*UNIVERSIDAD PERUANA LOS ANDES (UPLA)*

# FACULTAD DE INGENIERÍA SISTEMAS Y COMPUTACIÓN

# EXAMEN PARCIAL


**INTEGRANTE:**

- **Luis Ramirez Brayan**


**ASIGNATURA:**

- **Base de Datos**

Huancayo - 2024

**Universidad Peruana Los Andes**

**Facultad de Ingeniería**

**Escuela Profesional de Ingeniería de Sistemas y Computación**

*Examen Final*

---

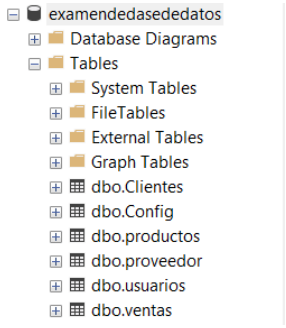
## Administración de Base de Datos

Apellidos y Nombres: __ Luis Ramirez Brayan Esnayder_____ Código: __R01086C__

Ciclo: ___V_____ Salón: _____

**Enunciado 01:**

De acuerdo con la **base de datos** desarrollada en **Microsoft SQL Server**, responda las siguientes preguntas:




### 1) Explique qué problema soluciona su base de datos:

La base de datos de Popeyes soluciona varios problemas comunes que enfrentan las pequeñas y medianas empresas en la gestión de información. Estos problemas incluyen:

1. **Desorganización en la información:** Muchas empresas carecen de un repositorio centralizado para almacenar y consultar información crítica, como los datos de clientes, proveedores y productos. La base de datos resuelve esto al centralizar toda la información en una base de datos estructurada, mejorando la organización y accesibilidad.
2. **Accesos no controlados:** Las empresas a menudo enfrentan dificultades para controlar el acceso a la información según el rol del usuario, lo que puede dar lugar a errores o accesos no autorizados. La base de datos implementa roles de usuario (administrador y asistente) con permisos diferenciados, asegurando que cada usuario solo pueda acceder a las funciones necesarias para su rol.
3. **Ineficiencia en el registro de transacciones:** Sin un sistema automatizado, el registro de ventas y otros procesos importantes pueden volverse manuales y desorganizados, lo que ralentiza las operaciones diarias. La base de datos automatiza estos procesos, optimizando el flujo de trabajo y reduciendo errores en el registro de transacciones.
4. **Problemas en la integridad de los datos:** El manejo manual de grandes volúmenes de datos puede generar inconsistencias e imprecisiones. La base de datos garantiza la integridad de los datos mediante un enfoque automatizado y centralizado, minimizando los riesgos de inconsistencias y mejorando la fiabilidad de la información almacenada.

2) Implemente un Script para crear una **vista** para crear utilizando tres tablas:

```sql
-- Creo la vista combinando las tablas ventas, clientes y productos
CREATE VIEW VistaVentasProductosClientes AS
SELECT
    v.id AS VentaID,                -- ID de la venta
    c.Nombre AS ClienteNombre,      -- Nombre del cliente
    p.nombre AS ProductoNombre,     -- Nombre del producto
    p.precio AS PrecioProducto,     -- Precio del producto
    v.total AS TotalVenta,          -- Total de la venta
    v.fecha AS FechaVenta           -- Fecha de la venta
FROM
    ventas v
JOIN
    clientes c ON v.cliente = c.ID  -- Unir ventas con clientes
JOIN
    productos p ON p.proveedor IN (SELECT id FROM proveedor)  -- Relación con proveedor, asumiendo lógica de proveedor
;
```

```
100 %
Messages
Commands completed successfully.

Completion time: 2024-12-21T11:48:30.0504717-05:00
```

3) Implemente un Script para crear un **procedimiento almacenado** para modificar el ingreso de datos en forma secuencial:

```sql
CREATE PROCEDURE ModificarDatosSecuenciales
    @ProductoID INT,                 -- ID del producto a modificar
    @NuevoNombreProducto NVARCHAR(180), -- Nuevo nombre del producto
    @NuevoPrecioProducto DECIMAL(10,2), -- Nuevo precio del producto
    @NuevoStockProducto INT,         -- Nuevo stock del producto
    @VentaID INT,                    -- ID de la venta a modificar
    @NuevoTotalVenta DECIMAL(10,2),  -- Nuevo total de la venta
    @ClienteID INT,                  -- ID del cliente a modificar
    @NuevoTelefonoCliente NVARCHAR(15) -- Nuevo teléfono del cliente
AS
BEGIN
    -- Iniciar una transacción para asegurar la consistencia de los datos
    BEGIN TRANSACTION;

    BEGIN TRY
        -- 1. Modificar el producto
        UPDATE productos
        SET nombre = @NuevoNombreProducto,
            precio = @NuevoPrecioProducto,
            stock = @NuevoStockProducto
        WHERE id = @ProductoID;

        -- 2. Modificar la venta
        UPDATE ventas
        SET total = @NuevoTotalVenta
        WHERE id = @VentaID;

        -- 3. Modificar el cliente
        UPDATE clientes
        SET Telefono = @NuevoTelefonoCliente
        WHERE ID = @ClienteID;

        -- Si todas las operaciones fueron exitosas, confirmar la transacción
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        -- Si ocurrió un error, deshacer la transacción
        ROLLBACK TRANSACTION;

        -- Manejar el error (opcional: puedes lanzar el error o devolver un mensaje)
        THROW;
    END CATCH;
END;
```

4) Implemente un Script para crear un **disparador** para verificar el control de datos (Ejemplo: que la nota ingresada este entre 0 y 20)

```sql
--4
CREATE TRIGGER VerificarDatosProducto
ON productos
INSTEAD OF INSERT
AS
BEGIN
    -- Verificar que no haya campos vacíos o inválidos
    IF EXISTS (SELECT * FROM inserted WHERE precio <= 0)
    BEGIN
        PRINT 'Error: El precio del producto debe ser mayor que 0.';
        ROLLBACK TRANSACTION;
        RETURN;
    END

    IF EXISTS (SELECT * FROM inserted WHERE stock < 0)
    BEGIN
        PRINT 'Error: El stock del producto no puede ser negativo.';
        ROLLBACK TRANSACTION;
        RETURN;
    END

    IF EXISTS (SELECT * FROM inserted WHERE stock < 0)
    BEGIN
        PRINT 'Error: El stock del producto no puede ser negativo.';
        ROLLBACK TRANSACTION;
        RETURN;
    END

    IF EXISTS (SELECT * FROM inserted WHERE nombre = '' OR nombre IS NULL)
    BEGIN
        PRINT 'Error: El nombre del producto no puede estar vacío.';
        ROLLBACK TRANSACTION;
        RETURN;
    END

    -- Si todas las condiciones son válidas, proceder con la inserción de los datos
    INSERT INTO productos (id, codigo, nombre, proveedor, stock, precio)
    SELECT id, codigo, nombre, proveedor, stock, precio
    FROM inserted;

    PRINT 'Producto insertado correctamente.';
END;
```

```
00 %
Messages  Client Statistics
Commands completed successfully.

Completion time: 2024-12-21T11:52:13.0364232-05:00
```

5) Utilizando Script Crear 03 usuarios con nombres de sus compañeros y uno suyo

```sql
INSERT INTO usuarios (id, nombre, correo, pass, rol)
VALUES
    (5, 'Gian Rojas Panduro', 'gian.rojas@email.com', 'password123', 'Asistente'),
    (6, 'Cajahuaringa Samaniego Gerson', 'gerson.cajahuaringa@email.com', 'password123', 'Asistente'),
    (7, 'Luis Ramírez Brayan', 'luis.ramirez@email.com', 'password123', 'Administrador');

SELECT * FROM usuarios;
```

%

Results | Messages | Client Statistics

| id | nombre | correo | pass | rol |
|---|---|---|---|---|
| 1 | Angel | angelADMI2@gmail.com | admin | Administrador |
| 2 | VIDAL | VIDAL15@gmail.com | admin | Asistente |
| 3 | brayan | brayan@gmail.com | 123456 | Administrador |
| 4 | MARIA | MARIA12@gmail.com | 123 | Asistente |
| 5 | Gian Rojas Panduro | gian.rojas@email.com | pass... | Asistente |
| 6 | Cajahuaringa Samaniego Gerson | gerson.cajahuaringa@... | pass... | Asistente |
| 7 | Luis Ramírez Brayan | luis.ramirez@email.com | pass... | Administrador |

6) Utilizando un script, copiar la base de datos (creada anteriormente) y compartir en cada uno de los usuarios

```
-- 1. Crear una copia de la base de datos original
-- Este comando crea una nueva base de datos llamada examenbaseDatos_copy
USE master;
GO

-- Si la base de datos de destino ya existe, se elimina (para evitar errores)
IF EXISTS (SELECT name FROM sys.databases WHERE name = 'examenbaseDatos_copy')
BEGIN
    DROP DATABASE examenbaseDatos_copy;
END
GO

-- Crear la base de datos copia
CREATE DATABASE examenbaseDatos_copy;
GO
```

```
USE examenbaseDatos;
GO

-- Copiar todas las tablas y datos a la nueva base de datos
INSERT INTO examenbaseDatos_copy.dbo.productos
SELECT * FROM examenbaseDatos.dbo.productos;

INSERT INTO examenbaseDatos_copy.dbo.clientes
SELECT * FROM examenbaseDatos.dbo.clientes;

INSERT INTO examenbaseDatos_copy.dbo.config
SELECT * FROM examenbaseDatos.dbo.config;

INSERT INTO examenbaseDatos_copy.dbo.proveedor
SELECT * FROM examenbaseDatos.dbo.proveedor;

INSERT INTO examenbaseDatos_copy.dbo.usuarios
SELECT * FROM examenbaseDatos.dbo.usuarios;

INSERT INTO examenbaseDatos_copy.dbo.ventas
SELECT * FROM examenbaseDatos.dbo.ventas;
GO
```

```
-- 3. Asignar permisos de la base de datos copia a los usuarios
-- Conceder permisos a los usuarios para acceder a la base de datos copia
USE examenbaseDatos_copy;
GO

-- Asignar permisos de lectura y escritura (si es necesario)
GRANT SELECT, INSERT, UPDATE, DELETE TO [Gian Rojas Panduro];
GRANT SELECT, INSERT, UPDATE, DELETE TO [Cajahuaringa Samaniego Gerson];
GRANT SELECT, INSERT, UPDATE, DELETE TO [Luis Ramírez Brayan];
GO
```

7) Utilizando un script, generar una copia de seguridad de la base de datos y compartir a cada uno de los usuarios

```
BACKUP DATABASE examenbaseDatos TO DISK = 'C:\Backup\examenbaseDatos.bak';
GO
```

8) Utilizando un script, encriptar una de las tablas para que no se puedan ver los datos

```
CREATE CERTIFICATE MiCertificado
WITH SUBJECT = 'Certificado de Encriptación de Datos';
GO

CREATE DATABASE ENCRYPTION KEY;
GO

ALTER DATABASE examenbaseDatos
SET ENCRYPTION ON;
GO

CREATE SYMMETRIC KEY MiClaveSimetrica
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE MiCertificado;
GO
```

```
OPEN SYMMETRIC KEY MiClaveSimetrica DECRYPTION BY CERTIFICATE MiCertificado;

UPDATE productos
SET precio = ENCRYPTBYKEY(KEY_GUID('MiClaveSimetrica'), precio);
GO

SELECT
    id,
    codigo,
    nombre,
    proveedor,
    stock,
    CONVERT(DECIMAL(10,2), DECRYPTBYKEY(precio)) AS precio
FROM productos;
GO
```

9) Utilizando un script, aplique la seguridad a nivel de columna, restringiendo el acceso a la columna DNI de la tabla empleado en el usuario con nombre de su compañero

```sql
-- 1. Crear una Vista para ocultar la columna DNI para el usuario 'Gian Rojas Panduro'
CREATE VIEW vw_empleado AS
SELECT
    id,
    nombre,
    telefono,
    direccion
FROM empleado;
GO

-- 2. Revoke acceso a la columna DNI en la tabla original para el usuario 'Gian Rojas Panduro'
REVOKE SELECT ON empleado(DNI) TO [Gian Rojas Panduro];
GO

-- 3. Conceder acceso a la vista para el usuario 'Gian Rojas Panduro'
GRANT SELECT ON vw_empleado TO [Gian Rojas Panduro];
GO
```

10) Utilizando un script, implementé seguridad a nivel de columna restringiendo el acceso a una de las columnas de una tabla.

```sql
-- 1. Crear una Vista para ocultar la columna 'precio' para ciertos usuarios
CREATE VIEW vw_productos AS
SELECT
    id,
    codigo,
    nombre,
    proveedor,
    stock
FROM productos;
GO

-- 2. Revoke acceso a la columna 'precio' en la tabla original para el usuario 'Gian Rojas Panduro'
REVOKE SELECT ON productos(precio) TO [Gian Rojas Panduro];
GO

-- 3. Conceder acceso a la vista para el usuario 'Gian Rojas Panduro'
GRANT SELECT ON vw_productos TO [Gian Rojas Panduro];
GO
```

11) Utilizando un script, realice el cifrado transparente de datos (TDE) para una las tablas.

```sql
-- 1. Crear una clave maestra en la base de datos
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'TuClaveSegura123!';
GO

-- 2. Crear un certificado que se utilizará para cifrar la base de datos
CREATE CERTIFICATE MiCertificadoTDE
WITH SUBJECT = 'Certificado TDE';
GO

-- 3. Crear una clave simétrica para el cifrado de la base de datos
CREATE DATABASE ENCRYPTION KEY
ENCRYPTION BY CERTIFICATE MiCertificadoTDE;
GO

-- 4. Habilitar el cifrado en la base de datos
ALTER DATABASE examenbaseDatos
SET ENCRYPTION ON;
GO
```

12) Utilizando un script, configure el usuario con el nombre de su compañero para otorgar permisos de SELECT, INSERT, UPDATE y DELETE en la base de datos.

```sql
-- 1. Otorgar permisos de SELECT, INSERT, UPDATE y DELETE al usuario 'Gian Rojas Panduro'
GRANT SELECT, INSERT, UPDATE, DELETE ON DATABASE::examenbaseDatos TO [Gian Rojas Panduro];
GO
```

13) Utilizando un script, configure la auditoría para el seguimiento y registro de acciones en la base de datos

```
-- 1. Crear un objeto de auditoría (archivo de auditoría)
CREATE SERVER AUDIT MiAuditoria
TO FILE (FILEPATH = 'C:\Auditorias\')
WITH (ON_FAILURE = CONTINUE);
GO

-- 2. Crear una especificación de auditoría para auditar acciones de base de datos
CREATE SERVER AUDIT SPECIFICATION MiEspecificacionAuditoria
FOR SERVER AUDIT MiAuditoria
ADD (DATABASE_OBJECT_PERMISSION_CHANGE_GROUP),
ADD (DATABASE_ROLE_MEMBER_CHANGE_GROUP),
ADD (SCHEMA_OBJECT_ACCESS_GROUP)
WITH (STATE = ON);
GO

-- 3. Activar la auditoría
ALTER SERVER AUDIT MiAuditoria WITH (STATE = ON);
GO
```

14) Utilizando un script, configure de la memoria y el disco duro

```
-- 1. Configurar el uso máximo de memoria para SQL Server (en MB)
EXEC sp_configure 'max server memory (MB)', 4096;  -- 4GB
RECONFIGURE;
GO

-- 2. Configurar el uso mínimo de memoria para SQL Server (en MB)
EXEC sp_configure 'min server memory (MB)', 1024;  -- 1GB
RECONFIGURE;
GO
```

```
-- 1. Aumentar el tamaño del archivo de base de datos (archivo de datos)
ALTER DATABASE examenbaseDatos
MODIFY FILE (NAME = 'examenbaseDatos_data', SIZE = 5GB);
GO

-- 2. Aumentar el tamaño del archivo de registro de la base de datos
ALTER DATABASE examenbaseDatos
MODIFY FILE (NAME = 'examenbaseDatos_log', SIZE = 2GB);
GO
```

15) Utilizando un script, genere una copia de seguridad de la base de datos

```
-- 1. Realizar una copia de seguridad completa de la base de datos
BACKUP DATABASE examenbaseDatos
TO DISK = 'C:\Backup\examenbaseDatos.bak'
WITH FORMAT, MEDIANAME = 'MiBackup', NAME = 'Copia de seguridad completa';
GO
```

16) Utilizando un script, genere la restauración de la base de datos

```
-- 1. Restaurar la base de datos desde el archivo de copia de seguridad
RESTORE DATABASE examenbaseDatos
FROM DISK = 'C:\Backup\examenbaseDatos.bak'
WITH REPLACE;
GO
```

## 17) Utilizando un script, cree un espejo de la base de datos

```
-- En el servidor primario
ALTER DATABASE examenbaseDatos SET PARTNER = 'TCP://ServidorSecundario:5022';
GO

-- En el servidor secundario
ALTER DATABASE examenbaseDatos SET PARTNER = 'TCP://ServidorPrimario:5022';
GO
-- En el servidor primario
ALTER DATABASE examenbaseDatos
SET PARTNER SAFETY FULL;
GO

-- En el servidor secundario
ALTER DATABASE examenbaseDatos
SET PARTNER SAFETY FULL;
GO
```

## 18) Utilizando un script, realice la replicación de bases de datos

```
-- 1. Configurar el servidor publicador
EXEC sp_replicationdboption
  @dbname = 'examenbaseDatos',
  @optname = 'publish',
  @value = 'true';
GO

-- 2. Configurar la publicación
EXEC sp_addpublication
  @publication = 'ExamenPublication',
  @description = 'Publicación de la base de datos examenbaseDatos',
  @pubtype = 'transational',
  @status = 'active';
GO
-- 3. Configurar el servidor suscriptor
EXEC sp_addsubscription
  @publication = 'ExamenPublication',
  @subscriber = 'ServidorSecundario',
  @destination_db = 'examenbaseDatos',
  @subscription_type = 'push',
  @sync_type = 'automatic';
GO
```

## 19) Explique que es Always On Availability Groups

Always On Availability Groups es una característica de SQL Server que asegura la alta disponibilidad de las bases de datos y la recuperación ante desastres. Permite replicar bases de datos entre varias instancias de SQL Server, de modo que si una base de datos se cae, se puede cambiar automáticamente a una copia secundaria sin perder datos. Además, las réplicas secundarias pueden usarse para consultas, lo que mejora el rendimiento.

## 20) Explique que es Log Shipping

Log Shipping es un método que permite hacer copias de seguridad de los registros de transacciones de una base de datos primaria y restaurarlos en una secundaria. Sin embargo, no se realiza de forma automática, sino manual. Puede haber un pequeño retraso entre los datos en la base de datos primaria y secundaria, ya que depende de los intervalos de copia y restauración de los registros.