GA6-220501096-AA1

P.8 Una vez diseñadas, modeladas e implementadas las bases de datos, es preciso darles el uso adecuado según el lenguaje estándar y las herramientas necesarias en el proceso de desarrollo.

Las siglas **SQL,** vienen del inglés **Structured Query Language,** es un lenguaje estándar que permite manipular los datos de una base de datos relacional. Casi todos los Sistemas de Gestión de Bases de Datos Relacionales (SGBD) implementan este lenguaje y mediante él se realizan todo tipo de actividad en la base de datos. En este punto se hace una presentación del **lenguaje SQL,** enfocándose en las sentencias de consulta de datos, empleando la interfaz de consola de línea de comandos con el fin de fortalecer las técnicas de administración remota. **Finalmente se presentará la interfaz de administración grafica para hacer las mismas operaciones.**

En nivel teórico, hay dos lenguajes para el uso de bases de datos:

DDL (Data Definition Language)

Es el lenguaje que se emplea para establecer bases de datos y tablas, y para transformar sus estructuras. Este lenguaje trabaja sobre ciertas tablas especiales llamadas diccionario de datos.

DML (Data Manipulation Language)

Es lo que se usa para cambiar y obtener datos desde las bases de datos.

SQL abarca ambos lenguajes **DDL+DML** y serán estudiados en esta unidad, ya que ambos hacen parte del conjunto de sentencias de **SQL.**

P.9 En esta actividad se describe el manejo de SQL enfocada en DDL y DML.

Te explico rápido para que tengas clara la estructura y puedas desarrollarla sin confusión.

1️⃣ Conceptos clave

DDL (Data Definition Language) → Lenguaje de Definición de Datos.

Sirve para crear, modificar o eliminar estructuras de la base de datos.

Ejemplos:

CREATE DATABASE → Crear una base de datos.

CREATE TABLE → Crear una tabla.

ALTER TABLE → Modificar una tabla.

DROP TABLE → Eliminar una tabla.

DML (Data Manipulation Language) → Lenguaje de Manipulación de Datos.

Sirve para insertar, modificar, eliminar o consultar los datos.

Ejemplos:

INSERT INTO → Insertar registros.

UPDATE → Actualizar registros.

DELETE → Eliminar registros.

SELECT → Consultar datos.

P.10 DDL – Data Definition Language

Definición: Son comandos que **definen** y **modifican** la estructura de la base de datos y sus objetos (tablas, vistas, índices, etc.).
⚠ No manipulan los datos, sino la estructura.

Comandos más comunes de DDL:

Comando	Función	Ejemplo
CREATE	Crear bases de datos u objetos (tablas, vistas, etc.)	CREATE DATABASE Inventario;
ALTER	Modificar la estructura de un objeto existente	ALTER TABLE Productos ADD stock INT;
DROP	Eliminar bases de datos u objetos	DROP TABLE Productos;
TRUNCATE	Eliminar **todos** los datos de una tabla, pero manteniendo su estructura	TRUNCATE TABLE Productos;

Ejemplo DDL:

1. Crear una base de datos

CREATE DATABASE Inventario;

- CREATE DATABASE: Crea una nueva base de datos en el servidor.
- Inventario: Es el nombre que le damos a la base de datos.
- Una vez ejecutada, el servidor tiene un espacio nuevo donde podremos crear tablas y otros objetos.

2. Seleccionar la base de datos para trabajar

USE Inventario;

- USE: Indica que todas las operaciones siguientes se realizarán sobre la base de datos especificada.
- Aquí estamos diciendo: "A partir de ahora, trabajaremos dentro de Inventario".

3. Crear una tabla

```
CREATE TABLE Productos (
  id INT PRIMARY KEY,
```

```
    nombre VARCHAR(100),
    precio DECIMAL(10,2)
);
```

- CREATE TABLE Productos: Crea una nueva tabla llamada Productos.
- id INT PRIMARY KEY: Columna llamada id, de tipo entero (INT), que será la clave primaria (PRIMARY KEY), es decir, no puede repetirse y no acepta valores nulos.
- nombre VARCHAR(100): Columna para texto variable, hasta 100 caracteres.
- precio DECIMAL(10,2): Número decimal con hasta 10 dígitos en total y 2 decimales (ideal para precios).

4. Agregar una columna a la tabla existente

```
ALTER TABLE Productos ADD stock INT;
```

- ALTER TABLE: Modifica la estructura de una tabla ya creada.
- ADD stock INT: Agrega una nueva columna llamada stock de tipo entero (INT), que guardará la cantidad disponible de cada producto.

5. Eliminar la tabla

```
DROP TABLE Productos;
```

- DROP TABLE: Elimina por completo una tabla, junto con todos sus datos y estructura.
- Productos: Es la tabla que vamos a borrar.
- Importante: Esta acción es irreversible, no se puede recuperar la información después de ejecutarla.

P.11 DML – Data Manipulation Language

Definición: Son comandos que **manipulan** los datos **dentro** de las tablas. ⚠ No afectan la estructura, solo el contenido.

Comandos más comunes de DML:

Comando	Función	Ejemplo
INSERT	Agregar nuevos registros	INSERT INTO Productos VALUES (1,'Pan',1500,20);
UPDATE	Modificar registros existentes	UPDATE Productos SET precio = 2000 WHERE id = 1;
DELETE	Eliminar registros	DELETE FROM Productos WHERE id = 1;
SELECT	Consultar datos	SELECT * FROM Productos;

Ejemplo DML:

1. Insertar datos

INSERT INTO Productos (id, nombre, precio, stock)

VALUES (1, 'Pan', 1500, 20);

- INSERT INTO Productos (...): Indica que vamos a agregar un nuevo registro (fila) a la tabla Productos.
- Columnas especificadas: id, nombre, precio, stock.
- VALUES (...): Contiene los valores que se insertarán en las columnas, en el mismo orden indicado:
- 1 → valor de la columna id.
- 'Pan' → valor de la columna nombre (entre comillas simples porque es texto).
- 1500 → valor de la columna precio.
- 20 → valor de la columna stock.
- Resultado: La tabla Productos ahora tendrá un producto con ID 1, nombre "Pan", precio 1500 y 20 unidades en stock.

2. Actualizar datos

UPDATE Productos SET precio = 2000 WHERE id = 1;

- UPDATE Productos: Indica que vamos a modificar datos en la tabla Productos.
- SET precio = 2000: Cambia el valor de la columna precio a 2000.
- WHERE id = 1: Condición para actualizar únicamente el registro cuyo id sea igual a 1.
- Resultado: El producto con ID 1 ahora tendrá un precio actualizado de 2000.

3. Consultar datos

SELECT * FROM Productos;

- SELECT: Sentencia para consultar (leer) datos de la base de datos.
- *: Significa "todas las columnas".
- FROM Productos: Indica que la consulta es sobre la tabla Productos.
- Resultado: Muestra todas las filas y columnas que existan en la tabla.

4. Eliminar un registro

DELETE FROM Productos WHERE id = 1;

- DELETE FROM Productos: Indica que vamos a eliminar uno o varios registros de la tabla Productos.
- WHERE id = 1: Condición para borrar únicamente el producto cuyo id sea igual a 1.
- Importante: Si se omite el WHERE, se eliminarían todos los registros de la tabla.
- Resultado: El producto con ID 1 desaparece de la tabla.

📌 **Resumen rápido:**

- **DDL** = "Arquitecto" → diseña y cambia la estructura.

- **DML** = “Operario” → trabaja con los datos que hay dentro.

P.15 📄 Informe Técnico – Taller SQL

Portada

Servicio Nacional de Aprendizaje – SENA Centro de Hotelería y turismo
Ficha: 3070301
Programa: Tecnólogo en Análisis y Desarrollo de Software
Aprendiz: (Nombre del Aprendiz)
Instructor: *(*Juan Manuel Aldana Zambrano*)*
Fecha: 20 de agosto de 2025

Título del Documento:
Informe Técnico – Taller SQL

Introducción

El presente informe describe la solución al taller “Taller SQL”, el cual tiene como objetivo practicar la creación, modificación, inserción y consulta de datos en una base de datos relacional utilizando sentencias SQL. Se trabajará con la tabla denominada **libreta**, que contendrá información básica de personas incluyendo nombre, domicilio y teléfono.

Se emplearán comandos **DDL (Data Definition Language)** para la definición de la estructura de la tabla y comandos **DML (Data Manipulation Language)** para la manipulación de datos.

Objetivo

Aplicar sentencias SQL de definición y manipulación de datos para:

1. Crear una tabla con campos específicos.
2. Visualizar las tablas existentes y su estructura.
3. Insertar, actualizar y consultar registros.
4. Contar el total de registros almacenados.

Desarrollo y Sentencias SQL

1. Creación de la tabla libreta (DDL):

```
CREATE TABLE libreta (
  nombre VARCHAR(20),
```

```
    domicilio VARCHAR(30),
    telefono VARCHAR(11)
);
```

Descripción del Código:

1**. CREATE TABLE libreta**

CREATE TABLE: Es una sentencia DDL (Data Definition Language) que sirve para crear una nueva tabla dentro de la base de datos activa.

libreta: Es el nombre de la tabla.

En este caso representa una agenda o libreta de contactos donde guardaremos información personal.

2. Definición de columnas

Entre los paréntesis se definen los campos (columnas) que tendrá la tabla, junto con su tipo de dato.

🔷 **nombre VARCHAR(20)**

nombre: Columna para almacenar el nombre de la persona.

VARCHAR(20): Tipo de dato de texto de longitud variable, con un límite máximo de 20 caracteres.

Esto ahorra espacio en comparación con CHAR, ya que solo usa el espacio necesario.

🔷 **domicilio VARCHAR(30)**

domicilio: Columna para guardar la dirección o lugar de residencia.

VARCHAR(30): Texto de longitud variable, hasta un máximo de 30 caracteres.

🔷 **telefono VARCHAR(11)**

telefono: Columna para almacenar el número de teléfono.

VARCHAR(11): Texto de longitud variable de hasta 11 caracteres.

Aunque sea un dato numérico, se usa VARCHAR para permitir formatos con guiones, espacios o código de país ("3015551234", "57-3015551234", etc.).

2. Visualizar las tablas existentes:

```
SHOW TABLES;  -- Muestra todas las tablas
```

3. Visualizar la estructura de la tabla libreta:

```
DESCRIBE libreta;
```

P.16 4. Insertar registros iniciales (DML):

INSERT INTO libreta (nombre, domicilio, telefono)

VALUES ('Alberto Mores', 'Colon 123', '4234567'),

('Juan Torres', 'Avellaneda 135', '4458787');

Descripcion:

1. INSERT INTO libreta (...)

INSERT INTO: Sentencia DML (Data Manipulation Language) usada para agregar registros a una tabla.

libreta: Es el nombre de la tabla donde se insertarán los datos.

Los nombres entre paréntesis (nombre, domicilio, telefono) indican las columnas específicas a las que vamos a asignar valores.

Esto es útil para:

Insertar en un orden específico.

No insertar en todas las columnas si no es necesario.

2. VALUES (...)

VALUES: Indica que a continuación se especifican los valores que se insertarán en las columnas listadas.

Cada paréntesis representa una fila (registro) que se insertará.

En este caso se están insertando dos registros en una sola sentencia.

3. Detalle de los registros

Primer registro

'Alberto Mores' → valor para la columna nombre (texto, por eso va entre comillas simples).

'Colon 123' → valor para domicilio.

'4234567' → valor para telefono. Aunque es numérico, se almacena como texto (VARCHAR) para permitir guiones o prefijos.

Segundo registro

'Juan Torres' → nombre.

'Avellaneda 135' → domicilio.

'4458787' → telefono.

4. Resultado

Después de ejecutar la sentencia, la tabla libreta contendrá:

nombre	domicilio	telefono
Alberto Mores	Colon 123	4234567
Juan Torres	Avellaneda 135	4458787

5. Mostrar todos los registros de la tabla Libreta:

SELECT * FROM libreta;

6. Actualizar datos insertados:

UPDATE libreta

SET domicilio = 'Colon 456', telefono = '4234500'

WHERE nombre = 'Alberto Mores';

UPDATE libreta

SET domicilio = 'Avellaneda 200', telefono = '4458700'

WHERE nombre = 'Juan Torres';

Descripción 6:

1. **UPDATE libreta**

UPDATE: Comando DML (Data Manipulation Language) que sirve para modificar registros existentes en una tabla.

libreta: Nombre de la tabla en la que se realizará el cambio.

2. SET

SET: Indica los nuevos valores que se asignarán a las columnas.

domicilio = 'Colon 456' → actualiza la dirección del registro.

telefono = '4234500' → actualiza el número de teléfono.

3. WHERE

WHERE nombre = 'Alberto Mores': Filtra para que la actualización solo afecte al registro cuyo nombre sea exactamente “Alberto Mores”.

Importante: Si no se coloca WHERE, el cambio afectará a todos los registros de la tabla.

Sentencia 2

UPDATE libreta

SET domicilio = 'Avellaneda 200', telefono = '4458700'

WHERE nombre = 'Juan Torres';

1. Estructura

Mismo funcionamiento que la sentencia anterior, pero ahora afecta a un registro distinto.

2. Cambios

domicilio = 'Avellaneda 200' → cambia la dirección.

telefono = '4458700' → cambia el número de teléfono.

3. Filtro

WHERE nombre = 'Juan Torres' → solo actualiza la fila de Juan Torres.

P17 7. Insertar 5 registros adicionales:

INSERT INTO libreta (nombre, domicilio, telefono)

VALUES ('Maria Lopez', 'San Martin 45', '4211111'),

('Pedro Gomez', 'Mitre 123', '4222222'),

('Luis Perez', 'Belgrano 789', '4233333'),

('Ana Torres', 'Rivadavia 321', '4244444'),

('Carlos Diaz', 'Sarmiento 654', '4255555');

Descripcion 7.

1. INSERT INTO libreta (...)

INSERT INTO → Comando DML (Data Manipulation Language) que agrega nuevos registros a una tabla existente.

libreta → Nombre de la tabla en la que se insertarán los datos.

(nombre, domicilio, telefono) → Lista de columnas a las que se asignarán los valores.

El orden aquí debe coincidir con el orden de los valores que se ingresan después.

2. VALUES (...)

VALUES → Palabra clave que indica que a continuación se especifican los datos a insertar.

Cada conjunto de valores entre paréntesis representa una fila nueva que se agregará a la tabla.

3. Registros que se insertarán

('Maria Lopez', 'San Martin 45', '4211111')

- nombre: Maria Lopez

- domicilio: San Martin 45
- telefono: 4211111

('Pedro Gomez', 'Mitre 123', '4222222')

- nombre: Pedro Gomez
- domicilio: Mitre 123
- telefono: 4222222

('Luis Perez', 'Belgrano 789', '4233333')

- nombre: Luis Perez
- domicilio: Belgrano 789
- telefono: 4233333

('Ana Torres', 'Rivadavia 321', '4244444')

- nombre: Ana Torres
- domicilio: Rivadavia 321
- telefono: 4244444

('Carlos Diaz', 'Sarmiento 654', '4255555')

- nombre: Carlos Diaz
- domicilio: Sarmiento 654
- telefono: 4255555

8. Contar total de registros ingresados:

SELECT COUNT(*) AS total_registros FROM libreta;

Descripcion 8:

1. SELECT

SELECT → Comando SQL usado para consultar datos de una base de datos.

En lugar de traer todos los registros, en este caso se pide un cálculo usando una función de agregación.

2. COUNT(*)

COUNT → Función de agregación que cuenta el número de filas en el resultado de una consulta.

(*) → Significa “contar todas las filas”, sin importar su contenido ni valores nulos.

Ejemplo: Si la tabla tiene 10 registros, COUNT(*) devolverá 10.

3. AS total_registros

AS → Sirve para asignar un alias (nombre temporal) a una columna en el resultado.

total_registros → Es el alias que aparecerá como nombre de columna en el resultado, en lugar de COUNT(*).

4. FROM libreta

Indica que la consulta se ejecutará sobre la tabla libreta.

Es donde se buscarán las filas para contarlas.

Conclusiones

A través de este ejercicio se reforzaron los conceptos y la sintaxis básica de SQL para la creación y manipulación de datos en una base de datos relacional. Se aplicaron comandos **DDL** para la definición de la estructura de la tabla y **DML** para la gestión de los datos.

P.21 Creación de la estructura de la BD y aplicación de restricciones.

La creación de la estructura de la base de datos y aplicación de restricciones es el proceso de definir, organizar y proteger cómo se van a almacenar los datos en un sistema de gestión de bases de datos (SGBD), usando principalmente sentencias DDL (Data Definition Language) como CREATE, ALTER y DROP.

Te lo explico en dos partes:

1️⃣ Creación de la estructura de la BD

Es la fase en la que:

- Se crea la base de datos (CREATE DATABASE).
- Se definen las tablas que contendrán los datos (CREATE TABLE).
- Se especifican los campos (columnas) de cada tabla, indicando:
- Nombre del campo (ej. nombre, telefono).
- Tipo de dato (ej. VARCHAR, INT, DATE).
- Tamaño máximo (ej. VARCHAR(50) → hasta 50 caracteres).

📌 Ejemplo básico:

```
CREATE DATABASE Agenda;
USE Agenda;
CREATE TABLE Contactos (
   id INT,
   nombre VARCHAR(50),
   telefono VARCHAR(15)
);
```

Esto ya es una **estructura básica**, pero todavía no tiene controles de calidad.

P.22 2️⃣ Aplicación de restricciones

Son reglas que se añaden a la estructura para garantizar la integridad y validez de los datos.

En SQL se definen al crear o modificar una tabla, y las más comunes son:

- PRIMARY KEY → Identifica de forma única cada registro.
- FOREIGN KEY → Relaciona datos entre tablas distintas.
- NOT NULL → Obliga a que un campo no esté vacío.
- UNIQUE → Impide que se repitan valores en una columna.
- DEFAULT → Asigna un valor por defecto si no se especifica.
- CHECK → Valida que los datos cumplan una condición.
- AUTO_INCREMENT → Genera valores numéricos automáticos.

📌 Ejemplo con restricciones:

```
CREATE TABLE Contactos (
  id INT PRIMARY KEY AUTO_INCREMENT,
  nombre VARCHAR(50) NOT NULL,
  telefono VARCHAR(15) NOT NULL UNIQUE
);
```

Aquí:

- PRIMARY KEY AUTO_INCREMENT → Crea un identificador único que crece automáticamente.
- NOT NULL → Evita valores vacíos.
- UNIQUE → Garantiza que no haya teléfonos repetidos.

✅ En resumen:

Creación de la estructura → Diseñar tablas, columnas y tipos de datos.

Aplicación de restricciones → Poner reglas para asegurar que los datos sean correctos, completos y coherentes.

P.24 Tomando como referencia mi **proyecto formativo** del restaurante **"Sabor & Hogar"** (gestión de inventarios, proveedores y compras)

📄 **Documento: Estructura de la Base de Datos – Proyecto Formativo "Sabor & Hogar"**

Portada

Servicio Nacional de Aprendizaje – SENA
Ficha: 3070301

Programa: Tecnólogo en Análisis y Desarrollo de Software
Aprendiz: Juan Manuel Aldana Zambrano
Instructor: *(Nombre del instructor)*
Fecha: 20 de agosto de 2025

Título del Documento:
Estructura de la Base de Datos – Proyecto Formativo "Sabor de Hogar"

Introducción

El presente documento describe la estructura de la base de datos diseñada para el software de gestión de inventarios, proveedores y compras del restaurante **Sabor & Hogar**. El sistema tiene como propósito optimizar la administración de los productos, proveedores y pedidos, permitiendo un control eficiente de existencias y transacciones.

La base de datos ha sido diseñada bajo principios de normalización y buenas prácticas, utilizando MySQL como gestor de base de datos y aplicando restricciones de integridad referencial mediante llaves primarias y foráneas.

Objetivo

Diseñar y documentar la estructura de la base de datos del proyecto formativo "Sabor & Hogar", definiendo atributos, tipos de datos y restricciones, con el fin de garantizar un modelo relacional funcional, escalable y seguro.

Estructura de la Base de Datos

Tablas y atributos

1. Tabla proveedores

Campo	Tipo de dato	Restricciones
id_proveedor	INT	PK, AUTO_INCREMENT, NOT NULL
nombre	VARCHAR(100)	NOT NULL
telefono	VARCHAR(15)	NULL
direccion	VARCHAR(150)	NULL
Email	VARCHAR(100)	UNIQUE

Explicación

- id_proveedor → Identificador único del proveedor. Crece automáticamente y no puede ser nulo.
- nombre → Nombre del proveedor, obligatorio, máximo 100 caracteres.
- telefono → Número de contacto, opcional.
- direccion → Dirección física del proveedor, opcional.
- email → Correo electrónico único para cada proveedor, evita duplicados en la base de datos.

2. Tabla productos

Campo	**Tipo de dato**	**Restricciones**
id_producto	INT	PK, AUTO_INCREMENT, NOT NULL
nombre	VARCHAR(100)	NOT NULL
descripcion	TEXT	NULL
precio_unitario	DECIMAL(10,2)	NOT NULL
Stock	INT	DEFAULT 0
id_proveedor	INT	FK → proveedores(id_proveedor)

Explicación

- id_producto: Identificador único del producto. Crece automáticamente y no puede ser nulo.
- nombre: Nombre del producto, obligatorio, máximo 100 caracteres.
- descripcion: Texto libre para describir el producto, opcional.
- precio_unitario: Precio de venta o compra del producto, con dos decimales, obligatorio.
- stock: Cantidad disponible en inventario. Si no se especifica, se guarda 0.
- id_proveedor: Identificador del proveedor. Es una clave foránea que debe existir en la tabla proveedores.

3. Tabla compras

Campo	**Tipo de dato**	**Restricciones**
id_compra	INT	PK, AUTO_INCREMENT, NOT NULL
Fecha	DATE	NOT NULL

Campo	Tipo de dato	Restricciones
Total	DECIMAL(10,2)	NOT NULL
id_proveedor	INT	FK → proveedores(id_proveedor)

Explicación

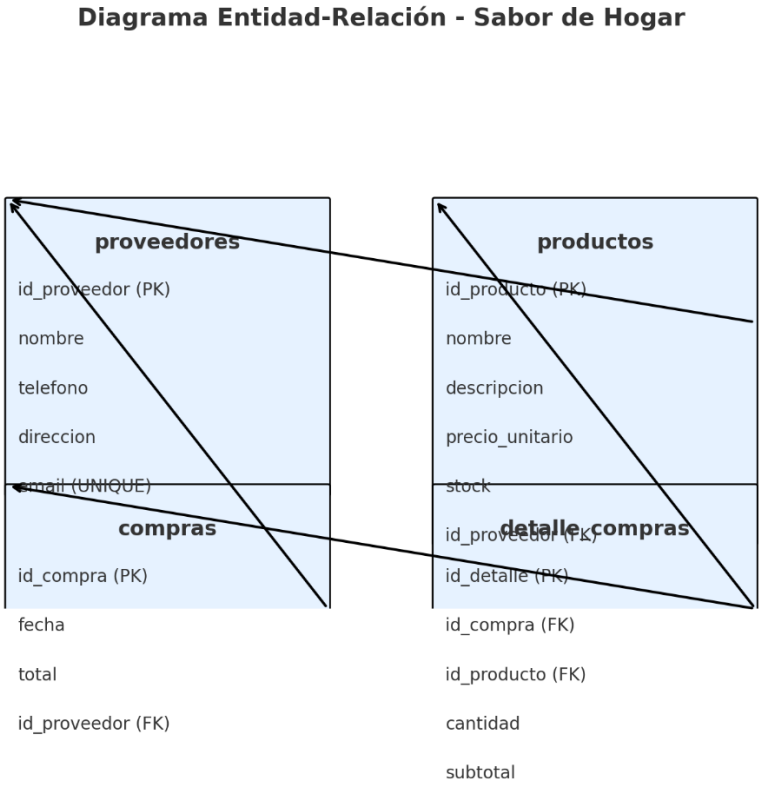
- **id_compra** → Identificador único de la compra. Crece automáticamente y no puede ser nulo.
- **fecha** → Fecha en la que se realiza la compra, obligatoria.
- **total** → Monto total de la compra con dos decimales, obligatorio.
- **id_proveedor** → Identificador del proveedor al que se le compra. Debe existir previamente en proveedores.

4. Tabla detalle_compras

Campo	Tipo de dato	Restricciones
id_detalle	INT	PK, AUTO_INCREMENT, NOT NULL
id_compra	INT	FK → compras(id_compra)
id_producto	INT	FK → productos(id_producto)
cantidad	INT	NOT NULL
subtotal	DECIMAL(10,2)	NOT NULL

- **id_detalle** → Identificador único del registro de detalle, autoincremental y obligatorio.
- **id_compra** → Relaciona el detalle con la compra principal en compras. Obligatorio.
- **id_producto** → Relaciona el detalle con un producto de la tabla productos. Obligatorio.
- **cantidad** → Número de unidades adquiridas, no puede ser nulo.
- **subtotal** → Resultado de cantidad × precio_unitario para ese producto en esa compra, obligatorio.

el **diagrama ER** para las tablas del proyecto *Sabor de Hogar*, mostrando las entidades, atributos y relaciones entre proveedores, productos, compras y detalle_compras.



P30. Script SQL (MySQL Workbench)

```
CREATE DATABASE SaborDeHogar;

USE SaborDeHogar;


-- Tabla Proveedores

CREATE TABLE proveedores (

   id_proveedor INT PRIMARY KEY AUTO_INCREMENT,

   nombre VARCHAR(100) NOT NULL,

   telefono VARCHAR(15),

   direccion VARCHAR(150),

   email VARCHAR(100) UNIQUE

);
```

```
-- Tabla Productos
CREATE TABLE productos (
    id_producto INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    precio_unitario DECIMAL(10,2) NOT NULL,
    stock INT DEFAULT 0,
    id_proveedor INT,
    FOREIGN KEY (id_proveedor) REFERENCES proveedores(id_proveedor)
);

-- Tabla Compras
CREATE TABLE compras (
    id_compra INT PRIMARY KEY AUTO_INCREMENT,
    fecha DATE NOT NULL,
    total DECIMAL(10,2) NOT NULL,
    id_proveedor INT,
    FOREIGN KEY (id_proveedor) REFERENCES proveedores(id_proveedor)
);

-- Tabla Detalle de Compras
CREATE TABLE detalle_compras (
    id_detalle INT PRIMARY KEY AUTO_INCREMENT,
    id_compra INT,
    id_producto INT,
    cantidad INT NOT NULL,
    subtotal DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (id_compra) REFERENCES compras(id_compra),
    FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
```

);

📌 **Notas técnicas**:

- Se aplicaron **llaves primarias** (PRIMARY KEY) para identificar de forma única cada registro.
- Se definieron **llaves foráneas** (FOREIGN KEY) para mantener la integridad referencial entre tablas.
- Se agregaron restricciones como NOT NULL, DEFAULT y UNIQUE para asegurar la consistencia de datos.

¡Domina las Sentencias DDL en SQL en 10 Minutos!

https://www.youtube.com/watch?v=B5i-_dSHQPA

CREAR BASE DE DATOS Y TABLAS EN MY SQL WORKBENCH

https://www.youtube.com/watch?v=g8c0wvsBJ64

Cómo Crear una Base de Datos y Tablas con MySQL Workbench - Paso a Paso

https://www.youtube.com/watch?v=gTrq_r8Kb6Y