Consultas SQL

Daniel Felipe Mora Rosas

Instituto Tecnológico del Putumayo

Octavo Semestre

Lic. Brayan Arcos

19 de septiembre

2024

Tabla de contenido

Resumen Ejecutivo 3

Introducción Al Informe 4

Contexto y Motivación 4

Alcance 5

Objetivo 6

Metodología 7

Herramientas Utilizadas 7

Procedimientos 8

Desarrollo 10

Descripción de la Base de Datos 10

Tablas y Llaves 10

Relaciones 11

Consultas SQL 12

Diseño de la Base de Datos 15

Análisis 18

Conclusión 19

Referencias 20

Resumen Ejecutivo

Abordamos la implementación y gestión de bases de datos relacionales utilizando MySQL, enfocándonos en lo práctico con la creación y manejo de tablas, relaciones, y la ejecución de consultas. A lo largo del documento, se presenta un caso de estudio basado en una base de datos de ventas de productos, donde se aplican conceptos clave de SQL como DML, DDL, tipos de datos, funciones y operadores SQL. El objetivo principal es mostrar cómo diseñar, poblar (llenar la BD) y consultar una base de datos de manera eficiente y segura, utilizando transacciones y técnicas de optimización para asegurar la integridad de los datos.

Introducción Al Informe

Contexto y Motivación

Se tiene como propósito demostrar la aplicación de SQL en la creación de una base de datos relacional, enfocándonos en el contexto de un sistema de ventas (“básico”). Se destacan una operación de actualización de stock sobre los productos y el uso de subconsultas para resolver problemas comunes en la gestión de datos. La importancia del tema radica en la necesidad de entender cómo estructurar una base de datos.

Alcance

El alcance de esta base de datos se centra en la gestión integral de un sistema de ventas, proporcionando una estructura eficaz para el manejo de productos, categorías, clientes y facturación. Este diseño es escalar, lo que significa que permite adaptarse a las necesidades de un negocio en crecimiento, mejorando la eficiencia operativa y el análisis de datos. Algunos puntos a tener en cuenta son:

Diseño de Bases de Datos: Consta de tablas con relaciones de uno a muchos y muchos a muchos, asegurando la correcta integración y consistencia de los datos.

Gestión de Transacciones: Por medio de una consulta nos podemos mantener atentos al stock de los productos para así asegurar que los cambios en los datos se realicen de manera controlada.

Tipos de Datos y Funciones SQL: Aplicación adecuada de tipos de datos y funciones que permiten manipular y analizar la información de manera efectiva.

Subconsultas y Operadores SQL: Se implementaron subconsultas y operadores que facilitan la creación de consultas complejas y la solución de problemas comunes en la gestión de bases de datos.

Manejo de FK (Foreign Key) y PK (Primary Key): El uso correcto de los datos a través de llaves primarias y llaves foráneas en las relaciones entre tablas.

Objetivo

El objetivo de este informe es presentar el diseño y la implementación de una base de datos relacional enfocándonos en un sistema de ventas en MySQL, destacando la gestión de inventarios, el registro del stock o ventas y la mejora en la administración de datos de clientes. Se quiere mostrar cómo la base de datos, junto con el uso de consultas SQL, puede optimizar los procesos operativos de un negocio, asegurando que la información sea la correcta.

Metodología

Herramientas Utilizadas

Figura 1. MySQL Workbench

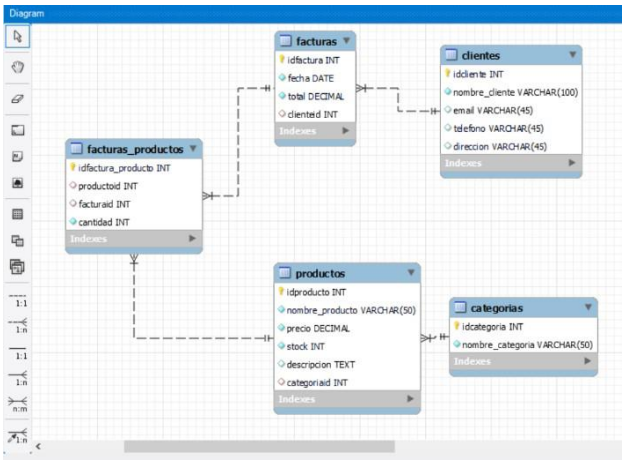


Figura 2. MySQL Server

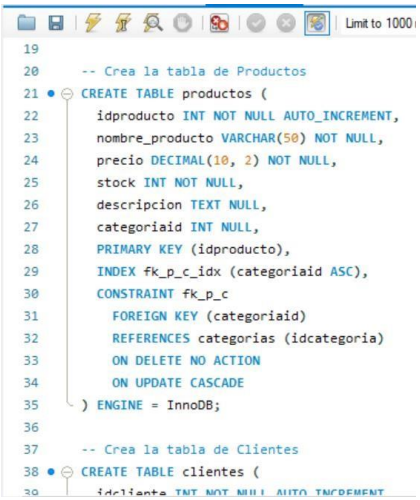


Figura 3. GitHub

Figura 4. Stack Overflow

Es una comunidad en línea donde distintos desarrolladores de todo el mundo hacen preguntas y comparten respuestas sobre una amplia gama de temas de programación, incluyendo consultas SQL

Procedimientos

I. Diseño del Esquema de la Base de Datos

- Comenzamos con la identificación de las entidades clave de la empresa en este caso serían: productos, categorías, clientes, facturas y ventas
- Posteriormente relacionamos las tablas, asegurando la correcta normalización de los datos para evitar redundancias y mejorar las referencias

II. Creación de la Base de Datos en MySQL Workbench

- Se usó MySQL Workbench para la creación de las tablas y la definición de las claves primarias y foráneas
- Se definieron los tipos de datos apropiados para cada columna (int, decimal, varchar, etc), asegurándonos de tener compatibilidad en el tipo de datos, para el correcto funcionamiento de consultas futuras

III. Llenado de Tablas

- Se añadieron datos de ejemplo en cada tabla para simular un entorno real de ventas
- Se verificó que las claves foráneas se actualizaran correctamente y que los datos estuvieran alineados con los tipos de datos definidos

IV. Ejecución de Consultas SQL

- Se desarrollaron consultas SQL para extraer, analizar y manipular datos de las tablas creadas

- Se aplicaron conceptos de SQL, incluyendo subconsultas joins, group by, filtros y operadores lógicos para resolver problemas específicos de la gestión de inventarios y ventas.

V. Análisis de Resultados

- Los resultados de las consultas fueron analizados para verificar que la respuesta sea lo que se pide
- Se realizaron análisis detallados de los resultados obtenidos, evaluando su utilidad

Desarrollo

Descripción de la Base de Datos

Tablas y Llaves:

1. **Productos**: Contiene información sobre los productos disponibles para la venta, como nombre, precio, stock y descripción.

    - Clave Principal (PK): idproducto
    - Se conecta con la tabla facturas_productos y categorias.

2. **Categorías**: Almacena las diferentes categorías a las que pertenecen los productos.

    - Clave Principal (PK): idcategoria
    - Relacionada con productos mediante una clave foránea.

3. **Clientes**: Registra los datos de los clientes que realizan las compras, como nombre, email, teléfono y dirección.

    - Clave Principal (PK): idcliente
    - Relacionada con facturas mediante una clave foránea

4. **Facturas**: Guarda los detalles de las transacciones, como la fecha y el cliente asociado.

    - Clave Principal (PK): idfactura
    - Conectada con clientes y facturas_productos.

5. **Facturas_Productos**: Tabla intermedia que conecta facturas y productos, permitiendo la relación de muchos a muchos entre ambas.

    - Clave Principal (PK): idfactura_producto
    - Llave foránea facturaid a facturas.
    - Llave foránea productoid a productos.

Relaciones:

- **Muchos a Muchos**: Entre facturas y productos a través de la tabla facturas_productos.
- **Uno a Muchos**: Entre categorias y productos.
- **Uno a Muchos**: Entre clientes y facturas.

Consultas SQL

1. Muestra el Stock del Producto Antes y Después de una Venta

```sql
Use tienda_prueba;

START TRANSACTION;
-- B1
SET @stock_actual = (
  SELECT stock
  FROM productos
  WHERE idproducto = 1);
-- B2
UPDATE productos
SET stock = stock - (
  SELECT cantidad
  FROM facturas_productos
  WHERE productoid = 1
  AND facturaid = 1)
WHERE idproducto = 1;
-- B3
SELECT
    nombre_producto as producto,
    @stock_actual AS stock_actual,
    stock AS stock_actualizado
FROM productos
WHERE idproducto = 1;
-- B4
COMMIT;
```

- Primero nos aseguramos de usar la BD por medio del comando USE
- **B1.** START TRANSACTION y COMMIT las usamos para encapsular el procedimiento (el cual si hay un error en medio de la consulta esta no se ejecutara en la BD o se puede usar ROLLBACK) y tener el uso de @variable_temporal, donde haremos una **subconsulta**, guardando el valor del stock_actual en la tabla PRODUCTOS por medio de un SET
- **B2.** Como vamos a hacer un **update** a la tabla **productos**, primero llamamos el campo a modificar y lo igualamos el mismo valor menos (-) una subconsulta, donde preguntamos la cantidad que se compró un producto de una factura especifica (esto por medio de ID's), el resultado deber el stock actual – la cantidad del producto comprado
- **B3.** Aquí solo llamamos los valores antes ejecutados, mostramos el nombre del producto, una columna del antes y del después sobre el stock del producto
- **B4.** Cerramos el encapsulamiento por medio del comando **commit**

	A	B	C
1	producto	stock_actual	stock_actualizado
2	jabón	30	28

2. Filtrar Productos en Baja Disponibilidad

```sql
1 SELECT nombre_producto, stock
2 FROM productos
3 WHERE stock < 10;
```

De la tabla **productos**, usamos el nombre, stock, **donde** debe mostrar los ítems de stock menores (<) a 10

	A	B
1	nombre_producto	stock
2	Cereal	4
3	Televisor	9

3. Según el registro de ventas, cuales son los 3 productos mas vendidos

```sql
1 SELECT
2   p.nombre_producto,
3   SUM(fp.cantidad) AS total_vendido
4 FROM facturas_productos fp
5 JOIN productos p
6 ON fp.productoid = p.idproducto
7 GROUP BY p.idproducto
8 HAVING total_vendido >= 10
9 ORDER BY total_vendido
10 DESC Limit 3;
```

- Comenzamos seleccionando y apodando la tabla facturas_productos como “fp”, a través de un **join** la conectamos con la tabla productos (apodada “p”) por medio de la **FK** y **PK.** (Linea 4 a 6)
- Seleccionamos las entidades a usar, en este caso **p.nombre_producto** y **fp.cantidad** (apodada total_vendido), estaríamos asignando una cantidad total (la cual se la consigue por medio de un operador llamado **sum)** a cada producto. (Linea 1 a 3)
- Agrupamos los resultados por producto por medio de un operador llamado **group by**
- Usamos el **having** para filtrar solo los productos con mas de 10 unidades vendidas
- Ordenamos los productos según el total vendido en orden descendente, por medio del uso de **order by** y finaliza el código con un **limit 3** lo cual tiene como función que por salida de la consulta nos mostrará solo 3 items

	A	B
1	nombre_producto	total_vendiddo
2	camiseta	13
3	televisor	12
4	cereal	11

4. Consulta para Calcular el Total de Ventas por Categoría

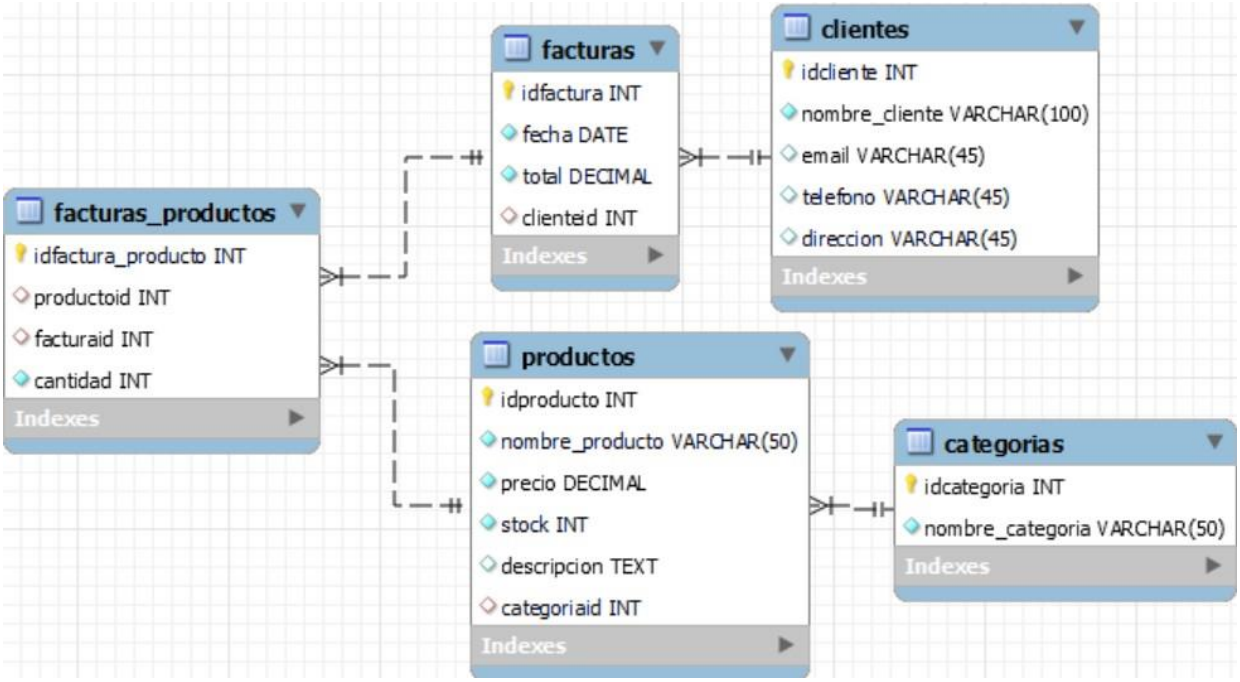
```sql
1 SELECT
2 c.nombre_categoria,
3 SUM(fp.cantidad*p.precio) AS total_ventas
4 FROM facturas_productos fp
5 JOIN productos p ON fp.productoid = p.idproducto
6 JOIN categorias c ON p.categoriaid = c.idcategoria
7 GROUP BY c.idcategoria
8 ORDER BY total_ventas DESC;
```

- Comenzamos seleccionando y apodando la tabla facturas_productos como "fp", a través de un **join** la conectamos con la tabla productos (apodada "p"), con el uso de otro **join** conectamos la anterior tabla con la de categorías (apodada "c") y todo esto por medio de la **FK** y **PK.** (Linea 4 a 6)
- Seleccionamos las entidades a usar, en este caso **c.nombre_categoria, fp.cantidad , p.precio**, y sacamos el total vendido de cada categoría por medio del operador **sum ()** y dentro de este multiplicamos la cantidad vendida del producto por el precio unitario. (Linea 1 a 3)
- Agrupamos los resultados por categoria, por medio del **group by** y usamos el id (c.idcategoria)
- Agrupamos los resultados por producto por medio de un operador llamado **group by**
- Usamos el **having** para filtrar solo los productos con mas de 10 unidades vendidas
- Ordenamos los productos según el total vendido en orden descendente, por medio del uso de **order by** y **desc.**

	A	B
1	nombre_categoria	total_ventas
2	Electronica	3600.00
3	Ropa	195.00
4	Alimentos	46.50
5	Higiene	29.00

Diseño de la Base de Datos

1. Diagrama entidad relación (ERD)



2. Normalización

a) Primera Forma Normal (1NF):

- **Objetivo:** Asegurar que los valores en cada columna sean minimos y que cada fila tenga un identificador único.
- **Aplicación en la BD:**
  - Se creó la tabla **productos** donde cada registro contiene información específica de un producto (nombre, precio, stock, etc.), asegurando que no haya datos repetidos dentro de una sola columna.
  - La tabla **categorías** contiene los nombres de las categorías de productos, con una columna única **idcategoria** que identifica cada categoría.

- La tabla **facturas** guarda información única de cada venta, con **idfactura** como su clave primaria.

b) Segunda Forma Normal (2NF):

- **Objetivo:** Cada columna debe depender completamente de la clave primaria, es decir cada contenido de la tabla debe ser lo necesario en informacion
- **Aplicación en la BD:**
  - Las tablas **productos** y **categorias** se separan para eliminar la dependencia parcial, es decir una redundancia por medio id's. En lugar de repetir la categoría de un producto en cada fila de la tabla **productos**, se conecta a la tabla **categorias** usando **categoriaid**.
  - Se evitó incluir detalles del cliente dentro de la tabla de facturas al crear la tabla **clientes**, relacionando las facturas con los clientes mediante **clienteid**.

c) Tercera Forma Normal (3NF):

- **Objetivo:** Eliminar las dependencias transitivas (cuando una columna depende de otra columna que no es clave primaria).
- **Aplicación en la BD:**
  - Se creó la tabla **facturas_productos** para gestionar la relación muchos a muchos entre **facturas** y **productos**. Esta tabla evita la duplicación de datos, ya que en lugar de tener una lista de productos en cada factura, simplemente relaciona el **idfactura** con **idproducto** y especifica la cantidad.

3. Consideraciones de Diseño

El diseño de la base de datos se desarrolló para asegurar su eficiencia, escalabilidad y facilidad de mantenimiento. A continuación presento las consideraciones clave que se tomaron en cuenta durante el diseño:

a) **Elección de claves primarias**

   Se seleccionaron claves primarias para cada tabla basadas en identificadores únicos y autoincrementales. Por ejemplo, **idproducto** para la tabla **productos**, **idfactura** para la tabla **facturas**, y **idcategoria** para la tabla **categorias**.

   Las claves primarias garantizan que el elemento sea único en los registros y permiten una referencia eficiente en otras tablas a través de claves foráneas.

b) **Uso de llaves foráneas y la relación**

   Las llaves foráneas se utilizaron para definir las relaciones entre las tablas. Por ejemplo, **categoriaid** en la tabla **productos** conecta los productos con su respectiva categoría en la tabla **categorias**.

   En la tabla de relación **facturas_productos**, las claves foráneas **facturaid** y **productoid** unen los registros de facturas y productos, representando la venta de cada producto

Análisis

Las consultas realizadas muestran cómo la estructura de la base de datos permite extraer información relevante de manera organizada y eficiente. Por ejemplo, la consulta que muestra los productos más vendidos nos permite identificar qué productos tienen mayor demanda, lo cual es muy importante para la toma de decisiones comerciales, como ajustar el inventario o lanzar promociones específicas, también se podría hablar de la consulta que calcula el total de ventas por categoría, este tipo de consulta permite identificar cuáles categorías generan más ingresos, lo cual es fundamental para el informe en el mercado. Con estos datos, se pueden tomar decisiones informadas, como enfocar esfuerzos de marketing en las categorías más rentables o ajustar las estrategias de compra y almacenamiento de productos para maximizar las ganancias. Entonces se tendría muy en cuenta que los resultados obtenidos de las consultas SQL demuestra la eficiencia al tener una base de datos bien estructurada. Las consultas permiten extraer información, donde también hay que tener en cuenta la toma de decisiones en la gestión de ventas y productos.

Conclusión

A lo largo de este trabajo, se aprendió la importancia de un buen manejo de bases de datos para optimizar la administración de información en un entorno comercial. El uso de MySQL permitió no solo diseñar una estructura de la base de datos, sino también entender cómo las consultas pueden ser herramientas cruciales para la toma de decisiones. Además, este proyecto demuestra que la teoría de bases de datos aplicada correctamente puede transformar datos vacíos en información importante para un negocio.

Referencias

- https://www.geeksforgeeks.org/introduction-of-database-normalization/?ref=gcse_ind
- https://stackoverflow.com/search?q=how+to+use+subqueries+in+SQL&s=dd753150-e643-48bd-85cf-d4792c22b881
- https://www.w3schools.com/sql/sql_join.asp