What is File Storage?

**File storage** refers to a method of storing data in files on a computer or server. Each file contains data and is organized in folders/directories. This is the traditional way of storing data, especially in early computing systems.

* Data is stored in formats like .txt, .csv, .xml, etc.
* Access is typically sequential or via file system APIs.
* Common in simple applications or when dealing with unstructured data.

A **Database Management System (DBMS)** offers a more structured and powerful way to store, manage, and retrieve data compared to basic file storage.

Advantage of using DBMS over file storage

**1. Data Integrity and Accuracy**

* DBMS enforces rules (constraints) to ensure data is valid and consistent.
* File systems lack built-in mechanisms to prevent data anomalies.

**2. Data Security**

* DBMS provides user authentication and access control.
* File systems offer limited security features.

**3. Efficient Data Access**

* DBMS uses indexing and query optimization for fast data retrieval.
* File systems often require scanning entire files to find data.

**4. Concurrent Access**

* Multiple users can access and modify data simultaneously in a DBMS.
* File systems struggle with concurrent access and risk data corruption.

**5. Backup and Recovery**

* DBMS supports automated backup and recovery mechanisms.
* File systems require manual processes and are more error-prone.

**6. Data Relationships**

* DBMS can model complex relationships using tables and keys (e.g., relational databases).
* File systems don't inherently support relationships between data.

**7. Scalability**

* DBMS can handle large volumes of data efficiently.
* File systems become inefficient as data grows.

**8. Query Language**

* DBMS uses SQL (Structured Query Language) for powerful data manipulation.
* File systems require custom code or scripts for data operations.

Commands

* **DDL**: Defines structure (CREATE, ALTER, DROP, TRUNCATE).
* **DML**: Manipulates data (SELECT, INSERT, UPDATE, DELETE).
* **DCL**: Controls access (GRANT, REVOKE).

Joins

* LEFT join
* RIGHT join
* FULL Outer Join
* UNION
* INNER Join
* CROSS join

**1. GROUP BY**

* **Purpose**: The GROUP BY clause is used to group rows that have the same values in specified columns into summary rows, typically used with aggregate functions (e.g., COUNT, SUM, AVG, MIN, MAX) to summarize data within each group.
* **How It Works**:
  + GROUP BY divides the result set into groups based on one or more columns.
  + An aggregate function is applied to each group to produce a single value (e.g., total amount per city).
  + Only the grouped columns and aggregated values can appear in the SELECT clause (in most SQL databases).

**2. ORDER BY**

* **Purpose**: The ORDER BY clause sorts the result set based on one or more columns in ascending (ASC) or descending (DESC) order.
* **How It Works**:
  + ORDER BY is applied after all other operations (e.g., WHERE, GROUP BY, JOIN) to sort the final result set.
  + You can sort by any column, whether it’s in the SELECT clause or not (though some databases may require it to be in SELECT).

SUBQUERY

INLINE VIEW

IMP TOPICS

* **VIEW**
* **INDEX**
* **NORMLIZING**
* **KEYS**

<https://grok.com/share/c2hhcmQtNA%3D%3D_d9bf2f5d-696f-4033-bfe2-90ba038ba593>

PL/SQL

PL/SQL (Procedural Language/SQL) is Oracle's proprietary extension of SQL, designed to add procedural programming capabilities to the declarative nature of SQL. It allows developers to write scripts, stored procedures, functions, triggers, and packages to manage and manipulate data in Oracle databases efficiently.

**Key Features of PL/SQL:**

* **Procedural Constructs**: Supports loops, conditionals (IF-THEN-ELSE), and exception handling for robust programming.
* **Block Structure**: Code is organized into blocks containing declarations, executable statements, and exception handling.
*  **Integration with SQL**: Seamlessly embeds SQL queries for data manipulation and retrieval.
*  **Modularity**: Supports stored procedures, functions, and packages for reusable, maintainable code.
*  **Exception Handling**: Built-in mechanisms to handle runtime errors gracefully.
*  **Triggers**: Automates actions (e.g., logging) when specific database events occur.
*  **Performance**: Executes on the database server, reducing network overhead and improving efficiency.

**Use Cases:**

* Automating database tasks (e.g., batch updates).
* Creating complex business logic in stored procedures.
* Building triggers for data validation or auditing.
* Developing reusable database applications.

| **Aspect** | **SQL** | **PL/SQL** |
| --- | --- | --- |
| **Definition** | A declarative query language for defining, manipulating, and querying data in relational databases. | A procedural extension of SQL, exclusive to Oracle, adding programming logic to SQL. |
| **Purpose** | Used for querying, inserting, updating, and deleting data, as well as defining database structures. | Used for writing procedural code (e.g., stored procedures, functions) to implement business logic. |
| **Nature** | Declarative: Specifies *what* to do, not *how* to do it. | Procedural: Supports logic to define *how* tasks are performed (loops, conditionals). |
| **Execution** | Executes single statements or queries directly on the database. | Executes blocks of code (procedures, functions, triggers) on the database server. |
| **Programming Constructs** | Limited to queries (SELECT, INSERT, UPDATE, DELETE) and DDL (CREATE, ALTER). | Supports variables, loops, conditionals (IF-THEN-ELSE), exception handling, and modularity. |
| **Code Structure** | Individual statements or scripts. | Organized in blocks: DECLARE, BEGIN, EXCEPTION, END. |
| **Example** | ```sql:disable-run |  |
| **Use Cases** | - Data retrieval and manipulation. - Schema creation/modification. - Simple reporting. | - Complex business logic. - Stored procedures/functions. - Triggers for automation. - Batch processing. |
| **Error Handling** | Limited; relies on database-level error reporting. | Robust; includes explicit exception handling (e.g., WHEN OTHERS). |
| **Performance** | Executes one statement at a time, which may involve multiple client-server trips. | Runs on the server, reducing network overhead and improving performance for complex tasks. |
| **Scope** | Standard across most relational databases (e.g., MySQL, PostgreSQL). | Oracle-specific; not portable to other database systems. |
| **Modularity** | No support for reusable code units. | Supports reusable units like procedures, functions, and packages. |

**Key Differences Summarized:**

* **Functionality**: SQL is for querying and managing data; PL/SQL adds procedural programming for complex logic.
* **Scope**: SQL is universal; PL/SQL is Oracle-specific.
* **Complexity**: SQL handles straightforward data operations; PL/SQL enables advanced programming with control structures and error handling.
* **Execution**: SQL runs individual statements; PL/SQL runs as blocks on the server, optimizing performance.

**When to Use:**

* **SQL**: For simple queries, data manipulation, or schema changes (e.g., SELECT \* FROM table).
* **PL/SQL**: For automating tasks, implementing business rules, or creating reusable database logic (e.g., stored procedures or triggers).

**1. Procedural Programming**

* Supports procedural constructs like loops (FOR, WHILE), conditionals (IF-THEN-ELSE), and sequential execution.

**2. Block Structure**

* Organizes code into structured blocks: DECLARE, BEGIN, EXCEPTION, END.
* Allows variable declarations, executable statements, and error handling in a single unit.

**3. Seamless SQL Integration**

* Embeds SQL queries (SELECT, INSERT, UPDATE, DELETE) directly within PL/SQL code.

**4. Exception Handling**

* Provides robust error handling with predefined and user-defined exceptions.
* Uses **EXCEPTION** block to catch and handle errors (e.g., NO\_DATA\_FOUND, TOO\_MANY\_ROWS).

**5. Modularity**

* Supports reusable code units:
  + **Stored Procedures**: Perform specific tasks (e.g., updating records).
  + **Functions**: Return a value for use in queries or logic.
  + **Packages**: Group related procedures, functions, and variables for modularity.

**6. Triggers**

* Automates actions based on database events (e.g., INSERT, UPDATE, DELETE).

**8. Cursors**

* Manages query results for row-by-row processing.
* Supports explicit cursors (user-defined) and implicit cursors (handled by PL/SQL).

https://grok.com/share/c2hhcmQtNA%3D%3D\_f007a18b-768f-413a-93b6-25f51101b344