within a block
Error Handling	Limited error handling	Advanced error handling using EXCEPTION blocks
Integration	Focuses solely on data manipulation	Integrates SQL within procedural code

Advantages of Oracle Database over other RDBMS systems

PLSQL
PL/SQL Language

Scalability and Performance
- Horizontal and vertical scalability.
- Real Application Clusters (RAC) for load balancing and high availability.
- Advanced query optimization.

High Availability and Disaster Recovery
- Oracle Data Guard for failover and data protection.
- Automatic Storage Management (ASM) for storage optimization.
- Flashback Technology for quick data recovery.

Security
- Advanced encryption (in transit and at rest).
- Virtual Private Database (VPD) for fine-grained access control.
- Data masking and redaction.

Support for Advanced Data Types
- Support for XML, JSON, spatial data, and multimedia.
- Object-relational model for complex data structures.

Cross-Platform Flexibility
- Runs on multiple platforms (Windows, Linux, Unix, Solaris).
- Cloud compatibility (Oracle Cloud, AWS, Azure).

Enterprise-Level Features
- Partitioning for improved query performance.
- Oracle Exadata for optimized performance.
- Multitenant architecture for managing multiple databases.

Comprehensive Tooling and Ecosystem
- Oracle Enterprise Manager (OEM) for centralized monitoring and management.
- Data integration and business intelligence tools.

Support and Community
- 24/7 enterprise-level support.
- Active user community and extensive documentation.

Automation
- Autonomous database for automatic tuning, backups, and patching.

Compliance and Auditing
- Extensive auditing tools for regulatory compliance (GDPR, HIPAA, etc.).

PL/SQL Disadvantage

PLSQL
PL/SQL Language

Licensing Costs

- Oracle's PL/SQL tools are expensive, making it costly for smaller organizations.

High Memory Usage

- Stored procedures in PL/SQL can use a lot of memory, which may slow down the system.

Limited Web Support

- PL/SQL is mainly for database tasks and has limited capabilities for building modern web applications compared to languages like PHP or Python.

Maintenance Issues

- Changes in the database often require updates in the code, leading to more maintenance work.

Difficult Debugging

- PL/SQL lacks advanced debugging tools, making it harder to find and fix errors.

Steep Learning Curve

- PL/SQL is difficult for beginners to learn quickly due to its complex syntax and structure.

Database DependencePL

- SQL works only with Oracle databases, so it can’t be used with other databases or platforms easily.

Summary of Oracle Environment Terms

PLSQL
PL/SQL Language

1. Oracle Database Instance

Definition: The combination of memory structures and background processes that manage database operations.
- **Memory Structures**:
  - **SGA (System Global Area)**: Shared memory region for all users.
  - **PGA (Program Global Area)**: Memory region specific to a user process.
- **Background Processes**: Essential processes that handle tasks such as writing to files and managing memory.

2. Physical Structures

Definition: Files that physically store the database's data and metadata.
- **Data Files**: Store actual database data.
- **Control Files**: Maintain information about the structure of the database.
- **Redo Log Files**: Record all changes made to the database for recovery purposes.

3. Logical Structures

Definition: Components that define how data is logically organized within the database.
- **Tablespaces**: High-level logical containers for data storage.
- **Segments**: Space allocated for a specific type of data, such as a table or index.
- **Extents**: Contiguous blocks of space within a segment.
- **Blocks**: Smallest unit of storage in the database.

4. Schema Objects

Definition: Logical database objects owned by a user.
- **Tables**: Store data in rows and columns.
- **Indexes**: Improve query performance by allowing faster data access.
- **Views**: Virtual tables created from queries.
- **Sequences**: Generate unique numeric values, often for primary keys.
- **Synonyms**: Aliases for tables, views, or other objects to simplify access.

5. User and Security Management

Definition: Security model based on user roles and privileges.
- **Users and Schemas**: Users own schemas containing database objects.
- **Roles**: Collections of privileges assigned to users.
- **System Privileges**: Permissions for administrative tasks (e.g., creating tables).
- **Object Privileges**: Permissions on specific objects (e.g., SELECT, INSERT, UPDATE).

6. Oracle Network Environment

Definition: Tools enabling communication between clients and the database.
- **Listeners**: Processes that handle client connection requests.
- **TNS (Transparent Network Substrate)**: Manages connection addresses between clients and servers.
- **Database Links**: Facilitate operations between multiple databases.

7. Oracle Multitenant Architecture (Optional)

Definition: Architecture introduced in Oracle 12c for managing multiple databases within a single container.
- **Container Database (CDB)**: Houses multiple Pluggable Databases (PDBs).
- **Benefits**: Improved consolidation, simplified upgrades, and efficient resource management.

8. Oracle Data Guard (High Availability)

Definition: High availability and disaster recovery solution.
- **Standby Databases**: Copies of the primary database for failover scenarios.
- **Switch Over**: Ability to manually or automatically move operations to a standby database.

9. Oracle Backup and Recovery

Definition: Tools and technologies for safeguarding data.
- **RMAN (Recovery Manager)**: Main tool for database backup, restore, and recovery.
- **Flashback Technology**: Allows rewinding the database to a specific point in time.

10. Oracle RAC (Real Application Clusters)

Definition: Clustering solution for high availability and scalability.
- **Features**:
  - Multiple instances accessing the same database.
  - Load balancing and failover for continuous availability.

11. Oracle Enterprise Manager (OEM)

Definition: Oracle's web-based platform for database and infrastructure management.
- **Capabilities**:
  - Performance monitoring.
  - Database administration.
  - Backup configuration and management.

Conclusion: Key Takeaways PL/SQL

PLSQL
PL/SQL Language

1. SQL vs. PL/SQL

- **SQL**: Used for single queries to retrieve/manipulate data (SELECT, INSERT, etc.).
- **PL/SQL**: Extends SQL with procedural logic (loops, conditionals, error handling) for tasks like stored procedures and triggers.

2. Why Choose PL/SQL?

- **Batch processing**: Execute multiple SQL commands in a single block.
- **Error handling**: Manage exceptions effectively.
- **Encapsulation**: Store business logic in procedures/functions for reuse.

3. When Not to Use PL/SQL

- For **simple queries**: Use plain SQL for one-off tasks.
- For **heavy computations**: Offload to application logic.
- For **portability**: PL/SQL is Oracle-specific.

4. Advantages of PL/SQL

- **Improved performance**: Reduces database calls by bundling logic.
- **Security**: Protects data with stored procedures.
- **Reusability**: Encapsulates complex logic.

5. Disadvantages of PL/SQL

SQL*Plus

- **Oracle-specific**: Limits portability to other databases.
- **Complexity**: Harder to debug and maintain procedural logic.

6. Practical Scenarios

- **When PL/SQL improves performance:**
  Example: Processing payroll for 10,000 employees with custom logic in one block.
- **When SQL is preferable:**
  Simple queries, ad-hoc analysis, or cross-database portability.

Practice Exercises

1. Write an SQL query to find the top 10 highest-paid employees.
2. Create a PL/SQL block to calculate and store average salaries by department.
3. Handle duplicate errors in a PL/SQL procedure.
4. Compare performance: Batch update in SQL vs. PL/SQL.

PLSQL
PL/SQL Language

Challenge yourself; it's the only path which leads to growth.
~Morgan Freeman

Stay Connected!