Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

alkemy

Clase grabada

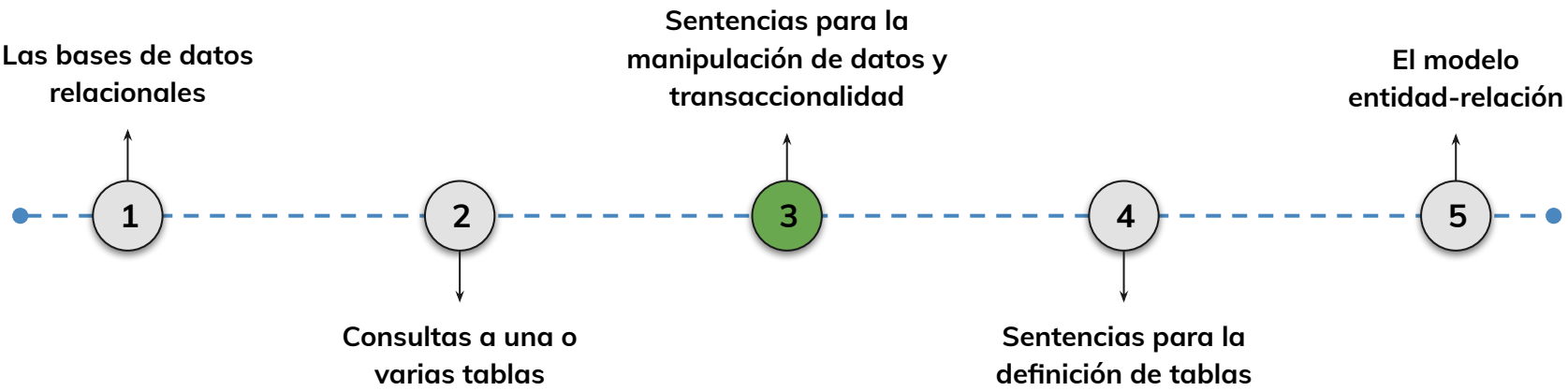
Sentencias para la manipulación de datos y transaccionalidad - Parte 1

AE3: Utilizar lenguaje de manipulación de datos DML para la modificación de los datos existentes en una base de datos dando solución a un problema planteado del entrenamiento.

alkemy

Roadmap de lecciones

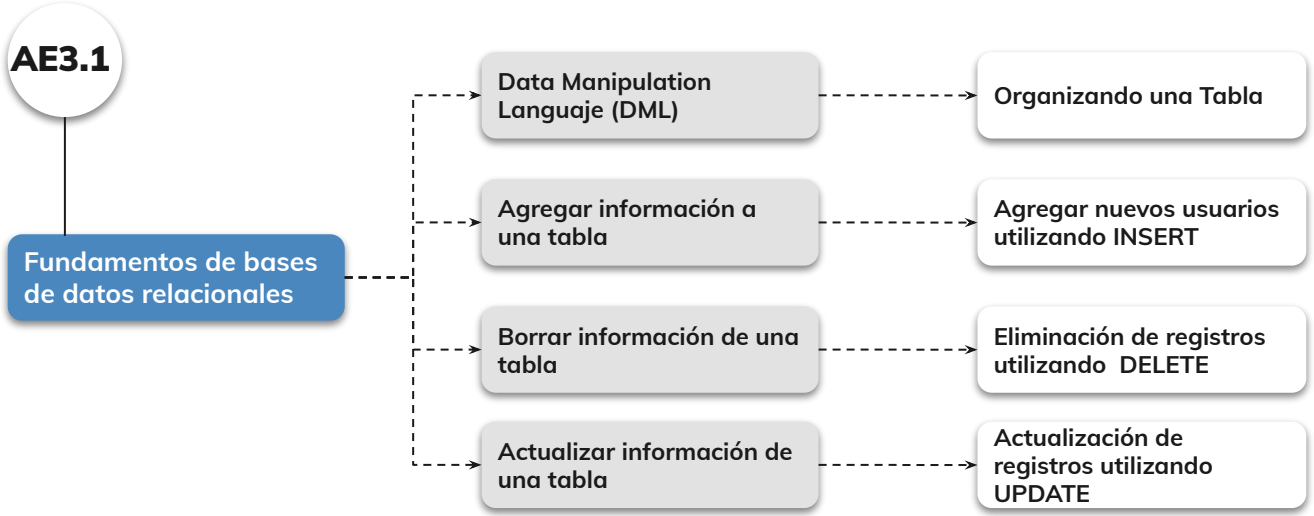
*¿Cuáles **lecciones** estaremos estudiando en este módulo?*



alkemy

Learning Path

¿Cuáles temas trabajaremos hoy?



alkemy

Objetivos de aprendizaje

¿Qué aprenderás?

- Conocer los componentes principales de Data Manipulation Language (DML)
- Insertar información en una tabla utilizando DML
- Actualizar información de una tabla utilizando DML
- Borrar información de una tabla utilizando DML

Repaso clase anterior

¿Quedó alguna duda?

En la clase anterior trabajamos 📚:

- → *Comprender qué son las consultas anidadas*
- → *Realizar consultas anidadas utilizando SQL*
- → *Realizar consultas con funciones de agrupación utilizando SQL*
- → *Realizar consultas con distintos tipos de JOIN*

#Momentode Preguntas...

¿Qué es DML y para qué se utiliza?

¿Cómo borrar datos de una tabla?

¿Cómo actualizar información en una base de datos?

alkemy

Data Manipulation Language (DML)

Data Manipulation Language (DML)

¿Qué es y para qué se utiliza?

Es una parte esencial del lenguaje SQL (Structured Query Language) que se utiliza para interactuar con los datos almacenados en una base de datos. Su **función principal es permitir la manipulación, inserción, actualización y eliminación de datos** dentro de las tablas de una base de datos.

Permite agregar nuevos datos, actualizar información existente y eliminar registros innecesarios. Esto asegura que la información almacenada en la base de datos esté siempre actualizada y sea coherente con la realidad que representa.

Data Manipulation Language (DML)

El DML se compone de varios comandos que permiten realizar acciones específicas sobre los datos, centrándose principalmente en la modificación de ellos y no tanto en la estructura de la Base de datos:

INSERT: Se utiliza para agregar nuevos registros (filas) a una tabla. Puedes especificar los valores que deseas insertar en cada columna o utilizar una subconsulta para insertar datos desde otra fuente.

```sql
/* 1) Insertar un registro puntual en la tabla Libros */
INSERT INTO Libros (id_libro, titulo, autor, año_publicacion, genero, id_editorial)
VALUES (101, 'Dune', 'Frank Herbert', 1965, 'Ciencia ficción', 4);

/* 2) Insertar varios registros de una sola vez en Editoriales */
INSERT INTO Editoriales (id_editorial, nombre_editorial, país) VALUES
(4, 'Ace Books',     'Estados Unidos'),
(5, 'Minotauro',     'Argentina');

/* 3) Insertar datos provenientes de otra tabla (subconsulta) */
INSERT INTO Libros (id_libro, titulo, autor, año_publicacion, genero, id_editorial)
SELECT id, titulo, autor, año_pub, genero, id_edit
FROM   CatalogoHistorico
WHERE  año_pub < 1950;
```

Data Manipulation Language (DML)

UPDATE: Permite modificar los valores de una o varias columnas en una o varias filas existentes de una tabla. Puedes utilizar una cláusula WHERE para especificar qué filas deben ser actualizadas.

```sql
-- 1) Cambiar el país de una editorial específica
UPDATE Editoriales
SET    país = 'Estados Unidos'
WHERE  id_editorial = 4;

-- 2) Actualizar el género de todos los libros publicados antes de 1950
UPDATE Libros
SET    genero = 'Clásico'
WHERE  año_publicacion < 1950;

-- 3) Incrementar en 1 el año de publicación de los libros de la editorial 5
UPDATE Libros
SET    año_publicacion = año_publicacion + 1
WHERE  id_editorial = 5;
```

Data Manipulation Language (DML)

DELETE: Se utiliza para eliminar filas de una tabla. Al igual que con UPDATE, puedes utilizar una cláusula WHERE para definir qué filas se eliminarán.

```sql
-- 1) Eliminar un libro puntual por su ID
DELETE FROM Libros
WHERE  id_libro = 101;

-- 2) Eliminar todos los libros publicados antes de 1900
DELETE FROM Libros
WHERE  año_publicacion < 1900;

-- 3) Eliminar una editorial que ya no existe
--    (y, por cascada, sus libros si definiste ON DELETE CASCADE)
DELETE FROM Editoriales
WHERE  id_editorial = 5;
```

Data Manipulation Language (DML)

WHERE: Es el comando más utilizado para consultar datos.

```sql
-- 1) Igualdad simple
SELECT titulo, autor
FROM   Libros
WHERE  genero = 'Ciencia ficción';

-- 2) Comparación numérica
SELECT nombre, salario
FROM   Empleados
WHERE  salario > 50000;

-- 3) Rango de fechas (BETWEEN)
SELECT *
FROM   Ventas
WHERE  fecha_venta BETWEEN '2024-01-01' AND '2024-03-31';

-- 4) Múltiples condiciones (AND / OR)
SELECT *
FROM   Productos
WHERE  precio < 100
  AND  stock > 0
  OR   categoria = 'Ofertas';

-- 5) Coincidencia de patrones (LIKE)
SELECT nombre_cliente, email
FROM   Clientes
WHERE  email LIKE '%@gmail.com';
```

Data Manipulation Language (DML)

SELECT: Aunque a menudo se asocia con la recuperación de datos, SELECT también es parte del DML.

Se utiliza para recuperar información de una o varias tablas de la base de datos.

```sql
-- 1) Recuperar todos los libros
SELECT *
FROM Libros;

-- 2) Seleccionar columnas específicas con alias
SELECT titulo   AS Título,
       autor    AS Autor,
       año_publicacion AS Año
FROM Libros;

-- 3) Libros publicados después de 2010
SELECT titulo, autor, año_publicacion
FROM Libros
WHERE año_publicacion > 2010;

-- 4) Libros con el nombre de su editorial (JOIN)
SELECT l.titulo,
       e.nombre_editorial
FROM   Libros       AS l
JOIN   Editoriales  AS e ON l.id_editorial = e.id_editorial;

-- 5) Cantidad de libros por género
SELECT genero, COUNT(*) AS total_libros
FROM   Libros
GROUP BY genero
ORDER BY total_libros DESC;
```

Aunque SELECT no modifica los datos en sí, es fundamental para la manipulación de datos al proporcionar los resultados que se pueden modificar utilizando INSERT, UPDATE y DELETE.

Data Manipulation Language (DML)

El **AutoCommit** es una característica de los sistemas de gestión de bases de datos que determina si las operaciones DML (Data Manipulation Language), como INSERT, UPDATE o DELETE, se confirman automáticamente después de ejecutarse o si requieren una confirmación manual mediante COMMIT.

Cuando el modo AutoCommit está activado, cada instrucción se guarda de inmediato en la base de datos, lo que evita la necesidad de confirmar los cambios manualmente, aunque también limita la posibilidad de revertirlos fácilmente.

```sql
/* Verificar el estado del AutoCommit (en MySQL) */
SELECT @@autocommit;

/* Desactivar AutoCommit para controlar manualmente la transacción */
SET autocommit = 0;

INSERT INTO Libros (id_libro, titulo, autor, año_publicacion)
VALUES (200, 'El Juego de Ender', 'Orson Scott Card', 1985);

/* Confirmar manualmente los cambios */
COMMIT;

/* En caso de error, revertir los cambios */
ROLLBACK;
```

Características principales:

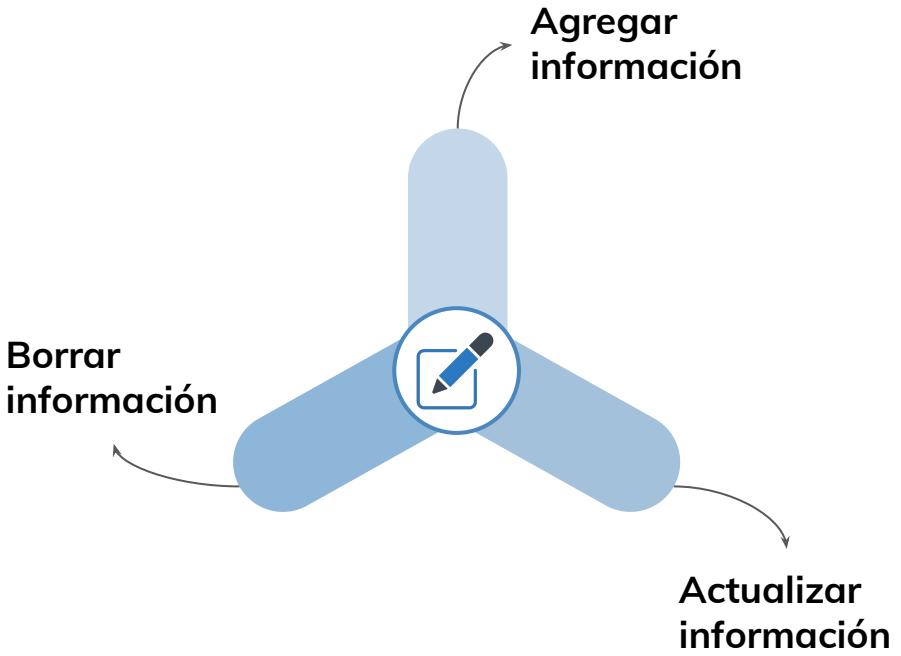
- Cuando **AutoCommit = ON**, cada instrucción DML se confirma automáticamente.
- Cuando **AutoCommit = OFF**, los cambios quedan pendientes hasta que el usuario ejecute un comando COMMIT (para guardar) o ROLLBACK (para deshacer).
- Es útil **desactivar el AutoCommit** en transacciones que involucren múltiples pasos dependientes entre sí.

Edición de información en una tabla ✍️

< > Edición de información en una tabla

En una base de datos, se pueden **editar datos** de varias formas utilizando operaciones clave: **Actualizar** (modificar información existente), **Borrar** (eliminar registros innecesarios) y **Agregar** (insertar nuevos registros).

Estas operaciones permiten gestionar y mantener los datos actualizados según las necesidades del sistema, asegurando que la base de datos refleje siempre la información más precisa y relevante.



➕Agregar información en una tabla

Agregar información en una tabla

Agregar información (INSERT INTO):

Para añadir nuevos registros a una tabla, se usa la instrucción **INSERT INTO**. Esto agrega **una nueva fila** con los valores que se proporcionen.

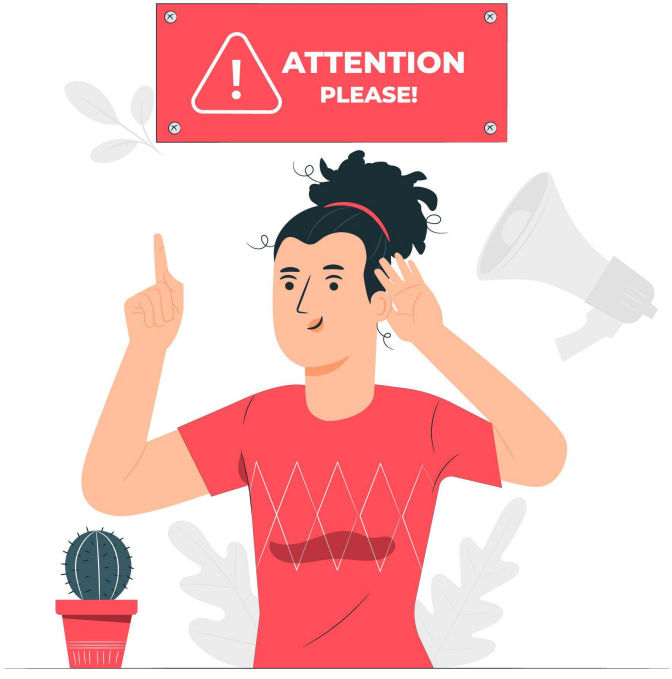
```
INSERT INTO nombre_de_tabla (columna1, columna2, columna3, ...)
VALUES (valor1, valor2, valor3, ...);
```

→ **nombre_de_tabla** es el nombre de la tabla a la que deseas insertar los datos.

→ **columna1, etc.** son los nombres de las columnas a las que deseas insertar datos.

→ **valor1, etc.** son los valores que deseas insertar en las respectivas columnas.

📌**NOTA:** Hay que tener en cuenta que si vas a insertar varias filas a la vez, puedes utilizar una lista de conjuntos de valores separados por comas en la cláusula VALUES.



Agregar información en una tabla

¿Cómo luce una sentencia INSERT en SQL?

Por ejemplo, si quieres agregar un nuevo producto a la tienda, puedes hacerlo así:

```
INSERT INTO productos (nombre, precio, cantidad)
VALUES ('Laptop HP', 900, 15);
```

Esta consulta agrega un nuevo producto, "Laptop HP", con su precio y cantidad.

Agregar información en una tabla

Veamos otro ejemplo:

Supongamos que tienes una tabla llamada "usuarios" con las columnas "id", "nombre" y "correo". Quieres agregar un nuevo usuario con ID 1, nombre "Juan" y correo "juan@example.com".

La sentencia INSERT sería:

```
INSERT INTO usuarios (nombre, correo)
VALUES ('Juan', 'juan@example.com');
```

Si la columna "id" es autoincremental puedes omitirla en la sentencia INSERT.

INSERT INTO y ID autogenerado e incremental

Cuando estamos agregando **nuevos registros** a una tabla mediante la instrucción **INSERT INTO**, generalmente necesitamos identificar de manera única cada registro con una columna que funcione como **clave primaria** (por ejemplo, un **ID**).

En muchas bases de datos, no queremos tener que asignar manualmente un valor único para cada nuevo registro. Para simplificar este proceso, se utiliza un **ID autogenerado** o **incremental**. Esto significa que, cada vez que insertamos un nuevo registro, el valor de la columna **ID** se genera automáticamente, aumentando de uno en uno, sin que el usuario tenga que preocuparse por hacerlo.

ID autogenerado e incremental

¿Cómo indicar que un ID debe ser autogenerado e incremental?

Para indicar que un ID debe ser autogenerado al realizar una operación de inserción (INSERT) en SQL, generalmente se utiliza una característica llamada **"autoincrement" o "identity"** dependiendo del sistema de gestión de bases de datos que estés utilizando.

Esta característica permite que la base de datos genere automáticamente un valor único para el ID cada vez que se inserta un nuevo registro.

No necesitas proporcionar un valor específico para la columna "id" *en la sentencia INSERT, ya que la base de datos manejará la generación automática.*

alkemy

Agregar información en una tabla

Ejemplo de cómo hacerlo ***MySQL***

En MySQL, puedes usar el atributo AUTO_INCREMENT en la definición de la columna para indicar que el ID debe ser autogenerado. La columna debe ser una clave primaria.

```
CREATE TABLE MiTabla (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50),
    ...
);

-- Al insertar, no es necesario proporcionar un valor para "id"
INSERT INTO MiTabla (nombre, ...)
VALUES ('Ejemplo', ...);
```

Agregar información en una tabla

Ejemplo de cómo hacerlo ***SQL Server***

En SQL Server, puedes usar la propiedad IDENTITY en la definición de la columna para lograr el mismo efecto.

```
CREATE TABLE MiTabla (
    id INT IDENTITY(1,1) PRIMARY KEY,
    nombre VARCHAR(50),
    ...
);

-- Al insertar, no es necesario proporcionar un valor para "id"
INSERT INTO MiTabla (nombre, ...)
VALUES ('Ejemplo', ...);
```

LIVE CODING

Ejemplo en vivo

Agregar nuevos usuarios utilizando INSERT:

En este espacio vamos a mostrar cómo agregar ***nuevos usuarios en una una tabla*** *llamada "Usuarios".*
No obstante antes de comenzar, debemos crear una conexión a una base de datos llamada "Alke Wallet".

1. ***Crear 3 tablas llamadas****:*

   → *Usuario, Transacción y Moneda.*

2. ***Crear la Tabla "Usuarios"****:*

   → *user_id (clave primaria)*
   → *nombre*
   → *correo electrónico*
   → *contraseña*
   → *saldo.*
   → *fecha de creación*

3. ***Crear la Tabla "Transaccion":***

   → *transaction_id (Primary Key)*
   → *sender_user_id (Foreign Key referenciando a User)*
   → *receiver_user_id (Foreign Key referenciando a User)*
   → *valor*
   → *transaction_date.*

Tiempo: 20 minutos

alkemy

LIVE CODING

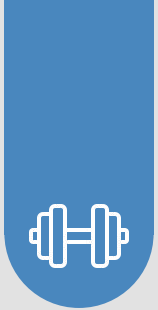
Ejemplo en vivo

Agregar nuevos usuarios utilizando INSERT:

En este espacio vamos a mostrar cómo agregar ***nuevos usuarios en una una tabla*** *llamada "Usuarios".*
No obstante antes de comenzar, debemos crear una conexión a una base de datos llamada "Alke Wallet".

4. *Crear la Tabla "Moneda":*
   ```
   currency_id (Primary Key)
   currency_name
   currency_symbol
   ```
5. *Insertar al menos 3 usuarios y 3 transacciones.*
6. *Consultar usuarios creados*

Tiempo: 20 minutos



Ejercicio N° 1

Aplicando el INSERT INTO

Aplicando el INSERT INTO

Contexto: 🙌

aprendimos sobre cómo insertar datos en una tabla usando la instrucción INSERT INTO y cómo gestionar la inserción de datos de manera más eficiente cuando se tiene un ID autogenerado e incremental. Ahora, es el momento de poner estos conceptos en práctica creando registros en una base de datos, mientras gestionamos de forma automática los valores del ID.

Este ejercicio les permitirá entender cómo agregar nuevos registros a las tablas de manera estructurada y sin tener que preocuparse por generar manualmente un identificador único.

Aplicando el INSERT INTO

Consigna: ✍️

El objetivo de este ejercicio es **crear una tabla** llamada `inventario` con las siguientes columnas: `ID`, `nombre_producto`, `precio` y `cantidad_disponible`.

Luego, deberán **insertar registros en la tabla** utilizando la instrucción **INSERT INTO**, asegurándose de que el campo **ID** se autogere y se incremente automáticamente. Esto les permitirá practicar cómo añadir productos al inventario sin tener que preocuparse por la gestión del **ID**.

Tiempo 🕐: *25 min.*

Aplicando el INSERT INTO

Paso a paso: ⚙️

1. **Abrir la herramienta SQL** :

   Usa un entorno de SQL Lite si lo tienes disponible. Asegúrate de elegir el tipo de base de datos que estás usando (por ejemplo, MySQL).

2. **Crear la tabla "inventario"** :

   Crea la tabla para gestionar el inventario con los siguientes campos:

```
CREATE TABLE inventario (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    nombre_producto VARCHAR(100),
    precio DECIMAL(10, 2),
    cantidad_disponible INT
);
```

Esto crea una tabla con 4 columnas: ***ID*** *(autogenerado e incremental),* `nombre_producto`, `precio` y `cantidad_disponible`.

alkemy

Aplicando el INSERT INTO

3. **Insertar un nuevo producto en el inventario:**

   Inserta un producto en la tabla de inventario. Como el ID es autogenerado, no necesitas asignarlo manualmente.

4. **Verificar la inserción del producto:**

   Ejecuta la siguiente consulta para ver cómo se ha insertado el producto en la tabla:

```
SELECT * FROM inventario;
```

→ *El valor de ID debería ser asignado automáticamente, por ejemplo, ID = 1.*

Aplicando el INSERT INTO

5. **Insertar más productos:**

   Inserta al menos dos productos adicionales utilizando el mismo formato. Los ID se autoincrementarán automáticamente con cada nuevo producto.

6. **Verificar los productos insertados:**

   Realiza otra consulta SELECT para ver cómo los productos se han añadido correctamente y cómo los ID han sido asignados e incrementados automáticamente:

```
SELECT * FROM inventario;
```

Aplicando el INSERT INTO

7. **Ejemplo adicional de inserción:**

   Inserta un nuevo producto con valores distintos para nombre_producto, precio y cantidad_disponible. Verifica cómo el sistema genera el siguiente ID de manera automática.

8. **Reflexión:**

   ¿Cómo te facilita esta característica el trabajo con grandes cantidades de productos sin tener que asignar manualmente los identificadores únicos?

Momento:

Time-out!

⏰5 -10 min.

Borrar información de una tabla

Borrar información en una tabla

¿Cómo borrar datos de una tabla?

Para borrar información de una tabla utilizando el Lenguaje de Manipulación de Datos (DML), puedes utilizar el comando **DELETE**. Este comando permite eliminar registros específicos de una tabla según una condición especificada.

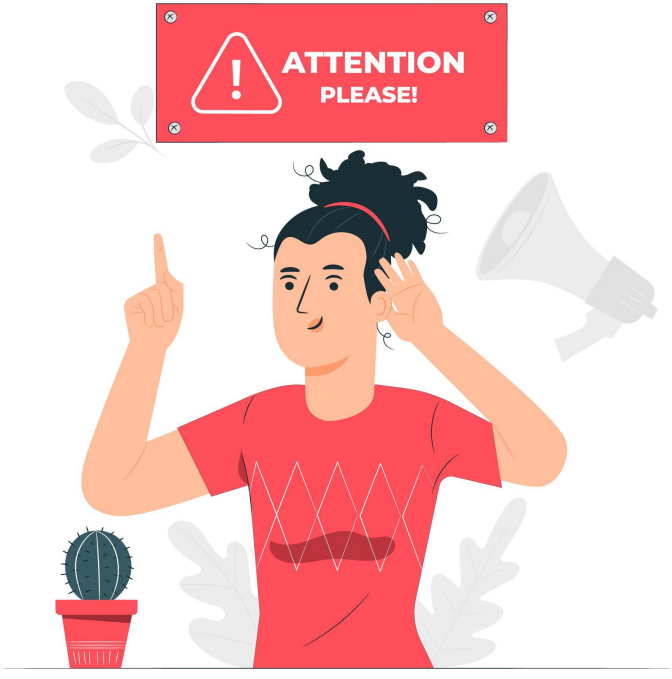

```
DELETE FROM productos
WHERE nombre = 'Laptop Dell';
```

Esta consulta eliminaría completamente el producto "Laptop Dell" de la tabla.

📌**NOTA:** ❗Asegúrate de tener una copia de seguridad de tus datos o de estar seguro de que los registros a eliminar son los correctos antes de ejecutar la sentencia DELETE ❗

→ **Condición WHERE:** Una condición opcional que especifica qué filas deben eliminarse. Si omite la cláusula WHERE, se eliminarán todas las filas de la tabla.

→ **La sentencia DELETE:** esta sentencia elimina toda la fila de la tabla. Si sólo desea eliminar los valores de columnas específicas pero conservar la fila, puede utilizar la sentencia UPDATE y establecer los valores en NULL o en un valor predeterminado adecuado.



Borrar información en una tabla

Aquí tienes la sintaxis básica:

```
DELETE FROM nombre_tabla
WHERE condicion;
```

En este ejemplo:

- **nombre_tabla** es el nombre de la tabla de la que deseas eliminar registros.
- **condición** es la condición que debe cumplir un registro para ser eliminado.

Borrar información en una tabla

Analicemos otro ejemplo:

Supongamos que tienes una tabla llamada "Clientes" y deseas borrar los registros de los clientes que no han realizado una compra en los últimos 12 meses.

```
DELETE FROM Clientes
WHERE fecha_ultima_compra IS NULL OR
fecha_ultima_compra< DATEADD(MONTH, -12, GETDATE());
```

En este ejemplo, la sentencia DELETE eliminará los registros de la tabla "Clientes" que no tienen una fecha de última compra registrada o cuya fecha de última compra sea anterior a 12 meses atrás desde la fecha actual.

Es importante tener cuidado al utilizar el comando DELETE, ya que eliminará permanentemente los registros de la tabla.

LIVE CODING

Ejemplo en vivo

Eliminación de registros utilizando el comando DELETE:

Eliminar registros de una tabla llamada "Usuarios" para borrar registros específicos según una condición dada:

1. Escribir una consulta para recuperar y mostrar todos los registros de la tabla "Usuarios" en la consola.
2. Utilizar el comando DELETE para eliminar los registros de los usuarios que fueron creados antes del 01/01/2020
3. Escribir una consulta final para recuperar y mostrar todos los registros restantes después de la eliminación.

Tiempo: 20 minutos

🔁Actualizar información en una tabla

Actualizar información en una tabla

¿Cómo actualizar información en una base de datos?

Para actualizar información en una tabla utilizando el Lenguaje de Manipulación de Datos (DML), puedes utilizar el comando **UPDATE**. Este comando te permite modificar los valores existentes en los registros de una tabla según una condición especificada.

📌**NOTA:**

Condición WHERE: Una condición opcional que especifica qué filas deben actualizarse. Si omite la cláusula WHERE, se actualizarán todas las filas de la tabla.

Actualizar información en una tabla

Aquí tienes la sintaxis básica:

```
UPDATE nombre_tabla
SET columna1 = valor1,
columna2 = valor2,
WHERE condicion;
```

- **nombre_tabla** es el nombre de la tabla en la que deseas realizar la actualización.
- **columna1,etc.** son las columnas que deseas actualizar.
- **valor1** son los nuevos valores que deseas asignar a las columnas.
- **condición** es la condición que debe cumplirse para determinar qué registros serán actualizados.

Actualizar información en una tabla

¡Veamos un ejemplo!

Supongamos que tienes una tabla llamada "Empleados" y deseas aumentar en un 10% el salario de todos los empleados que tienen más de 5 años de experiencia.

```
UPDATE Empleados
SET salario = salario * 1.10
WHERE años_experiencia > 5;
```

Analicemos:

En este ejemplo, la sentencia ***UPDATE*** *modifica el valor de la columna "salario" en la tabla "Empleados".*
Los empleados con más de 5 años de experiencia verán su salario aumentado en un 10%.

LIVE CODING

Ejemplo en vivo

Actualización de registros:

Actualizar de información en una tabla llamada "Transacciones". Utilizar el comando UPDATE para modificar los precios de las transacciones según ciertos criterios.

1. *Utilizar el comando UPDATE para aumentar en un 15% el precio de las transacciones que tienen un precio menor a $50.*
2. *Recuperar y mostrar todos los registros de la tabla "Transacciones" después de la actualización.*

Tiempo: 20 minutos

Ejercicio N° 2

INSERT + UPDATE + DELETE

Ejercicio integral: INSERT + UPDATE + DELETE

Contexto: 🙌

Hasta aquí ya practicamos:

- Creación de tablas y tipos de datos.
- Consultas simples, subconsultas y JOIN.
- Sentencias DML básicas.

Ahora cerraremos la clase con un workflow completo de manipulación de datos sobre la tabla Empleados, consolidando los tres comandos CRUD principales.

Consigna: ✍️

Trabaja sobre la tabla Empleados (id_empleado PK AUTO_INCREMENT, nombre, apellido, salario, fecha_ingreso, departamento).

Tiempo 🕐**:** *30 minutos*

Ejercicio integral: INSERT + UPDATE + DELETE

1. **Alta masiva — INSERT (10 min)**
    - Inserta cinco empleados con los datos de la tabla:
    - (Lucía Pérez 85 000 2024-02-01 IT, ..., Sofía Ruiz 68 000 2025-01-10 Marketing).
2. **Ajustes salariales y movimientos — UPDATE (12 min)**
    - Aumenta un 7 % a quienes ganen < 80 000.
    - Suma 5 000 fijos a los que tengan > 3 años de antigüedad.
    - Cambia a Sofía Ruiz al departamento Ventas.
3. **Depuración — DELETE (6 min)**
    - Elimina todos los empleados de RRHH.
4. **Verificación**
    - Tras cada bloque, ejecuta SELECT * FROM Empleados; para confirmar los cambios.

¿Alguna consulta?

alkemy

Resumen

¿Qué logramos en esta clase?

- ✓ ***Diferenciar los componentes principales de Data Manipulation Languaje (DML)***
- ✓ ***Insertar datos en una tabla***
- ✓ ***Crear ID autogenerados***
- ✓ ***Aprendimos a actualizar información de una tabla utilizando el comando UPDATE***
- ✓ ***Aprendimos a borrar información de una tabla utilizando el comando DELETE***

¡Ponte a prueba!

Momento de ejercitación

Te invitamos a aprovechar esta última sección del espacio sincrónico para realizar de manera individual las ***actividades disponibles en la plataforma.*** *Estas propuestas son clave para afianzar lo trabajado y* ***forman parte obligatoria del recorrido de aprendizaje.***

💪 Análisis de caso ——— 💪 Selección Múltiple

💪 Comprensión lectora

Si al resolverlas surge alguna duda, compártela o tráela al próximo encuentro sincrónico.

alkemy

#Checkout

¿Qué les pareció la clase de hoy?

3. Estuvo bien, pero podría mejorar
¿Qué se puede potenciar?

4. ¡Estuvo bien!
¿Qué me llevo?

2. No me convenció mucho
¿Qué no te gustó de la clase?

5. ¡Me encantó!
¿Qué fue lo mejor de la clase?

1. Red flag
¿Qué te hubiera gustado que fuera diferente?

¡Muchas gracias!

alkemy