BOTIC-SOFOF-24-28-13-0081

Módulo 5: Fundamentos de bases de datos relacionales (56 Horas) / ELE069950089613

M5_Evaluación del Módulo [Actividad Evaluada]

M5_Evaluación del Módulo [Actividad Evaluada]

Evaluación del módulo

objetivo

El objetivo de esta actividad es que puedas diseñar y desarrollar un sistema de gestión de inventario para una empresa que utiliza una base de datos relacional para almacenar y consultar información sobre productos, proveedores y transacciones. Además, aprenderás cómo manejar transacciones, restricciones de integridad referencial, y consultas complejas utilizando SQL.

Contexto

En una empresa de ventas, es necesario gestionar un inventario de productos que se van almacenando en diferentes proveedores, y a su vez, realizar transacciones de compra y venta. Para esto, se utiliza una base de datos relacional (RDBMS) para organizar toda la información de manera estructurada. El sistema debe permitir agregar productos, actualizar el inventario, registrar compras y ventas, y consultar los datos de manera eficiente.

Requisitos del Proyecto

1. Diseño del Modelo Relacional

- Crea un modelo de datos para representar la siguiente información:
  - **Productos**: nombre, descripción, precio, cantidad en inventario.
  - **Proveedores**: nombre, dirección, teléfono, email.
  - **Transacciones**: tipo (compra o venta), fecha, cantidad, id de producto, id de proveedor.
- Utiliza el modelo **Entidad-Relación** (ER) para abstraer las entidades y sus relaciones, y luego tradúcelas al modelo relacional.
- Asegúrate de identificar claves primarias y foráneas para establecer la integridad referencial entre las tablas.

2. Creación de la Base de Datos y Tablas

- Utiliza SQL para crear las tablas **productos, proveedores y transacciones** en la base de datos.
- Define las restricciones de **nulidad, llaves primarias y llaves foráneas** para garantizar la integridad de los datos.
- Establece el tipo de dato adecuado para cada atributo (por ejemplo, V**ARCHAR, INT, DECIMAL**).

3. Consultas Básicas

- Realiza consultas básicas utilizando el lenguaje SQL:
  - Recupera todos los productos disponibles en el inventario.
  - Recupera todos los proveedores que suministran productos específicos.
  - Consulta las transacciones realizadas en una fecha específica.
  - Realiza consultas de selección con funciones de agrupación, como **COUNT()** y **SUM()**, para calcular el número total de productos vendidos o el valor total de las compras.

4. Manipulación de Datos (DML)

- Inserta datos en las tablas **productos, proveedores y transacciones**.
- Actualiza la cantidad de inventario de un producto después de una venta o compra.
- Elimina un producto de la base de datos si ya no está disponible.
- Asegúrate de aplicar **integridad referencial** al actualizar o eliminar registros relacionados.

5. Transacciones SQL

- Realiza una transacción para registrar una compra de productos. Utiliza el comando **BEGIN TRANSACTION, COMMIT y ROLLBACK** para asegurar que los cambios se apliquen correctamente.
- Asegúrate de que los cambios en la cantidad de inventario y las transacciones se realicen de forma atómica.
- Utiliza el modo **AUTOCOMMIT** para manejar operaciones individuales si es necesario.

6. Consultas Complejas

- Realiza una consulta que recupere el total de ventas de un producto durante el mes anterior.
- Utiliza **JOINs** (INNER, LEFT) para obtener información relacionada entre las tablas productos, **proveedores y transacciones**.
- Implementa una consulta con subconsultas (subqueries) para obtener productos que no se han vendido durante un período determinado.

7. Normalización y Desnormalización

- Asegúrate de que las tablas estén **normalizadas** hasta la tercera forma normal (3NF) para evitar redundancias y asegurar la integridad de los datos.
- Si es necesario, discute en tu informe los casos en los que la desnormalización podría ser útil para mejorar el rendimiento de las consultas.

8. Manejo de Excepciones y Restricciones

- Implementa **restricciones** en los campos de las tablas para garantizar que los datos ingresados sean válidos (por ejemplo, asegurando que la cantidad de inventario no sea negativa o que los precios sean mayores que cero).
- Utiliza el comando **TRY/CATCH** para manejar excepciones en caso de errores durante las transacciones.

9. Documentación

- Documenta el proceso de creación del modelo de datos y las decisiones tomadas al diseñar las tablas, restricciones y relaciones entre entidades.
- Incluye una breve explicación de la normalización aplicada en el modelo de datos y su impacto en la estructura de la base de datos.
- Presenta ejemplos de las consultas SQL utilizadas y explica cómo funcionan.

Estado de la entrega

Estado de la entrega	Todavía no se han realizado envíos
Estado de la calificación	Sin calificar
Última modificación	-

DESARROLLO M5_EVALUACIÓN DE MÓDULO

1. Diseño del Modelo Relacional

Para el sistema de gestión de inventario se identificaron las siguientes entidades y atributos principales:

- Productos: producto_id (PK), nombre, descripción, precio, cantidad.
- Proveedores: proveedor_id (PK), nombre, dirección, teléfono, email.
- Transacciones: transaccion_id (PK), tipo (compra o venta), fecha, cantidad, producto_id (FK), proveedor_id (FK).

Se definieron las relaciones de la siguiente manera:

- Un Proveedor puede suministrar uno o varios Productos.
- Un Producto puede estar asociado a varias Transacciones.
- Una Transacción siempre hace referencia a un Producto y opcionalmente a un Proveedor.

De este análisis se obtiene el siguiente modelo relacional normalizado en 3FN:

Tabla PRODUCTOS

- producto_id (PK)
- nombre
- descripcion
- precio
- cantidad

Tabla PROVEEDORES

- proveedor_id (PK)
- nombre
- direccion
- telefono
- email

Tabla TRANSACCIONES

- transaccion_id (PK)
- tipo
- fecha
- cantidad
- producto_id (FK → productos.producto_id)
- proveedor_id (FK → proveedores.proveedor_id)

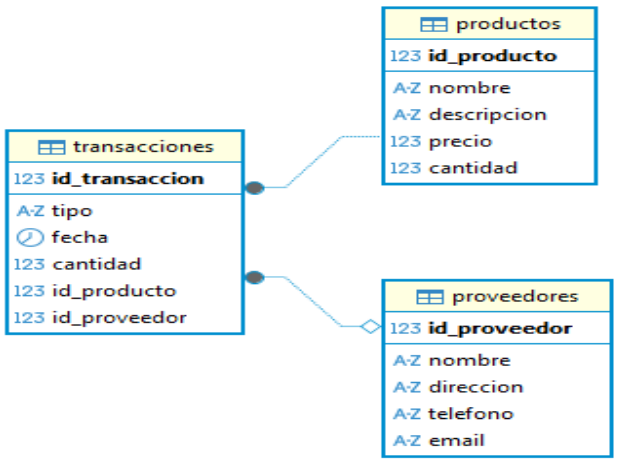
El diseño asegura la integridad referencial mediante el uso de claves primarias y foráneas, evita redundancias y mantiene la información organizada.

2. Creación de la Base de Datos y Tablas

```
/*
 * Crear la base de datos
 */
CREATE DATABASE inventario;

/*
 * Conectarse a la base manualmente, ya esta opción no es reconocida en DBeaver.
 */
-- \c inventario;
```

```
/*
 * Creación de tablas
 */
CREATE TABLE proveedores (
   id_proveedor   SERIAL PRIMARY KEY,
   nombre         VARCHAR(100) NOT NULL,
   direccion      VARCHAR(150),
   telefono       VARCHAR(10),
   email          VARCHAR(100) UNIQUE
);

CREATE TABLE productos (
   id_producto    SERIAL PRIMARY KEY,
   nombre         VARCHAR(100) NOT NULL,
   descripcion    VARCHAR(150),
   precio         NUMERIC(10,2) NOT NULL CHECK (precio > 0),
   cantidad       INT NOT NULL CHECK (cantidad >= 0)
);

CREATE TABLE transacciones (
   id_transaccion   SERIAL PRIMARY KEY,
   tipo             VARCHAR(10) NOT NULL CHECK (tipo IN ('compra','venta')),
   fecha            DATE NOT NULL DEFAULT CURRENT_DATE,
   cantidad         INT NOT NULL CHECK (cantidad > 0),
   id_producto      INT NOT NULL,
   id_proveedor     INT,
   CONSTRAINT fk_producto  FOREIGN KEY (id_producto) REFERENCES productos (id_producto) ON DELETE
CASCADE,
   CONSTRAINT fk_proveedor FOREIGN KEY (id_proveedor) REFERENCES proveedores (id_proveedor) ON DELETE
SET NULL
);
```

Diagrama Entidad–Relación (ERD)



3. Consultas Básicas

```sql
/*
 *  Recupera todos los productos disponibles en el inventario
 */
SELECT * FROM productos WHERE cantidad > 0;

/*
 *  Recupera todos los proveedores que suministran un producto específico
 */
SELECT DISTINCT p.nombre, p.direccion, p.telefono, p.email
FROM proveedores p
JOIN transacciones t ON p.id_proveedor = t.id_proveedor
WHERE t.id_producto = 1;

/*
 *  Consultar las transacciones realizadas en una fecha específica
 */
SELECT * FROM transacciones WHERE fecha = '2025-08-01';

/*
 *  Calcular el número total de productos vendidos
 */
SELECT SUM(cantidad) AS total_vendidos
FROM transacciones
WHERE tipo = 'venta';

/*
 * -- Calcular el valor total de compras
 */
SELECT SUM(t.cantidad * pr.precio) AS total_compras
FROM transacciones t
JOIN productos pr ON t.id_producto = pr.id_producto
WHERE t.tipo = 'compra';
```

4. Manipulación de Datos (DML)

```sql
/*
 *  Inserta datos en las tablas productos, proveedores y transacciones.
 */
INSERT INTO proveedores (nombre, direccion, telefono, email)
VALUES  ('TCHILE' ,'Av. Uno' ,'981276346','ventas@tchile.com'),
        ('CDCOMP' ,'Av. Dos' ,'123456789','ventas@cdcom.com'),
        ('INTDATA','Av. Tres','987654321','ventas@intdata.com');

INSERT INTO productos (nombre, descripcion, precio, cantidad)
VALUES   ('Parlante','Parlante HD Inalambrico', 2000   , 10),
         ('Mouse'   ,'Mouse óptico'           , 1000   , 20),
         ('Monitor' ,'Monitor 24" '           , 125000,  5)

INSERT INTO transacciones (tipo, fecha, cantidad, producto_id, proveedor_id)
VALUES   ('compra','2025-08-01',  5,  1, 1),
         ('venta'    ,'2025-08-10',  2, 1, NULL),
         ('compra','2025-08-15', 10, 2, 2),
         ('venta'    ,'2025-08-20',  1, 3, NULL);
```

```sql
/*
 * Actualiza la cantidad de inventario de un producto después de una venta o compra.
 */
UPDATE productos SET cantidad = cantidad - 5 WHERE id_producto = 3;

/*
 * Elimina un producto de la base de datos si ya no está disponible.
 */
DELETE FROM productos WHERE id_producto = 2 and cantidad = 0;
```

5. Transacciones SQL

```sql
/*
 * Muestra las transacciones actuales, antes de efectuar requerimientos
 */
SELECT * FROM transacciones;
-- Existen solo existen 4 transacciones (compra y ventas) registradas, y cantidad del producto 2 es 10
```

transacciones 1

SELECT * FROM transacciones | Enter a SQL expression to filter results (use Ctrl+Space)

	id_transaccion	tipo	fecha	cantidad	id_producto	id_proveedor
1	1	compra	2025-08-01	5	1	1
2	2	venta	2025-08-10	2	1	[NULL]
3	3	compra	2025-08-15	10	2	2
4	4	venta	2025-08-20	1	3	[NULL]

```sql
/*
 * Realiza una nueva transacción para registrar una compra de productos y actualiza la
 * cantidad del producto 2 de la tabla productos
 */
BEGIN;
   INSERT INTO transacciones (tipo, cantidad, id_producto, id_proveedor)
   VALUES ('compra', 5, 2, 1);
   UPDATE productos SET cantidad = cantidad + 5 WHERE id_producto = 2;
COMMIT;

SELECT * FROM transacciones;
```

transacciones 1

SELECT * FROM transacciones

	id_transaccion	tipo	fecha	cantidad	id_producto	id_proveedor
1	1	compra	2025-08-01	5	1	1
2	2	venta	2025-08-10	2	1	[NULL]
3	3	compra	2025-08-15	10	2	2
4	4	venta	2025-08-20	1	3	[NULL]
5	5	compra	2025-09-05	5	2	1

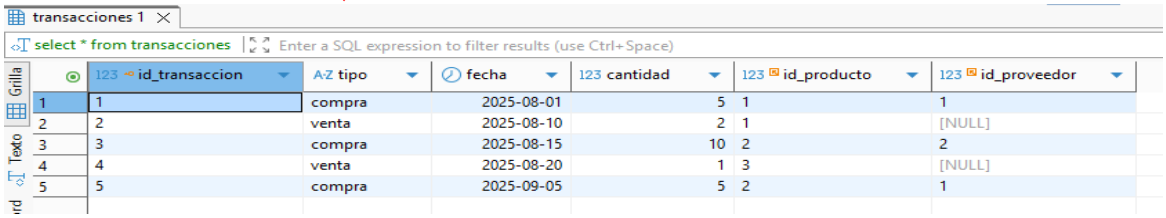
```sql
SELECT * FROM productos;
```

productos 1

SELECT * FROM productos | Enter a SQL expression to filter results (use Ctrl+Space)

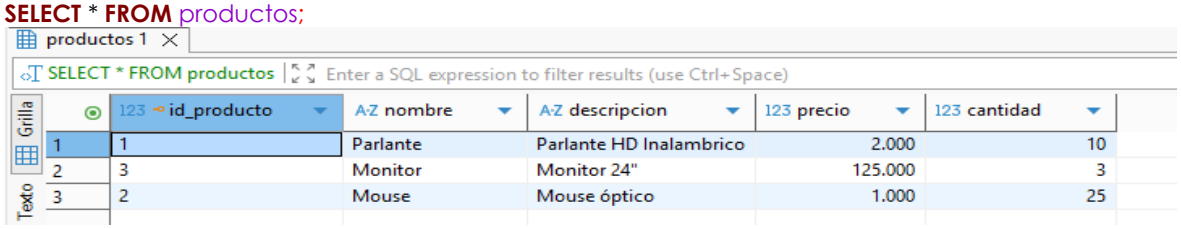
	id_producto	nombre	descripcion	precio	cantidad
1	1	Parlante	Parlante HD Inalambrico	2.000	10
2	3	Monitor	Monitor 24"	125.000	3
3	2	Mouse	Mouse óptico	1.000	25

```
/*
 * Realiza una nueva transacción para registrar una compra de productos y actualiza la
 * cantidad del producto 2 de la tabla productos, pero se efectuará un ROLLBACK para
 * anular las transacciones en las tablas respectivas.
 */
BEGIN;
   INSERT INTO transacciones (tipo, cantidad, id_producto, id_proveedor)
   VALUES ('venta', 100, 3, 1);
   UPDATE productos SET cantidad = cantidad - 100 WHERE id_producto = 3 and cantidad >= 100;
ROLLBACK;

-- A continuación, se muestran los registros sin cambios
SELECT * FROM transacciones;
```



	id_transaccion	tipo	fecha	cantidad	id_producto	id_proveedor
1	1	compra	2025-08-01	5	1	1
2	2	venta	2025-08-10	2	1	[NULL]
3	3	compra	2025-08-15	10	2	2
4	4	venta	2025-08-20	1	3	[NULL]
5	5	compra	2025-09-05	5	2	1

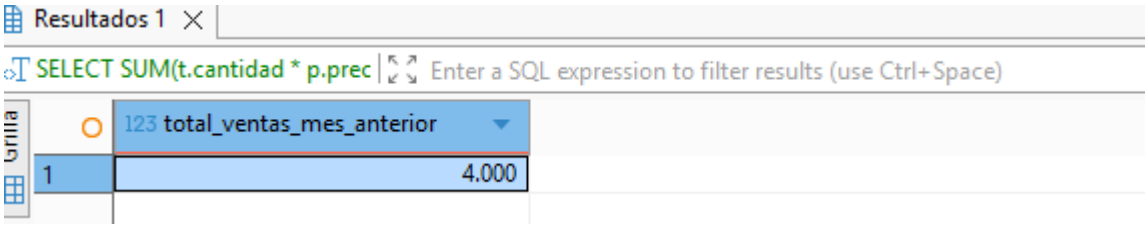
```
SELECT * FROM productos;
```



	id_producto	nombre	descripcion	precio	cantidad
1	1	Parlante	Parlante HD Inalambrico	2.000	10
2	3	Monitor	Monitor 24"	125.000	3
3	2	Mouse	Mouse óptico	1.000	25

6. Consultas Complejas

```
/*
 *  Realiza una consulta que recupere el total de ventas de un producto durante el mes
 *  anterior.
 */
SELECT SUM(t.cantidad * p.precio) AS total_ventas_mes_anterior
FROM transacciones t
JOIN productos p ON t.id_producto = p.id_producto
WHERE t.tipo = 'venta'
  AND t.id_producto = 1
  AND DATE_PART('month', t.fecha) = DATE_PART('month', CURRENT_DATE - INTERVAL '1 month')
  AND DATE_PART('year', t.fecha)  = DATE_PART('year', CURRENT_DATE - INTERVAL '1 month');
```



	total_ventas_mes_anterior
1	4.000

```sql
/*
 * Utiliza JOINs (INNER, LEFT) para obtener información relacionada entre las tablas
 * productos, proveedores y transacciones.
 */
SELECT    t.id_transaccion, t.tipo, t.fecha, t.cantidad,
          p.nombre AS producto, pr.nombre AS proveedor
FROM transacciones t
JOIN productos p ON t.id_producto = p.id_producto
LEFT JOIN proveedores pr ON t.id_proveedor = pr.id_proveedor;
```

transacciones(+) 1

SELECT t.id_transaccion, t.tipo, t.fecha, t.cantidad, p.nombre AS prod | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla | Texto

	id_transaccion	tipo	fecha	cantidad	producto	proveedor
1	1	compra	2025-08-01	5	Parlante	TCHILE
2	2	venta	2025-08-10	2	Parlante	[NULL]
3	3	compra	2025-08-15	10	Mouse	CDCOMP
4	4	venta	2025-08-20	1	Monitor	[NULL]
5	5	compra	2025-09-05	5	Mouse	TCHILE

```sql
/*
 * Implementa una consulta con subconsultas para obtener productos que no
 * se han vendido durante un período determinado.
 */
SELECT * FROM productos
WHERE id_producto NOT IN (
          SELECT id_producto FROM transacciones WHERE tipo = 'venta'
          AND fecha >= CURRENT_DATE - INTERVAL '30 days'
          );
```

productos 1

SELECT * FROM productos WHE | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla

	id_producto	nombre	descripcion	precio	cantidad
1	2	Mouse	Mouse óptico	1.000	25

Consideraciones con TRY/CATCH en PostgreSQL con DO $$

PostgreSQL no existe literalmente **TRY/CATCH** como en otros lenguajes, pero sí se logra el mismo comportamiento con el bloque BEGIN … EXCEPTION … END;.

En PL/pgSQL (lenguaje de procedimientos de PostgreSQL), la sintaxis equivalente a un **try/catch** es esta:

```
-- mostrar registro de las tablas productos y transacciones.
SELECT * FROM productos;
SELECT * FROM transacciones;

/*
 *  Al intentar registrar una venta mayor al stock, lanzará un error.
 *  El bloque captura la excepción y deja la base de datos consistente (no se descuenta
 *  inventario):
 */

DO $$
BEGIN
   BEGIN
      -- Venta mayor al cantidad existente (3), error  por stock insuficiente
      INSERT INTO transacciones (tipo, cantidad, id_producto, id_proveedor)
      VALUES ('venta', 100, 3, NULL);

      -- El producto id_producto = 3 no tiene suficiente stock.
      -- El UPDATE intenta dejar la columna cantidad en negativo → se activa el CHECK y falla.
      -- Error al registrar la transacción: el nuevo registro para la relación «productos»
      -- viola la restricción «check» «productos_cantidad_check»
      UPDATE productos
      SET cantidad = cantidad - 25
      WHERE id_producto = 3;

      RAISE NOTICE 'Venta registrada correctamente';
   EXCEPTION WHEN others THEN
      RAISE NOTICE 'Error al registrar la transacción: %', SQLERRM;
   END;
END$$;
rollback;
```