Documento SQL para la gestión de la tabla "Productos"

1. Creación de la Tabla "Productos"

Para gestionar datos estructurados como el inventario de productos, es esencial crear tablas bien definidas en una base de datos. Esta tarea se realiza con **sentencias DDL (Data Definition Language)**, que permiten crear y modificar la estructura de objetos en la base de datos (como tablas, índices o vistas). La sentencia utilizada en este caso es **CREATE TABLE**, que crea una nueva tabla con los campos especificados.

Código:

```
CREATE TABLE Productos (
    idproducto INT PRIMARY KEY,
    descripcion VARCHAR(255) NOT NULL,
    cantidad INT NOT NULL,
    precio DECIMAL(10, 2) NOT NULL
);
```

Explicación de los campos:

1. **idproducto INT PRIMARY KEY**:
   - **Tipo de Dato (INT)**: Se elige **INT** (entero) para el campo **idproducto** porque los identificadores únicos de productos suelen ser números enteros, ya que se pueden manejar de manera eficiente en la base de datos.
   - **Clave Primaria (PRIMARY KEY)**: Declaramos este campo como clave primaria. Esto significa que:
     No puede haber dos productos con el mismo **idproducto**, garantizando así la unicidad de cada entrada en la tabla.
     No puede contener valores **NULL**, ya que todos los registros deben tener un identificador único.
   - **¿Por qué usamos una clave primaria?**: El objetivo de una clave primaria es asegurar que cada fila (producto) en la tabla pueda identificarse de manera única. Además, muchas bases de datos utilizan índices en las claves primarias para mejorar la velocidad de búsqueda y recuperación de datos.

2. **descripcion VARCHAR(255) NOT NULL**:
   - **Tipo de Dato (VARCHAR)**: **VARCHAR** (variable character) permite almacenar texto de longitud variable. En este caso, se establece un límite de 255 caracteres.

- **Restricción NOT NULL**: Esto indica que la descripción no puede estar vacía; cada producto debe tener una descripción.
- **¿Por qué usar VARCHAR en lugar de CHAR?**: **VARCHAR** es una mejor opción cuando las longitudes de texto varían mucho, ya que solo ocupa el espacio necesario para cada entrada. **CHAR** reservaría siempre el máximo de espacio (255 caracteres en este caso) y, por lo tanto, podría ocupar más espacio innecesario en disco.

3. **cantidad INT NOT NULL**:
   - **Tipo de Dato (INT)**: Este campo también es un número entero porque suele ser más eficiente para almacenar cantidades que otros tipos numéricos.
   - **Restricción NOT NULL**: Se aplica esta restricción para que la cantidad nunca esté vacía, es decir, cada producto debe tener una cantidad específica en inventario (aunque sea 0 para indicar que está agotado).

4. **precio DECIMAL(10, 2) NOT NULL**:
   - **Tipo de Dato (DECIMAL(10, 2))**: Usamos **DECIMAL** para almacenar valores monetarios. En este caso, **DECIMAL(10, 2)** especifica:
     - **10:** El número total de dígitos (incluyendo decimales y enteros).
     - **2:** El número de dígitos decimales.
   - Esto permite almacenar precios con hasta 8 dígitos en la parte entera y 2 en la parte decimal, lo cual es ideal para valores monetarios que requieren precisión.
   - **Restricción NOT NULL**: Al igual que en los otros campos, esta restricción asegura que siempre se almacene un valor en el campo de precio.

¿Por qué DECIMAL para el precio en lugar de FLOAT o DOUBLE?

FLOAT o **DOUBLE** podrían introducir imprecisiones, especialmente en cálculos financieros, debido a su naturaleza de punto flotante. **DECIMAL** es preferible para cantidades que requieren precisión exacta, como precios, ya que se almacena en una representación exacta sin errores de redondeo.

2. Inserción de Datos en la Tabla "Productos"

Una vez creada la tabla, pasamos a la **sentencia DML (Data Manipulation Language)**, que en este caso es la sentencia INSERT INTO. Esto permite añadir registros a la tabla con valores específicos para cada campo.

Código:

```
INSERT INTO Productos (idproducto, descripcion, cantidad, precio)
VALUES (1, 'Mesa de comedor', 10, 199.99);
```

Explicación de cada elemento de la sentencia:

1. **INSERT INTO Productos**:
   - La sentencia comienza indicando en qué tabla se insertará el registro, en este caso, la tabla **Productos**.
2. **Especificación de campos**:
   - **(idproducto, descripcion, cantidad, precio):** Se especifican los nombres de los campos en los que queremos insertar valores. Esto es útil para que la base de datos asigne correctamente cada valor al campo adecuado.
   - Si omitiéramos esta lista y simplemente escribiéramos INSERT INTO Productos VALUES (...), la base de datos asumiría que estamos insertando un valor en cada columna, en el orden en que están definidas en la tabla. Es buena práctica especificar los nombres de los campos para evitar errores en caso de que cambie el orden de las columnas en la tabla.
3. **VALUES (1, 'Mesa de comedor', 10, 199.99)**:
   - **1**: Se asigna el valor **1** a **idproducto**. Este valor debe ser único porque es la clave primaria.
   - **'Mesa de comedor'**: Se asigna esta descripción al campo **descripcion**.
   - **10**: Indica que tenemos **10** unidades en inventario.
   - **199.99**: Define el precio del producto. Aquí estamos usando dos decimales, aunque podrían especificarse más si fueran necesarios en otros casos.

Comentarios adicionales

1. **Mantenimiento de la integridad de los datos**:
   - Las restricciones **NOT NULL** y **PRIMARY KEY** ayudan a asegurar la integridad de los datos, garantizando que no se ingresen registros incompletos o duplicados.
   - Esto es especialmente importante en sistemas de inventario y gestión de datos empresariales, donde los errores en los registros pueden afectar la disponibilidad y precisión de la información.

2. **Escalabilidad**:
   - Esta estructura es sencilla y puede ampliarse para añadir más campos en caso de ser necesario, como **categoria, proveedor_id, o fecha_de_creacion**. La base de datos se puede diseñar de forma modular, con tablas adicionales para gestionar información relacionada, como proveedores o categorías de productos, lo que facilita la escalabilidad.
3. **Ejemplo de ampliación de la tabla**:
   - Si más adelante se requiere información adicional, podríamos utilizar la sentencia **ALTER TABLE** para añadir nuevos campos o modificar los existentes:

Código:

```
ALTER TABLE Productos
ADD COLUMN categoria VARCHAR(50);
```

Ejemplo práctico:

1- Crear base de datos y seleccionar:

```
mysql> CREATE DATABASE IF NOT EXISTS TiendaMuebles;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> USE TiendaMuebles;
Database changed
```

2- Crear la tabla Productos:

```
mysql> CREATE TABLE Productos (
    ->     idproducto INT PRIMARY KEY AUTO_INCREMENT,
    ->     descripcion VARCHAR(255) NOT NULL,
    ->     cantidad INT NOT NULL,
    ->     precio DECIMAL(10, 2) NOT NULL
    -> );
Query OK, 0 rows affected (0.05 sec)
```

3- **Insertar un registro**:

```
mysql> INSERT INTO Productos (descripcion, cantidad, precio)
    -> VALUES ('Mesa de comedor', 10, 199.99);
Query OK, 1 row affected (0.01 sec)
```

4-**Verificar el contenido de la tabla**:

```
mysql> SELECT * FROM Productos;
+------------+-----------------+----------+--------+
| idproducto | descripcion     | cantidad | precio |
+------------+-----------------+----------+--------+
|          1 | Mesa de comedor |       10 | 199.99 |
+------------+-----------------+----------+--------+
1 row in set (0.00 sec)
```