

Proyecto: Base de Datos TaconMadre

Alumno: Abner Soberon Martinez

Profesor: Ricardo Monrroy

Clase: G4-25 Introduccion a bases de datos

Este documento evalúa el proyecto de base de datos MySQL **TaconMadre** con base en cinco criterios clave requeridos para un sistema de gestión de pedidos en un restaurante.

1. Requisitos del sistema e identificación de entidades

El proyecto define e identifica correctamente todas las entidades esenciales para gestionar las operaciones de un restaurante, incluyendo usuarios, empleados, productos, pedidos, proveedores, ventas y compras. Cada entidad cuenta con atributos adecuados, relaciones bien establecidas mediante claves foráneas, y restricciones claras como `NOT NULL`, campos auto-incrementales y tipos enumerados (`ENUM`). La estructura refleja de manera fiel los procesos del mundo real y permite manejar tanto pedidos físicos como digitales de forma eficiente.

2. Coherencia, normalización y estructura organizativa

La base de datos está altamente normalizada y organizada de forma lógica, lo cual garantiza coherencia, integridad de los datos y facilidad de mantenimiento. Los módulos funcionales están claramente separados (usuarios, inventario, ventas, compras, logística, recursos humanos, etc.), y se integran componentes avanzados como gestión de almacenes, movimientos de inventario, bitácora de actividades del personal, y diversos métodos y canales de pago. Esta estructura facilita el crecimiento futuro del sistema y un desempeño óptimo en operaciones reales.

3. Datos de muestra y validación práctica

El archivo ``consultas.sql`` incluye ejemplos básicos de datos de muestra, como inserciones en las tablas ``locations`` y ``orders``, lo cual permite validar parcialmente el funcionamiento del sistema. Sin embargo, sería recomendable complementar con más registros de prueba que incluyan productos, empleados, clientes y ventas completas para realizar simulaciones más amplias y comprobar a fondo la funcionalidad e integridad de la base de datos.

4. Uso de consultas SQL avanzadas

El proyecto incorpora consultas SQL avanzadas que permiten extraer información valiosa y generar reportes estratégicos. Se utilizan correctamente estructuras como ``JOIN``, ``GROUP BY``, funciones de agregación (``SUM``, ``COUNT``), filtros de fechas (``INTERVAL``, ``DATE_FORMAT``), y condiciones lógicas. Las consultas presentadas permiten responder preguntas clave del negocio como: ventas por mes, productos más vendidos, análisis por canal de venta y pedidos pendientes de envío, lo cual demuestra un dominio sólido del lenguaje SQL orientado al análisis de datos.

5. Documentación técnica y claridad

La documentación técnica es clara y bien estructurada. El archivo principal contiene un encabezado con el nombre del proyecto, autor, además de comentarios explicativos en cada tabla. Las claves foráneas están bien nombradas y se sigue una convención consistente. El archivo de consultas también cuenta con comentarios que explican cada bloque de código. Para una documentación

Consultas:

-- 1. Ventas totales por mes (últimos 12 meses)

SELECT DATE_FORMAT(sale_date,'%Y-%m') AS mes,

ROUND(SUM(total_amount),2) AS total_ventas

FROM sales

WHERE sale_date >= DATE_SUB(CURDATE(), INTERVAL 12 MONTH)

GROUP BY mes

ORDER BY mes;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1. Ventas totales por mes (últimos 12 meses)
2. SELECT DATE_FORMAT(sale_date,'%Y-%m') AS mes,
3. ROUND(SUM(total_amount),2) AS total_ventas
4. FROM sales
5. WHERE sale_date >= DATE_SUB(CURDATE(), INTERVAL 12 MONTH)
6. GROUP BY mes
7. ORDER BY mes;
```

The Results tab shows the following data:

mes	total_ventas
2024-06	1540.00
2024-07	7650.00
2024-08	8000.00
2024-09	7200.00
2024-10	7800.00
2024-11	7800.00
2024-12	7500.00

The bottom of the screenshot shows the 'Action Output' tab with the following information:

Time	Action	Rows	Bytes	Temp
15.15.12	SELECT (name, SUM(quantity)) AS total_venta FROM sales JOIN sales AS s2 ON sale_date = s2.sale_date JOIN products p ON p.product_id = s2.product_id WHERE sale_date = DATE_SUB(CURDATE(), INTERVAL 12 MONTH)	15 rows returned	0.016 sec / 0.000 sec	0 rows
15.15.12	SELECT DATE_FORMAT(sale_date,'%Y-%m') AS mes, ROUND(SUM(total_amount),2) AS total_ventas FROM sales GROUP BY mes ORDER BY mes DESC --reverse DESC LIMIT 1000	10 rows returned	0.016 sec / 0.000 sec	0 rows
15.15.12	SELECT DATE_FORMAT(sale_date,'%Y-%m') AS mes, ROUND(SUM(total_amount),2) AS total_ventas FROM sales WHERE sale_date = DATE_SUB(CURDATE(), INTERVAL 12 MONTH) GROUP BY mes	7 rows returned	0.005 sec / 0.000 sec	0 rows

-- 2. Top-5 productos más vendidos por cantidad

SELECT p.name, SUM(si.quantity) AS unidades

FROM sale_items si

JOIN products p ON p.product_id = si.product_id

GROUP BY p.product_id

ORDER BY unidades DESC

LIMIT 5;

The screenshot shows the DBeaver SQL editor interface. The main window displays a SQL query with line numbers 1 through 31. The query is as follows:

```
1 use Ecommerce;
2
3 -- 1. Ventas totales por mes (21lines 12 rows)
4 SELECT DATE_FORMAT(sale_date, '%Y-%m') AS mes,
5        SUM(quantity) AS total_venta
6 FROM sales
7 WHERE sale_date >= DATE_SUB(CURRENT(), INTERVAL 12 month)
8 GROUP BY mes;
9
10 ORDER BY mes;
11
12
13 -- 2. Top-5 productos más vendidos por cantidad
14 SELECT p.name, SUM(si.quantity) AS unidades
15 FROM sales si
16 JOIN products p ON p.product_id = si.product_id
17 GROUP BY p.product_id
18 ORDER BY unidades DESC;
19
20
21 -- 3. Ventas por canal
22 SELECT channel, SUM(quantity) AS total
23 FROM sales
24 GROUP BY channel;
25
26 ORDER BY total DESC;
27
28
29 -- 4. Top-5 productos más vendidos en el último año
30 SELECT p.name, SUM(si.quantity) AS total_venta
31 FROM sales si
32 JOIN products p ON p.product_id = si.product_id
33 WHERE sale_date >= DATE_SUB(CURRENT(), INTERVAL 12 month)
34 GROUP BY p.product_id;
```

The results pane at the bottom shows the output of the query. It contains a table with two columns: 'name' and 'unidades'. The data is as follows:

name	unidades
homburgueta	312
Paños de limpieza	288
Taza de cerámica	105
Paños de limpieza	105
Taza de cerámica	105

The status bar at the bottom indicates that the query was completed successfully.

-- 3. Ventas por canal

SELECT channel, ROUND(SUM(total_amount),2) AS total

FROM sales

GROUP BY channel

ORDER BY total DESC;

The screenshot shows a database management tool interface with a SQL query editor and a results pane. The query is as follows:

```
USE factwarehouse;

-- 1. Ventas totales por mes (21 lines 12 mins)
1 SELECT DATE_TRUNC('month', sales_date) AS mes,
2 ROUND(SUM(total_amount)) AS total_venta
3 FROM sales
4 GROUP BY mes
5 ORDER BY mes;
```

The results pane shows the following data:

channel	total
uber_app	14430.00
uber_web	14300.00
app_ios	14010.00
app_android	13800.00

The interface also includes a sidebar with a database schema tree, a top menu bar, and a bottom status bar.

-- 4. Top-5 productos más vendidos en el último año

SELECT p.name, SUM(si.quantity) AS total_sold

FROM sale_items si

JOIN sales s ON s.sale_id = si.sale_id

JOIN products p ON p.product_id = si.product_id

WHERE s.sale_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)

GROUP BY p.product_id

ORDER BY total_sold DESC

LIMIT 5;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
7 WHERE sale_date >= DATE_SUB(CURDATE(), INTERVAL 12 MONTH)
8 GROUP BY mes
9 ORDER BY mes;
10
11 -- 3. Top-5 productos más vendidos por categoría
12 *
13 SELECT p.category, SUM(si.quantity) AS total_sold
14 FROM sale_items si
15 JOIN products p ON p.product_id = si.product_id
16 GROUP BY p.product_id
17 ORDER BY unidades DESC
18 LIMIT 5;
19
20 -- 3. Ventas por canal
21 *
22 SELECT channel, SUM(s.total_amount) AS total
23 FROM sales
24 GROUP BY channel
25 ORDER BY total DESC;
26
27 -- 4. Top-5 productos más vendidos en el último año
28 *
29 SELECT p.name, SUM(si.quantity) AS total_sold
30 FROM sale_items si
31 JOIN sales s ON s.sale_id = si.sale_id
32 JOIN products p ON p.product_id = si.product_id
33 WHERE s.sale_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
34 GROUP BY p.product_id
35 ORDER BY total_sold DESC
36 LIMIT 5;
37
38 -- 5. Ventas por canal y mes
39 *
40 SELECT DATE_FORMAT(sale_date, '%Y-%m') AS mes,
41        ROUND(SUM(s.total_amount)) AS total_venta FROM sales WHERE sale_date >= DATE_SUB(CURDATE(), INTERVAL 12 MONTH) GROUP BY mes ...
42
43 -- 6. Top-5 productos más vendidos en el último año
44 *
45 SELECT p.name, SUM(si.quantity) AS total_sold FROM sale_items si JOIN products p ON p.product_id = si.product_id WHERE s.sale_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR) GROUP BY p.product_id ORDER BY total_sold DESC LIMIT 5;
```

The Results tab shows the following data:

name	total_sold
Hamburguesas	54
Paquete de 100 unidades	46
Paquete de 20 unidades	20
Paquete de 50 unidades	20
Paquete de 10 unidades	20

The Output tab shows the execution of the query, with the following results:

Query	Duration / Rows
17:14:50 SELECT DATE_FORMAT(sale_date, '%Y-%m') AS mes, ROUND(SUM(s.total_amount)) AS total_venta FROM sales WHERE sale_date >= DATE_SUB(CURDATE(), INTERVAL 12 MONTH) GROUP BY mes ...	0.000 sec / 0.000 rows
18:14:57 SELECT p.name, SUM(si.quantity) AS total_sold FROM sale_items si JOIN products p ON p.product_id = si.product_id WHERE s.sale_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR) GROUP BY p.product_id ORDER BY total_sold DESC LIMIT 5;	0.016 sec / 0.000 rows
19:14:59 SELECT channel, SUM(s.total_amount) AS total FROM sales GROUP BY channel ORDER BY total DESC LIMIT 5;	0.000 sec / 0.000 rows
20:14:56 SELECT p.name, SUM(si.quantity) AS total_sold FROM sale_items si JOIN products p ON p.product_id = si.product_id WHERE s.sale_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR) GROUP BY p.product_id ORDER BY total_sold DESC LIMIT 5;	0.000 sec / 0.000 rows

-- 5. Ventas por canal y mes

SELECT DATE_FORMAT(sale_date,'%Y-%m') AS yyyyymm,

channel,

SUM(total_amount) AS revenue

FROM sales

GROUP BY yyyyymm, channel

ORDER BY yyyyymm DESC, revenue DESC;

