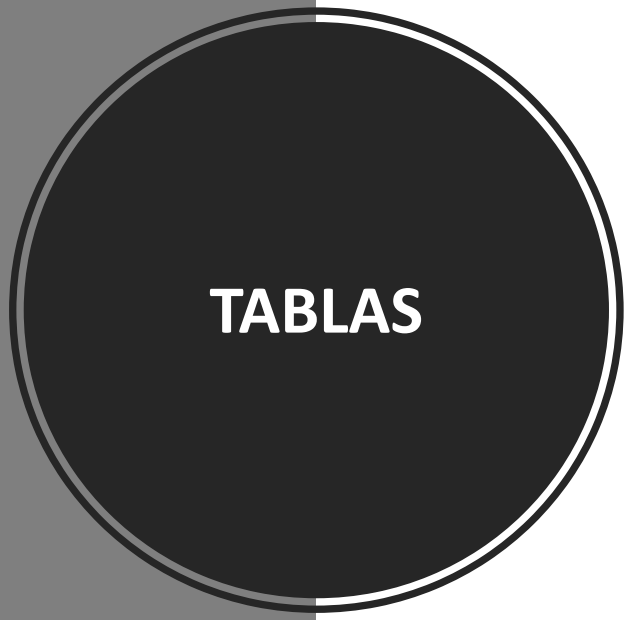
Elementos de una base de datos

Semana 5

Temas a Desarrollar

Clase 5

- Vistas y tablas.Vistas almacenadas versus vistas temporales.
- Arquitectura medallón
- UDF y procedimientos almacenados (solo teoría)
- Data lake versus data warehouse.
- SQL: qué es una query. Generalmente
- DDL, DCL



Tablas - La Base del Almacenamiento de Datos

Una tabla es la estructura fundamental en una base de datos relacional, organizada en filas (registros) y columnas (campos/atributos).

- **Almacenamiento Físico:**
  - Los datos de las tablas se almacenan físicamente en el disco duro del servidor de la base de datos.
  - La forma exacta del almacenamiento físico puede variar según el sistema de gestión de bases de datos (DBMS) utilizado (ej. MySQL, PostgreSQL, SQL Server, Oracle).
  - Generalmente, los datos se organizan en archivos y bloques de datos en el disco.

Almacenamiento en Memoria (Caché):

- Para mejorar el rendimiento, los DBMS utilizan la memoria RAM del servidor para almacenar en caché las partes de las tablas que se **acceden con frecuencia.**
- Esto permite que las consultas posteriores a los mismos datos sean mucho más rápidas, ya que no es necesario leerlos desde el disco.
- El DBMS gestiona automáticamente esta caché, decidiendo qué datos mantener en memoria según patrones de acceso y disponibilidad de recursos.

VISTAS

Una vista es una tabla virtual basada en el resultado de una consulta SQL. No almacena datos por sí misma.

Propósito:

- Simplificar consultas complejas.
- Proporcionar una perspectiva específica de los datos a los usuarios.
- Mejorar la seguridad al limitar el acceso a ciertas columnas o filas.
- Ofrecer una capa de abstracción sobre la estructura de las tablas subyacentes.

Tomemos por ejemplo que el equipo de marketing quiere una tabla con los clientes de Buenos Aires para la tabla clientes

id_cliente	nombre	apellido	email	ciudad
1	Ana	Pérez	ana.perez@email.com	Buenos Aires
2	Juan	Gómez	juan.gomez@email.com	Córdoba
3	María	López	maria.lopez@email.com	Buenos Aires
4	Carlos	Ruiz	carlos.ruiz@email.com	Mendoza

El script de SQL sería

```sql
CREATE VIEW ClientesBuenosAires AS
SELECT Nombre, Apellido, Email
FROM Clientes
WHERE Ciudad = 'Buenos Aires';
```

VISTAS

Entonces cada vez que el equipo de marketing quiera acceder a esos mismos datos ya no tiene que escribir una query filtrada. Sino que simplemente debería correr:

```
SELECT Nombre, Apellido, Email
FROM ClientesBuenosAires;
```

Y el resultado sería:

nombre	apellido	email
Ana	Pérez	ana.perez@email.com
María	López	maria.lopez@email.com

Vistas Almacenadas (Materialized Views)

Una vista almacenada (en algunos DBMS se denomina "vista materializada") es una vista cuyo resultado se guarda físicamente en el disco.

Almacenamiento en Memoria:

- Al igual que las tablas, los datos de una vista almacenada se guardan en el disco.
- Por lo tanto, también pueden ser cacheados en la memoria RAM del servidor para mejorar el rendimiento de las consultas que las utilizan.

Ventajas:

- Mejor rendimiento para consultas frecuentes que utilizan la vista, ya que los datos ya están pre calculados y almacenados.
- Reduce la carga en el servidor al no tener que recalcular la consulta de la vista cada vez que se accede a ella.

Desventajas:

- Requiere espacio de almacenamiento adicional.
- Necesita ser actualizada cuando cambian los datos de las tablas base (la estrategia de actualización puede ser inmediata o diferida, dependiendo del DBMS y la configuración).
- La gestión de la actualización puede generar una sobrecarga adicional.

Vistas Temporales (Virtual Views)

Una vista temporal (o simplemente "vista") es una vista que solo existe durante la duración de la sesión actual o hasta que se elimina explícitamente. Su definición se guarda, pero los resultados no se almacenan físicamente.

Almacenamiento en Memoria:

- La *definición* de la vista (la consulta SQL) se almacena en el catálogo del sistema de la base de datos.
- Los *resultados* de la consulta que define la vista se generan dinámicamente cada vez que se consulta la vista. Estos resultados pueden residir temporalmente en la memoria durante la ejecución de la consulta, pero no se persisten en el disco como una estructura separada.

Ventajas:

- No requiere espacio de almacenamiento adicional para los datos de la vista.
- Siempre refleja los datos más recientes de las tablas base, ya que se recalcula cada vez.
- Útil para simplificar consultas complejas que se utilizan con frecuencia dentro de una sesión.

Desventajas:

- Puede tener un menor rendimiento en comparación con las vistas almacenadas para consultas complejas o frecuentes, ya que la consulta debe ejecutarse cada vez.

VISTAS

Característica	Vistas Almacenadas (Materialized)	Vistas Temporales (Virtual)
Almacenamiento de Datos	Físico en disco	No almacena datos
Rendimiento de Lectura	Generalmente mejor para consultas frecuentes	Puede ser menor para consultas complejas
Espacio de Almacenamiento	Requiere espacio adicional	No requiere espacio adicional
Actualización de Datos	Necesita ser actualizada	Siempre refleja datos actuales
Sobrecarga de Escritura	Puede tener sobrecarga al actualizar	No tiene sobrecarga directa
Persistencia	Persistente entre sesiones	Temporal (solo durante la sesión)

VISTAS Y TABLAS

Casos de Uso

Tablas: Almacenamiento de la información fundamental de la aplicación (clientes, productos, pedidos, etc.).

Vistas Temporales:

- Simplificar consultas complejas para informes ad-hoc.
- Proporcionar una interfaz más amigable para usuarios no técnicos.
- Implementar lógicas de negocio complejas sin modificar la estructura de las tablas base.

Vistas Almacenadas:

- Mejorar el rendimiento de dashboards e informes que se generan con frecuencia.
- Crear agregaciones precalculadas (ej. totales, promedios) para acelerar las consultas.
- Optimizar consultas complejas en entornos de data warehousing.

ARQUITECTURA MEDALLÓN

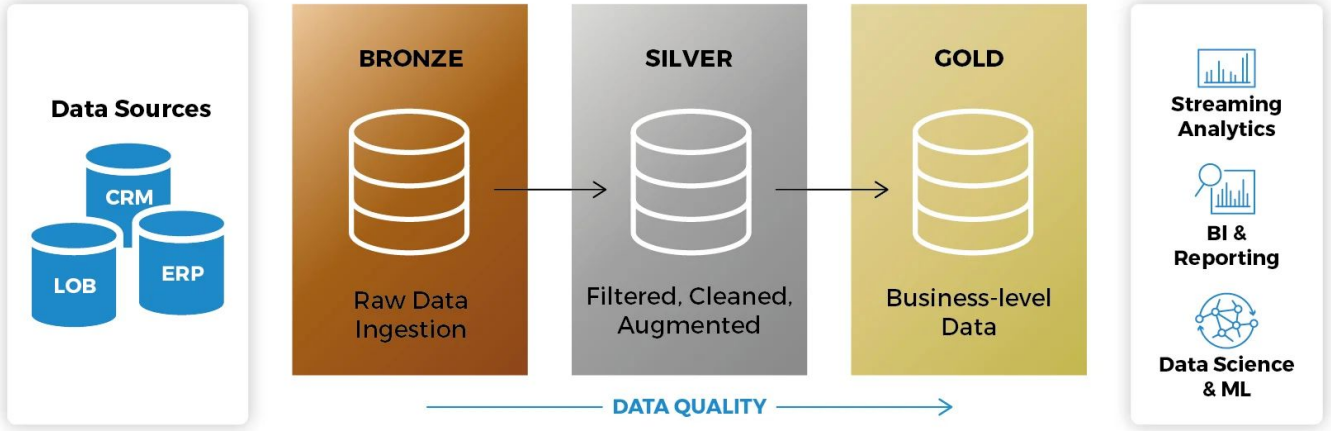
La Arquitectura Medallón es un patrón de diseño de datos utilizado en data lakes y plataformas de procesamiento de datos a gran escala.

Se asemeja a una medalla con diferentes anillos o capas, donde cada capa representa un nivel de refinamiento y calidad de los datos.

Objetivo Principal: Organizar los datos en diferentes etapas de procesamiento para mejorar la calidad, la gobernanza y la eficiencia del análisis.

Beneficios Clave:

- **Calidad de Datos Mejorada:** A través de la limpieza y transformación en cada capa.
- **Gobernanza Simplificada:** Facilita el seguimiento del linaje de los datos y la aplicación de reglas.
- **Optimización del Rendimiento:** Cada capa está optimizada para su propósito específico.
- **Flexibilidad:** Permite trabajar con diferentes tipos de datos y casos de uso.



ARQUITECTURA MEDALLÓN
BRONZE LAYER

Bronce (Raw/Landing Zone)

Propósito: Almacenar los datos en su formato original, tal como se reciben de las fuentes.

Características:

- Datos brutos, sin transformaciones significativas.
- Retención del esquema original (si existe).
- Puede incluir datos estructurados, semiestructurados y no estructurados.
- Enfoque en la ingesta rápida y confiable.

Por ejemplo, en la siguiente tabla podemos ver que el id_cliente = 1 se encuentra 2 veces ya que el cliente se mudó de provincia y el dato ha llegado 2 días distintos. En una capa bronce, se conservan ambos datos.

id_cliente	nombre	apellido	email	ciudad	fecha_del_dato
1	**Ana**	**Pérez**	**Ana.Perez@email.com**	**Buenos Aires**	**2025-01-03**
2	Juan	Gómez	juan.gomez@email.com	Córdoba	2025-01-03
3	María	López	maria.lopez@email.com	Buenos Aires	2025-01-03
4	Carlos	Ruiz	Carlos.Ruiz@email.com	Mendoza	2025-01-03
1	**Ana**	**Pérez**	**Ana.Perez@email.com**	**Córdoba**	**2025-01-07**

ARQUITECTURA MEDALLÓN
SILVER LAYER

Plata (Curated/Transformed Zone)

Propósito: Almacenar datos que han sido limpiados, estandarizados y transformados para un consumo más fácil.

Características:

- Aplicación de reglas de calidad de datos (detección y corrección de errores).
- Estandarización de formatos y unidades.
- Enriquecimiento de datos .
- Creación de un esquema consistente y optimizado para el análisis.

Por ejemplo, en la siguiente tabla podemos ver que el id_cliente = 1 se encuentra una sola vez ya que solo nos interesa conservar el dato más reciente. Además, almacenamos todos los emails en minúsculas. Estos cambios se deciden dependiendo de las reglas de negocio.

id_cliente	nombre	apellido	email	ciudad	fecha_del_dato
1	Ana	Pérez	ana.perez@email.com	Córdoba	2025-01-07
2	Juan	Gómez	juan.gomez@email.com	Córdoba	2025-01-03
3	María	López	maria.lopez@email.com	Buenos Aires	2025-01-03
4	Carlos	Ruiz	carlos.ruiz@email.com	Mendoza	2025-01-03

ARQUITECTURA MEDALLÓN
GOLD LAYER

Oro (Serving/Bussiness Zone)

Propósito: Almacenar datos altamente refinados, agregados y modelados para casos de uso analíticos específicos y el consumo por parte de usuarios finales.

Características:

- Datos listos para informes, dashboards y análisis.
- Modelos dimensionales (ej. Star Schema, Snowflake Schema).
- Agregaciones y cálculos predefinidos.
- Enfoque en el rendimiento de las consultas y la facilidad de uso.

Por ejemplo, si vamos a necesitar construir un reporte que indique cuántos clientes hay por provincia podemos almacenar la siguiente tabla en la capa de oro:

provincia	cantidad_de_clientes
Buenos Aires	1
Córdoba	2
Mendoza	2

UDF y Store Procedures

Funciones Definidas por el Usuario (UDFs)

Una Función Definida por el Usuario (UDF) es un bloque de código SQL (o en algunos DBMS, en lenguajes procedurales como PL/pgSQL, T-SQL, PL/SQL) que acepta parámetros de entrada, realiza un cálculo o una serie de operaciones, y devuelve un único valor escalar o una tabla.

Propósito Principal:

- **Encapsular lógica reutilizable:** Evita la repetición del mismo código en múltiples consultas.
- **Simplificar consultas complejas:** Permite abstraer cálculos o transformaciones complejas en una función fácil de usar.
- **Mejorar la legibilidad de las consultas:** Las consultas se vuelven más concisas y fáciles de entender al invocar funciones.
- **Aplicar lógica de negocio consistente:** Asegura que las mismas reglas de cálculo o transformación se apliquen en toda la base de datos.

Tipos de UDFs (según el DBMS):

- **Escalares:** Devuelven un único valor (ej. un número, una cadena de texto, una fecha).
- **Tabulares :** Devuelven un conjunto de filas (una tabla) basado en los parámetros de entrada y una única sentencia.

UDF y Store Procedure

Procedimientos Almacenados

Un Procedimiento Almacenado (Stored Procedure) es un bloque de código SQL (o en lenguajes procedurales) que se almacena y se ejecuta directamente en el servidor de la base de datos. Puede aceptar parámetros de entrada, realizar una secuencia de operaciones (incluyendo consultas, actualizaciones, inserciones, eliminaciones, lógica de control de flujo), y opcionalmente devolver parámetros de salida.

Propósito Principal:

- **Encapsular lógica de negocio compleja:** Permite agrupar múltiples pasos y operaciones en una sola unidad ejecutable.
- **Mejorar el rendimiento:** Almacenar el código en el servidor reduce la necesidad de enviar múltiples consultas desde la aplicación, disminuyendo el tráfico de red y permitiendo al DBMS optimizar la ejecución.
- **Mejorar la seguridad:** Permite controlar el acceso a los datos al otorgar permisos de ejecución a los procedimientos almacenados en lugar de acceso directo a las tablas.
- **Promover la consistencia:** Asegura que las operaciones complejas se realicen de la misma manera cada vez que se ejecutan.
- **Modularidad:** Facilita la organización y el mantenimiento del código de la base de datos.

Características Clave:

- Pueden contener múltiples sentencias SQL.
- Pueden incluir lógica de control de flujo (ej. `IF-ELSE`, bucles).
- Pueden aceptar parámetros de entrada y devolver parámetros de salida.
- Pueden realizar operaciones DML (INSERT, UPDATE, DELETE) y DDL (en algunos casos limitados).
- Se ejecutan directamente en el servidor de la base de datos.

UDF y Store Procedure

	Funciones Definidas por el Usuario (UDFs)	Procedimientos Almacenados
Propósito Principal	Cálculo o transformación de datos para su uso en consultas	Encapsulación de lógica de negocio y operaciones complejas
Valor de Retorno	Generalmente un único valor escalar o una tabla	Puede devolver parámetros de salida (múltiples valores) o ningún valor explícito
Uso en Consultas	Se invocan directamente dentro de las cláusulas de las consultas (ej. SELECT, WHERE)	Se ejecutan mediante comandos específicos (CALL, EXEC)
Efectos Secundarios	Generalmente diseñadas para no tener efectos secundarios (modificación de datos)	Pueden tener efectos secundarios (modificar datos, realizar otras operaciones en la base de datos)
Lógica de Control de Flujo	Limitada, principalmente expresiones	Pueden incluir lógica de control de flujo compleja (IF-ELSE, bucles)
Operaciones DML	Generalmente no realizan operaciones DML directamente	Pueden realizar operaciones DML (INSERT, UPDATE, DELETE)
Transacciones	Participan en la transacción de la consulta que las invoca	Pueden gestionar sus propias transacciones (dependiendo del DBMS)

Data Lakehouse vs Data Warehouse

Data Lake

Un Data Lake es un repositorio centralizado que permite almacenar grandes volúmenes de datos en su formato original, sin necesidad de [1] estructurarlos previamente.

Características Clave:

- **Almacenamiento de Datos Crudos:** Guarda datos en su forma nativa, ya sean estructurados (bases de datos relacionales), semiestructurados (JSON, XML, CSV) o no estructurados (texto, imágenes, audio, video).
- **Esquema en Lectura (Schema-on-Read):** El esquema de los datos no se define al momento de la ingesta, sino cuando se consultan y analizan. Esto proporciona gran flexibilidad.
- **Escalabilidad:** Diseñado para manejar grandes cantidades de datos (terabytes, petabytes o incluso más) de manera rentable, a menudo utilizando almacenamiento de objetos de bajo costo.
- **Diversidad de Usuarios:** Atrae a científicos de datos, ingenieros de datos y analistas que necesitan explorar y descubrir patrones en los datos.
- **Agilidad:** Permite la ingesta rápida de nuevos datos sin la necesidad de un modelado inicial complejo.

Propósito Principal: Exploración de datos, descubrimiento, ciencia de datos, machine learning, análisis avanzado, almacenamiento de datos diversos para casos de uso futuros aún no definidos.

Data Lakehouse vs Data Warehouse

Data Warehouse

Un Data Warehouse es un repositorio centralizado de datos estructurados que han sido filtrados, limpiados, transformados y catalogados para un propósito específico: el análisis y la inteligencia de negocios (BI).

Características Clave:

- **Almacenamiento de Datos Procesados:** Contiene datos limpios, transformados e integrados de diversas fuentes, organizados en un esquema predefinido.
- **Esquema en Escritura (Schema-on-Write):** El esquema de los datos se define cuidadosamente antes de la ingesta. Esto asegura la calidad y consistencia de los datos.
- **Optimizado para Consultas:** Estructurado para facilitar consultas complejas y análisis de tendencias históricas.
- **Gobernanza y Seguridad:** Implementa políticas robustas de gobernanza, seguridad y calidad de datos.
- **Usuarios Principalmente Analistas de Negocios:** Diseñado para usuarios que necesitan generar informes, dashboards y realizar análisis para la toma de decisiones empresariales.

Propósito Principal: Inteligencia de negocios (BI), informes, dashboards, análisis de tendencias históricas, soporte a la toma de decisiones estratégicas.

Query

Una query en SQL es una instrucción o un conjunto de instrucciones escritas en lenguaje SQL que se envían a un RDBMS para realizar una o varias operaciones sobre los datos almacenados.
Propósito Principal: El objetivo fundamental de una query es **recuperar información específica** de una o más tablas dentro de la base de datos.

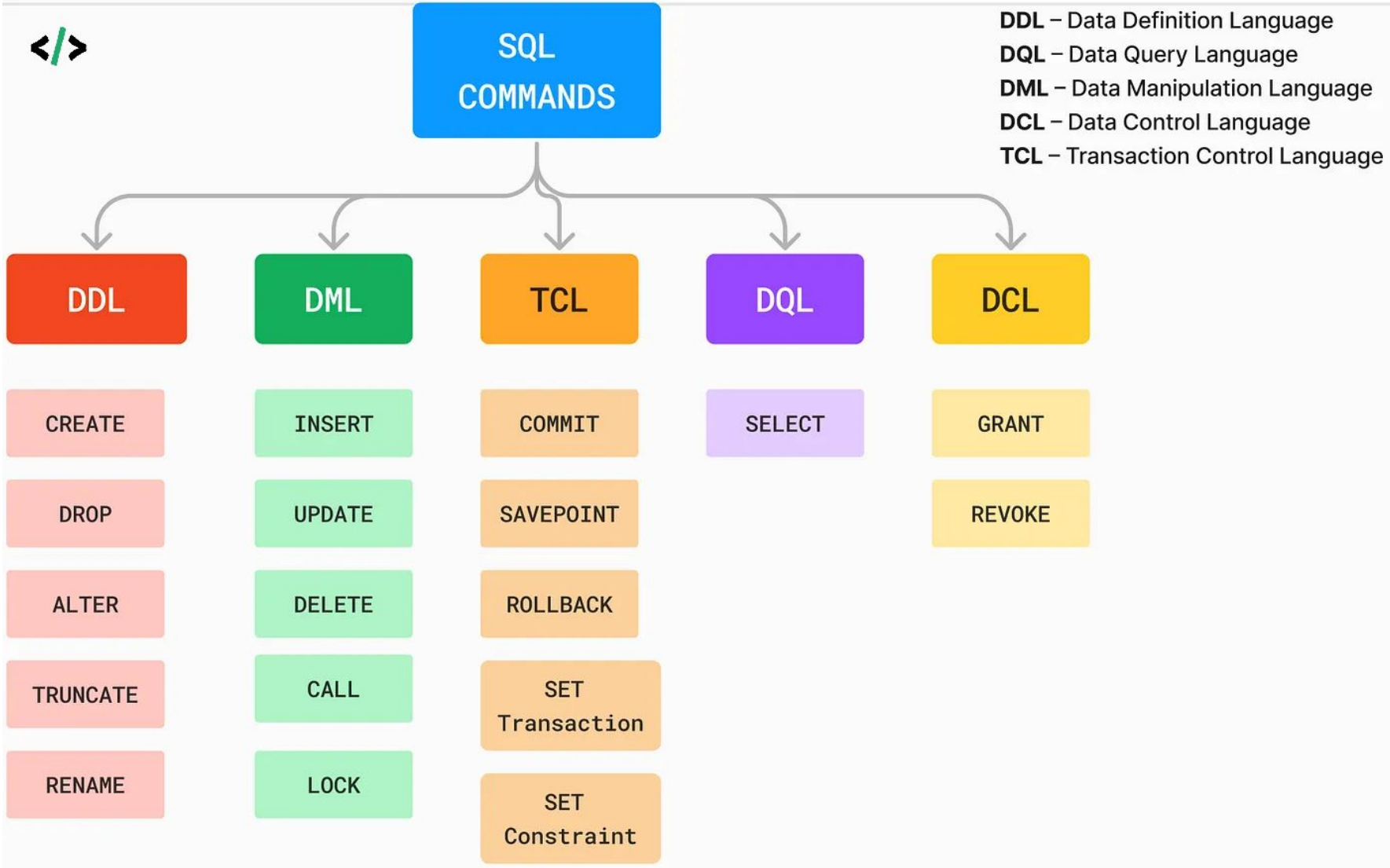
La mayoría de las veces, cuando hablamos de una "query", nos referimos a una sentencia `SELECT`, que se utiliza para especificar qué datos queremos recuperar.

El Proceso de Ejecución de una Query

- **Envío al Servidor:** La aplicación cliente (ej. un software de gestión de bases de datos, una aplicación web) envía la query SQL al servidor de la base de datos.
- **Análisis y Validación:** El servidor analiza la sintaxis de la query para asegurarse de que sea válida según las reglas de SQL.
- **Optimización:** El optimizador de consultas del DBMS determina la forma más eficiente de acceder a los datos solicitados (ej. qué índices usar, en qué orden acceder a las tablas).
- **Ejecución:** El motor de ejecución del DBMS lleva a cabo el plan de ejecución optimizado, accediendo a los datos almacenados.
- **Retorno de Resultados:** El servidor envía el conjunto de resultados (los datos solicitados) de vuelta a la aplicación cliente.

```
SELECT *
FROM Clientes;
```

Sublenguajes de SQL



CREACIÓN Y REEMPLAZO DE TABLAS

- **`CREATE TABLE`:** Crea una nueva tabla con nombre y definición de columnas (nombre, tipo de dato, restricciones).
  - **`PRIMARY KEY (columna)`:** Define una o más columnas como la clave primaria de la tabla (identificador único).

Existen dos variaciones del comando CREATE TABLE que nos ayudan a evitar errores del tipo “la tabla que querés crear ya existe”:

- **`CREATE OR REPLACE TABLE nombre_tabla (...)`:**
  - Si la tabla no existe, la crea.
  - Si la tabla existe, la **reemplaza completamente** con la nueva definición (se pierden los datos).
- **`CREATE TABLE IF NOT EXISTS nombre_tabla (...)`:**
  - Si la tabla no existe, la crea.
  - Si la tabla existe, no hace nada.

```sql
CREATE OR REPLACE TABLE Usuarios (
    UserID INT PRIMARY KEY,
    Nombre VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    FechaCreacion DATE DEFAULT CURRENT_DATE
);
```

ELIMINACIÓN DE FILAS Y TABLAS

Comando	Propósito	Registros Afectados	Estructura Tabla	Rollback
DROP	Elimina la tabla y su estructura.	Todos los registros	Eliminada	Depende del DBMS y configuración
DELETE	Elimina registros específicos (con WHERE).	Los que cumplen la condición	Intacta	Sí
TRUNCATE	Elimina todos los registros rápidamente.	Todos los registros	Intacta	No (generalmente)

Existe una variación del comando DROP TABLE para evitar errores del tipo “la tabla que querés eliminar no existe”:

- `DROP TABLE IF EXISTS nombre_tabla;`: Elimina la tabla si y solo si ya existe.