

ORACLE

- INSTRUCTOR
- I.S.C. E.T.W. M.T.W. CARLOS URIEL DE JESÚS SÁNCHEZ GONZÁLEZ
- [HTTPS://WWW.LINKEDIN.COM/IN/CARLOS-URIEL-DE-JESUS-SANCHEZ-GONZALEZ-8920A1372/](https://www.linkedin.com/in/carlos-uriel-de-jesus-sanchez-gonzalez-8920a1372/)

¿QUÉ ES ORACLE?

- Oracle Database (u “Oracle DB”) es un sistema de gestión de bases de datos relacional (RDBMS) desarrollado por Oracle Corporation.
- Es uno de los gestores de base de datos más usados a nivel empresarial.
- Usa SQL como lenguaje principal para manejar datos.
- Está diseñado para aplicaciones de misión crítica: bancos, aerolíneas, telecomunicaciones, ERPs, etc.
-  Características destacadas:
- Soporta transacciones ACID (consistencia y confiabilidad de datos).
- Maneja grandes volúmenes de datos.
- Tiene herramientas de seguridad, replicación, backup y recuperación.
- Corre en múltiples plataformas (Windows, Linux, Unix, en la nube).
- Desde versiones recientes, maneja también JSON, XML, Spatial Data y Big Data.

¿QUÉ ES UN GESTOR DE BASE DE DATOS?

- Un gestor de base de datos (DBMS – Database Management System) es un software que:
- Crea y administra bases de datos.
- Almacena datos de manera estructurada.
- Permite consultar, modificar y eliminar información.
- Controla la seguridad y los accesos de los usuarios.
- Se asegura de que los datos sean consistentes y confiables.

TIPOS PRINCIPALES DE GESTORES

- Relacionales (RDBMS) → usan tablas con filas y columnas.
 - Ejemplos: Oracle, MySQL, PostgreSQL, SQL Server.
- NoSQL → orientados a documentos, grafos, clave-valor, etc.
 - Ejemplos: MongoDB, Cassandra, Redis.
- En resumen:
 - **Oracle** es un **gestor de bases de datos relacional** muy potente, usado en grandes empresas.
 - Un **gestor de base de datos (DBMS)** es el software que te permite **crear, organizar, consultar y proteger** los datos.

¿QUÉ ES UNA BASE DE DATOS?

- Una base de datos es un conjunto organizado de información que se guarda de forma estructurada en un sistema, para que pueda ser accedida, gestionada y actualizada fácilmente.
-  En palabras simples:
- Es como una biblioteca digital donde los datos están clasificados y se pueden buscar, agregar, modificar o eliminar de manera rápida y segura.
-  Características principales
-  Organización → los datos no están “tirados”, sino que siguen un modelo (tablas, documentos, grafos, etc.).
-  Acceso rápido → permite consultas con lenguajes como SQL.
-  Seguridad → controla quién puede ver o modificar la información.
-  Consistencia → asegura que los datos sean correctos y actualizados.

TIPOS DE BASES DE DATOS

- Relacionales (RDBMS)
 - Datos organizados en tablas (filas y columnas).
 - Se usan en la mayoría de sistemas empresariales.
 - Ejemplos: MySQL, Oracle, PostgreSQL, SQL Server.
- No relacionales (NoSQL)
 - Usan otros modelos: documentos, clave-valor, grafos, columnas.
 - Ideales para Big Data y sistemas distribuidos.
 - Ejemplos: MongoDB (documentos), Redis (clave-valor), Neo4j (grafos).
- En memoria
 - Los datos se almacenan directamente en la RAM → súper rápidas.
 - Ejemplo: Redis.
- Distribuidas
 - La información se guarda en varios servidores, pero parece una sola base de datos.
 - Ejemplo: Cassandra, CockroachDB.

TIPOS DE DATOS NUMÉRICOS

- **NUMBER(p, s)** → números con precisión y escala.p = número total de dígitos.s = dígitos después del punto decimal.
- Ej: **NUMBER(5,2)** → permite valores como 123.45.
- **INTEGER / INT** → alias de NUMBER sin decimales.
- **FLOAT** → número en coma flotante (precisión simple).
- **DOUBLE** → número en coma flotante (precisión doble).

TIPOS DE DATOS CARACTERES

- CHAR(n) → texto de longitud fija (hasta 2000).
- VARCHAR2(n) → texto de longitud variable (hasta 4000).
- CLOB → texto muy grande (Character Large Object), hasta 4 GB.
- NCHAR(n) / NVARCHAR2(n) → como los anteriores pero para Unicode (caracteres multilingües).
- NCLOB → igual que CLOB pero en Unicode.

TIPOS DE DATOS FECHAS Y TIEMPOS

- DATE → fecha + hora (dd-mm-yyyy hh24:mi:ss).
- TIMESTAMP → como DATE pero con fracciones de segundo.
- TIMESTAMP WITH TIME ZONE → incluye zona horaria.
- TIMESTAMP WITH LOCAL TIME ZONE → ajusta a la zona del servidor.
- INTERVAL YEAR TO MONTH → intervalo de tiempo en años y meses.
- INTERVAL DAY TO SECOND → intervalo de tiempo en días, horas, minutos y segundos.

TIPOS DE DATOS BINARIOS

- **RAW(n)** → datos binarios de longitud fija (hasta 2000 bytes).
- **LONG RAW** → datos binarios más largos (obsoleto).
- **BLOB** → Binary Large Object, hasta 4 GB (ej. imágenes, videos, PDFs).
- **BFILE** → referencia a un archivo binario almacenado fuera de la BD.

TIPOS DE DATOS ESPECIALES

- ROWID → identificador único de una fila en la tabla.
- UROWID → ROWID universal (para tablas indexadas o remotas).
- XMLTYPE → para almacenar y consultar datos XML.
- JSON (desde Oracle 21c soportado como tipo nativo).

TABLA

- Es una estructura lógica que organiza los datos en filas y columnas.
- Representa una entidad del mundo real (ejemplo: Clientes, Productos, Pedidos).
- Cada tabla tiene un nombre único en la base de datos.

FILA (REGISTRO, TUPLA)

- Cada fila es un registro individual dentro de la tabla.
- Contiene todos los valores de una entidad concreta.
- Ejemplo: La fila (1, 'Ana', 'ana@mail.com') representa a un cliente específico.

CAMPO (COLUMNA, ATRIBUTO)

- Es cada **columna** de la tabla.
- Representa una **propiedad** de la entidad.
- Tiene un tipo de dato definido (número, texto, fecha, etc.).
- Ejemplo: en la tabla Cliente → `id_cliente`, `nombre`, `correo` son campos.

CARDINALIDAD

- La cardinalidad describe la relación entre tablas (cuántas filas de una tabla pueden estar relacionadas con filas de otra).
- Tipos principales: 1 a 1 (1:1) → Una fila de la tabla A se relaciona con una sola fila de la tabla B.
 - Ejemplo: Persona  Pasaporte.
- 1 a muchos (1:N) → Una fila de la tabla A puede relacionarse con muchas filas de la tabla B.
 - Ejemplo: Cliente  Pedidos (un cliente hace varios pedidos).
- Muchos a muchos (N:M) → Muchas filas de la tabla A se relacionan con muchas de la tabla B.
 - Ejemplo: Alumno  Curso (un alumno toma varios cursos y un curso tiene varios alumnos). Esto se resuelve con una tabla intermedia (tabla puente).

¿QUÉ ES UN DIAGRAMA ENTIDAD–RELACIÓN (ERD)?

- Es una representación gráfica del modelo lógico de una base de datos.
- Muestra entidades (tablas), sus atributos (campos) y las relaciones (cómo se conectan entre sí).
- Se usa en la fase de diseño de bases de datos, antes de implementarlas físicamente en un gestor (Oracle, MySQL, PostgreSQL, etc.).
- ➡️ Es como el plano arquitectónico de una base de datos.

ELEMENTOS PRINCIPALES DE UN ERD

- Entidad 

 - Representa un objeto del mundo real (ejemplo: Cliente, Producto).
 - Se dibuja como un rectángulo.

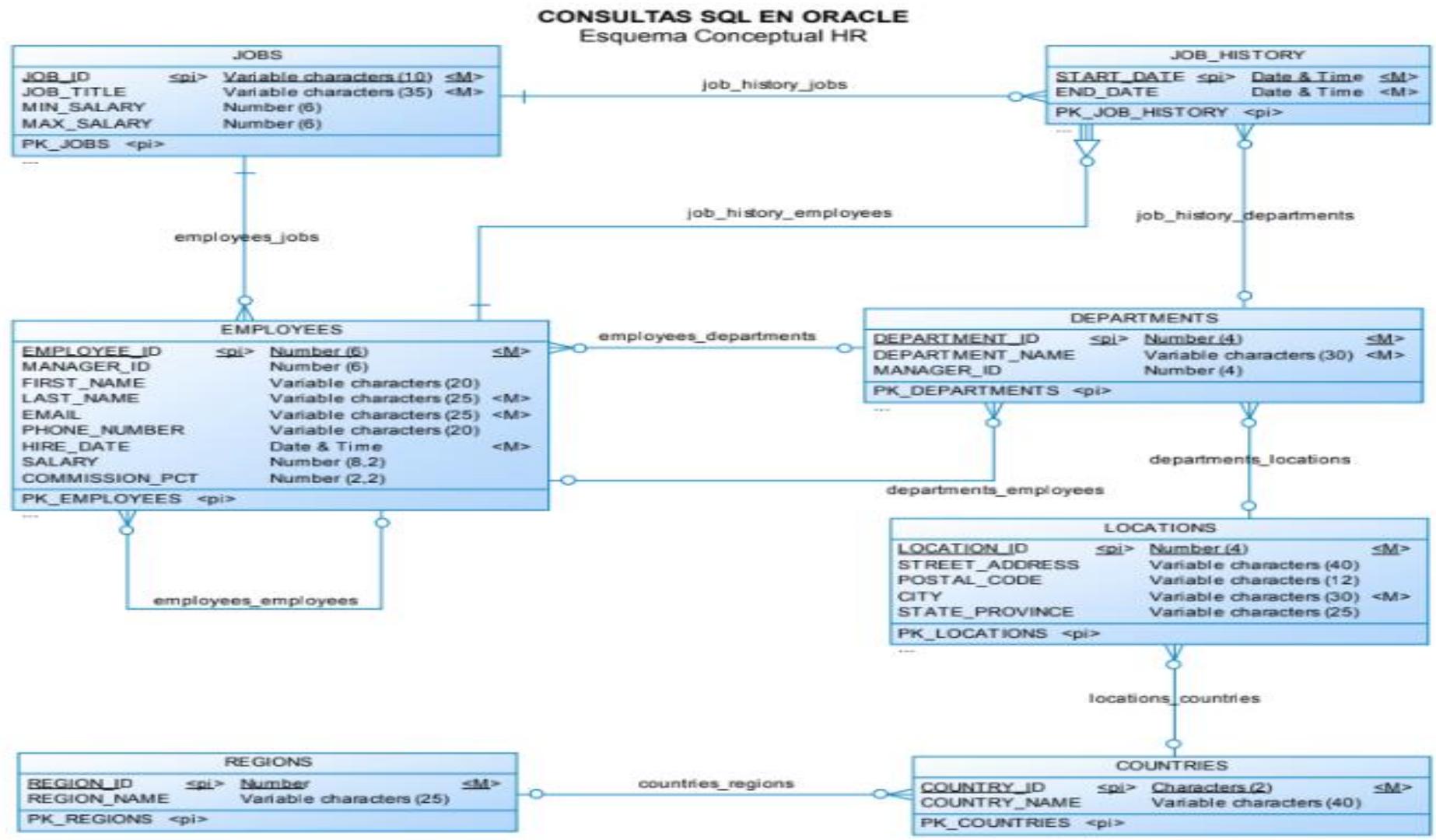
- Atributos  Son las propiedades de la entidad (ejemplo: nombre, correo).

 - Se dibujan como óvalos conectados al rectángulo.
 - El atributo que identifica a la entidad (clave primaria) se subraya.

- Relaciones 

 - Explican cómo se vinculan dos o más entidades.
 - Se dibujan como un rombo.
 - Llevan la cardinalidad (1:1, 1:N, N:M).

MODELO ENTIDAD - RELACIÓN



¿QUÉ ES LA NORMALIZACIÓN?

- La normalización es un proceso mediante el cual se organizan las tablas y los datos de una base de datos relacional para:
- Evitar duplicidad de datos.
- Evitar inconsistencias.
- Mejorar la eficiencia en las consultas.
- Se hace mediante formas normales (FN), que son reglas que indican cómo deben estructurarse las tablas.

PRIMERA FORMA NORMAL (1FN)

- Regla:Cada columna debe contener un solo valor (no listas o arrays).Cada fila debe ser única.
 - Ejemplo NO 1FN:

id_cliente	nombre	telefonos
1	Ana	555-1111, 555-2222
2	Carlos	555-3333

- Problema:La columna telefonos tiene varios valores.
 - Solución 1FN:

id_cliente	nombre	telefono
1	Ana	555-1111
1	Ana	555-2222
2	Carlos	555-3333

SEGUNDA FORMA NORMAL (2FN)

- Cumple 1FN.
- Todas las columnas no clave dependen completamente de la clave primaria, no solo de una parte de ella.
- Esto aplica cuando la clave primaria es compuesta. Ejemplo No “FN”

id_pedido	id_producto	nombre_producto	cantidad
101	1	Laptop	2
101	2	Mouse	1

Problema: **nombre_producto** depende solo de **id_producto**, no de toda la clave (**id_pedido, id_producto**).

Solución 2FN: Separar en dos tablas:

id_pedido	id_producto	cantidad
101	1	2
101	2	1
id_producto	nombre_producto	
1	Laptop	
2	Mouse	

TERCERA FORMA NORMAL (3FN)

- Regla:
- Cumple 2FN.
- Ninguna columna no clave depende de otra columna no clave.
- Evita dependencias transitivas. Ejemplo NO 3FN:

id_cliente	nombre	ciudad	codigo_postal
1	Ana	León	37000
2	Carlos	Guanajuato	36000

Problema:

codigo_postal depende de **ciudad**, no directamente de **id_cliente**.

Solución 3FN:

id_cliente	nombre	id_ciudad	codigo_postal
1	Ana	1	37000
2	Carlos	2	36000
id_ciudad	ciudad		
1	León		
2	Guanajuato		

RESUMEN RÁPIDO

Forma Normal

1FN

2FN

3FN

Qué evita

Valores múltiples en una columna

Dependencia parcial de la clave primaria

Dependencia transitiva entre columnas no clave



En resumen:
La normalización organiza tus tablas para que no haya datos repetidos ni inconsistentes, haciendo tu base de datos más eficiente y fácil de mantener.

DDL (DATA DEFINITION LANGUAGE)



- Son sentencias que definen o modifican la estructura de la base de datos: tablas, índices, esquemas, etc.
- No manejan los datos en sí, sino cómo se almacenan.
- Sentencias DDL principales:

Sentencia

CREATE

ALTER

DROP

TRUNCATE

RENAME

Qué hace

Crea un objeto (tabla, vista, índice, secuencia, etc.)

Modifica la estructura de un objeto existente

Elimina un objeto de la base de datos

Elimina todos los datos de una tabla (estructura intacta)

Cambia el nombre de un objeto

DML (DATA MANIPULATION LANGUAGE)

- Son sentencias que manipulan los datos dentro de las tablas: insertar, modificar, eliminar o consultar.
- Actúan sobre el contenido de las estructuras creadas con DDL.
- Sentencias DML principales:

Sentencia

INSERT

UPDATE

DELETE

MERGE

SELECT

Qué hace

Inserta nuevos registros en una tabla

Modifica registros existentes

Elimina registros de la tabla

Combina insert/update según exista o no el registro

Consulta o extrae datos de la tabla

RESUMEN RÁPIDO

- En resumen:
- **DDL** → estructura / esquema de la base de datos.
- **DML** → datos dentro de las estructuras.

Tipo

DDL

DML

Qué hace

Define la estructura de la BD

Manipula los datos en la BD

Ejemplos

CREATE, ALTER, DROP, TRUNCATE

INSERT, UPDATE, DELETE, SELECT,
MERGE

FUNCIÓN



- Una función es un bloque de código PL/SQL que realiza una tarea específica y devuelve un valor único.
- Se usa para calcular, transformar o procesar datos y luego devolver un resultado.
- Las funciones pueden recibir parámetros de entrada, pero siempre devuelven un valor con RETURN.
- Devuelve un valor (a diferencia de un procedimiento que no necesariamente devuelve).
- Puede usarse dentro de sentencias SQL (SELECT, WHERE, etc.) si es determinística.
- Permite reutilizar lógica para no repetir código.

TIPOS DE FUNCIONES

- Funciones definidas por el usuario (UDF)
 - Creas tus propias funciones en PL/SQL.
- Funciones incorporadas (built-in)
 - Oracle ya tiene muchas funciones listas para usar:
 - Numéricas: ROUND(), TRUNC(), MOD()
 - Cadenas: SUBSTR(), LENGTH(), UPPER(), LOWER()
 - Fecha y hora: SYSDATE, ADD_MONTHS(), MONTHS_BETWEEN()
 - Conversión: TO_CHAR(), TO_NUMBER(), TO_DATE()

FUNCIÓN MAX

- MAX() es una función agregada en SQL.
- Sirve para obtener el valor máximo de una columna en un conjunto de registros.
- Funciona principalmente con números, fechas y cadenas (según orden lexicográfico).
- ➡ Es muy usada para saber el mayor valor, la última fecha o el último registro por algún criterio.

FUNCIÓN MIN



- **MIN()** es una función agregada en SQL.
- Sirve para obtener el valor mínimo de una columna en un conjunto de registros.
- Funciona con números, fechas y cadenas (según orden lexicográfico).
- ➡ Se usa para saber el menor valor, la fecha más antigua o el primer registro según algún criterio.

FUNCIÓN COUNT

- COUNT() es una función agregada en SQL.
- Sirve para contar la cantidad de filas que cumplen un criterio dentro de una tabla.
- Muy útil para resúmenes, estadísticas y análisis de datos.

FUNCION AVG



- AVG() significa average, o sea, promedio.
- Es una función agregada que calcula el promedio aritmético de los valores de una columna numérica.
- Solo funciona con columnas numéricas y ignora los valores NULL automáticamente.

FUNCIÓN SUM

- **SUM()** significa suma.
- Es una función agregada que suma todos los valores de una columna numérica de un conjunto de registros.
- Ignora automáticamente los valores NULL.

CLAUSULAS

- Una cláusula es una parte de una sentencia SQL que especifica condiciones o acciones.
- Las cláusulas permiten filtrar, ordenar, agrupar o limitar los datos que queremos consultar o manipular.
- Cada sentencia SQL puede tener varias cláusulas combinadas.

GROUP BY

- Agrupa filas que tienen el mismo valor en una o varias columnas.
- Se usa normalmente con funciones agregadas (SUM, AVG, COUNT, etc.).

ORDER BY

- Ordena los resultados según una o varias columnas.
- Por defecto es ASC (ascendente). Puedes usar DESC (descendente).

LIKE

- Se usa en la cláusula WHERE para buscar patrones dentro de cadenas de texto.
- Permite usar comodines para coincidencias parciales.

BETWEEN

- Se usa en la cláusula WHERE para filtrar valores dentro de un rango.
- Incluye los valores extremos (inclusive).

JOIN (INNER, LEFT, RIGHT, FULL)

- Combina filas de dos o más tablas según una relación.
- Tipos de JOIN:
 - INNER JOIN → solo filas que coinciden en ambas tablas.
 - LEFT JOIN → todas las filas de la izquierda + coincidencias de la derecha.
 - RIGHT JOIN → todas las filas de la derecha + coincidencias de la izquierda.
 - FULL JOIN → todas las filas de ambas tablas.

TRIGGER

- Un trigger (o disparador) es un bloque de código PL/SQL que se ejecuta automáticamente cuando ocurre un evento específico en la base de datos.
- Se asocia generalmente a una tabla o una vista.
- Sirve para automatizar tareas, validar datos o mantener integridad sin necesidad de que el usuario ejecute manualmente un procedimiento.
- Se ejecuta automáticamente al ocurrir un evento (INSERT, UPDATE o DELETE).
- Puede ser antes o después del evento:
 - BEFORE → antes de que se ejecute la operación.
 - AFTER → después de que se ejecuta la operación.
- Puede afectar una fila específica (FOR EACH ROW) o toda la operación.
- No se invoca directamente, se activa por el evento en la tabla.

SECUENCIA

- Una secuencia (SEQUENCE) es un objeto de base de datos que genera valores numéricos únicos de manera automática.
- Se usa principalmente para generar claves primarias o identificadores únicos sin necesidad de calcularlos manualmente.
- Cada vez que se solicita un valor de la secuencia, aumenta automáticamente según su configuración.
- Automática → no necesitas calcular ni incrementar manualmente.
- Única → garantiza valores únicos si se usa correctamente.
- Configurable → se puede definir:
 - Valor inicial (START WITH)
 - Incremento (INCREMENT BY)
 - Valor máximo (MAXVALUE) y mínimo (MINVALUE)
 - Ciclo (CYCLE) o detenerse (NOCYCLE)
 - Cache para mejorar rendimiento (CACHE n)

PROCEDIMIENTO ALMACENADO

- Un procedimiento almacenado (*stored procedure*) es un bloque de código PL/SQL que se guarda dentro de la base de datos.
- Se ejecuta cuando el usuario lo llama y puede realizar una o varias tareas sobre la base de datos.
- Sirve para automatizar procesos, reutilizar lógica y mejorar seguridad y rendimiento.
- Se almacena en la base de datos → no hay que escribir el código cada vez.
- Se puede llamar varias veces → permite reutilización de lógica.
- Puede recibir parámetros de entrada (IN) y devolver valores de salida (OUT).
- Puede manejar transacciones, consultas y operaciones DML (INSERT, UPDATE, DELETE).
- A diferencia de una función, no devuelve obligatoriamente un valor, aunque sí puede hacerlo mediante parámetros OUT.

VISTA (VIEW)

- Una vista (VIEW) es una tabla virtual creada a partir de una consulta SQL.
- No almacena los datos físicamente; simplemente muestra datos de una o varias tablas según la consulta definida.
- Sirve para simplificar consultas complejas, proteger datos sensibles y facilitar la reutilización de consultas.
- Tabla virtual → los datos permanecen en las tablas originales.
- Puede combinar varias tablas usando JOIN.
- Permite filtrar columnas o filas según condiciones.
- Se puede usar como una tabla en consultas SELECT.
- Puede ser actualizable si cumple ciertas condiciones, o solo de lectura.