

UNIVERSIDAD PERUANA LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y



Base de Datos II

AUTOR

CONDOR HUAMAN ERICK

DOCENTE

RAUL FERNANDEZ, Bejarano

CICLO

V

HUANCAYO – PERÚ – 2025

Practica-semana 13

Practica Semana 16

1. Enunciado del Proyecto

Título: Implementación y Carga Automatizada de un Sistema de Gestión de Ventas Corporativas (EnterpriseSales).

Objetivo: Diseñar e implementar una base de datos relacional para el control de inventarios y ventas, asegurando la integridad referencial. Se requiere automatizar la generación de datos de prueba mediante estructuras de control (bucles) y validar la consistencia de la información mediante reportes analíticos de rendimiento y stock.

2. Script Completo (Consolidado)

SQL

```
USE master;
```

```
GO
```

```
-- 1. ELIMINACIÓN FORZADA (Cierra conexiones activas antes de borrar)
```

```
IF EXISTS (SELECT name FROM sys.databases WHERE name = 'EnterpriseSales') BEGIN
```

```
    -- Cambia a SINGLE_USER para desconectar a otros usuarios y borra
```

```
    ALTER DATABASE EnterpriseSales SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
```

```
    DROP DATABASE EnterpriseSales;
```

```
END
```

```
GO
```

```
-- 2. CREACIÓN Y CONFIGURACIÓN
```

```
CREATE DATABASE EnterpriseSales;
```

```
GO
```

```
ALTER DATABASE EnterpriseSales SET RECOVERY FULL; GO
```

```
USE EnterpriseSales;
```

```
GO
```

```
-- 3. ESTRUCTURA (Aseguramos que Ciudad esté presente para evitar el error de columna)
```

```
CREATE TABLE Clientes (
```

```
    ClienteID INT PRIMARY KEY IDENTITY(1,1),
```

```
    Nombre VARCHAR(100) NOT NULL,
```

```
    Email VARCHAR(100) UNIQUE,
```

```
    Ciudad VARCHAR(50), -- Columna necesaria para los inserts de abajo
```

```
    FechaRegistro DATETIME DEFAULT GETDATE()
```

```
);
```

```
CREATE TABLE Categorias (
```

```
    CategoriaID INT PRIMARY KEY IDENTITY(1,1),           NombreCategoria VARCHAR(50) NOT NULL
```

```
);
```

```
CREATE TABLE Productos (
```

```
    ProductoID INT PRIMARY KEY IDENTITY(1,1),
```

```
    NombreProducto VARCHAR(100) NOT NULL,
```

```
    CategoriaID INT FOREIGN KEY REFERENCES Categorias(CategoriaID),
```

```

    PrecioUnitario DECIMAL(18,2) NOT NULL,
    StockActual INT DEFAULT 0
);

CREATE TABLE Ventas (
    VentaID INT PRIMARY KEY IDENTITY(1,1),
    ClienteID INT FOREIGN KEY REFERENCES Clientes(ClienteID),
    FechaVenta DATETIME DEFAULT GETDATE(),
    Total DECIMAL(18,2) NOT NULL,
    Estado VARCHAR(20) DEFAULT 'Completado'
);

CREATE TABLE DetalleVentas (
    DetalleID INT PRIMARY KEY IDENTITY(1,1),
    VentaID INT FOREIGN KEY REFERENCES Ventas(VentaID),
    ProductoID INT FOREIGN KEY REFERENCES Productos(ProductoID),
    Cantidad INT NOT NULL,
    PrecioAplicado DECIMAL(18,2) NOT NULL
);
GO

-- 4. CARGA DE DATOS
INSERT INTO Categorias (NombreCategoria) VALUES ('Electrónica'), ('Computación');
INSERT INTO Clientes (Nombre, Email, Ciudad) VALUES ('Juan Pérez',
'juan.perez@email.com', 'Lima'),
('Corporación X', 'contacto@corpx.com', 'Trujillo');

INSERT INTO Productos (NombreProducto, CategoriaID, PrecioUnitario, StockActual)
VALUES
('Laptop Pro 15', 2, 1200.00, 50),
('Mouse Ergonómico', 1, 25.00, 200);

INSERT INTO Ventas (ClienteID, Total) VALUES (1, 1225.00);
INSERT INTO DetalleVentas (VentaID, ProductoID, Cantidad, PrecioAplicado) VALUES
(1, 1, 1, 1200.00),
(1, 2, 1, 25.00);
GO

-- 5. VERIFICACIÓN FINAL
SELECT * FROM Clientes;
SELECT * FROM Ventas

USE EnterpriseSales;
GO

```

```

-- 1. Insertar 50 Categorías
DECLARE @i INT = 1;
WHILE @i <= 50
BEGIN
    INSERT INTO Categorias (NombreCategoria)
    VALUES ('Categoría Especializada ' + CAST(@i AS VARCHAR(10)));
    SET @i = @i + 1;
END;

-- 2. Insertar 50 Clientes
SET @i = 1;
WHILE @i <= 50
BEGIN
    INSERT INTO Clientes (Nombre, Email, Ciudad, FechaRegistro)
    VALUES (
        'Cliente Prueba ' + CAST(@i AS VARCHAR(10)),
        'usuario' + CAST(@i AS VARCHAR(10)) + '@empresa.com',
        CASE WHEN @i % 2 = 0 THEN 'Lima' ELSE 'Provincias' END,
        GETDATE()
    );
    SET @i = @i + 1;
END;

-- 3. Insertar 50 Productos
SET @i = 1;
WHILE @i <= 50
BEGIN
    INSERT INTO Productos (NombreProducto, CategoriaID, PrecioUnitario,
StockActual)
    VALUES (
        'Producto Industrial ' + CAST(@i AS VARCHAR(10)),
        (SELECT TOP 1 CategoriaID FROM Categorias ORDER BY NEWID()), -- Asigna
categoría aleatoria
        RAND() * 500 + 10, -- Precio aleatorio entre 10 y 510
        CAST(RAND() * 100 AS INT)
    );
    SET @i = @i + 1;
END;

-- 4. Insertar 50 Ventas
SET @i = 1;
WHILE @i <= 50
BEGIN
    INSERT INTO Ventas (ClienteID, Total, FechaVenta)
    VALUES (
        (SELECT TOP 1 ClienteID FROM Clientes ORDER BY NEWID()), -- Cliente aleatorio
        0, -- Se actualizará después con el detalle
        DATEADD(DAY, -@i, GETDATE()) -- Ventas en diferentes días
    );

```

```

        SET @i = @i + 1;
    END;

-- 5. Insertar 50 DetalleVentas y actualizar Totales
SET @i = 1;
WHILE @i <= 50
BEGIN
    DECLARE @VentaID INT = (SELECT TOP 1 VentaID FROM Ventas ORDER BY NEWID());
    DECLARE @ProdID INT = (SELECT TOP 1 ProductoID FROM Productos ORDER BY NEWID());
    DECLARE @Precio DECIMAL(18,2) = (SELECT PrecioUnitario FROM Productos WHERE
    ProductoID = @ProdID);
    DECLARE @Cant INT = CAST(RAND() * 5 + 1 AS INT);

    INSERT INTO DetalleVentas (VentaID, ProductoID, Cantidad, PrecioAplicado)
    VALUES (@VentaID, @ProdID, @Cant, @Precio);
    -- Actualizar el total de la venta sumando el nuevo detalle
    UPDATE Ventas SET Total = Total + (@Precio * @Cant) WHERE VentaID = @VentaID;
    SET @i = @i + 1;
END;
GO

-- VERIFICACIÓN: Contar registros generados
SELECT 'Clientes' as Tabla, COUNT(*) as Total FROM Clientes
UNION SELECT 'Categorias', COUNT(*) FROM Categorias
UNION SELECT 'Productos', COUNT(*) FROM Productos
UNION SELECT 'Ventas', COUNT(*) FROM Ventas
UNION SELECT 'DetalleVentas', COUNT(*) FROM DetalleVentas;

```

--codigo de verificacion de datos de base de datos

```

USE EnterpriseSales;
GO

-- 1. REPORTE GENERAL DE VENTAS (Muestra el flujo completo del negocio)
-- Este reporte es ideal para verificar la consistencia de datos tras un Failover.
SELECT
    V.VentaID,
    C.Nombre AS Cliente,
    CAT.NombreCategoria AS Categoria,
    P.NombreProducto,
    DV.Cantidad,
    DV.PrecioAplicado,

```

```

(DV.Cantidad * DV.PrecioAplicado) AS Subtotal,
V.FechaVenta
FROM Ventas V
INNER JOIN Clientes C ON V.ClienteID = C.ClienteID
INNER JOIN DetalleVentas DV ON V.VentaID = DV.VentaID
INNER JOIN Productos P ON DV.ProductoID = P.ProductoID
INNER JOIN Categorias CAT ON P.CategoríaID = CAT.CategoríaID
ORDER BY V.VentaID DESC;

-- 2. RESUMEN DE REGISTROS POR TABLA
-- Útil para confirmar que los 50 registros por tabla se insertaron correctamente.
SELECT 'Clientes' AS Tabla, COUNT(*) AS Cantidad FROM Clientes UNION ALL
SELECT 'Categorías', COUNT(*) FROM Categorias
UNION ALL
SELECT 'Productos', COUNT(*) FROM Productos
UNION ALL
SELECT 'Ventas', COUNT(*) FROM Ventas
UNION ALL
SELECT 'Detalle Ventas', COUNT(*) FROM DetalleVentas;
-- 3. PRODUCTOS CON BAJO STOCK
-- Simula un reporte administrativo de inventario.
SELECT NombreProducto, StockActual
FROM Productos
WHERE StockActual < 10
ORDER BY StockActual ASC;

-- 4. TOP 5 CLIENTES QUE MÁS HAN COMPRADO
SELECT TOP 5
C.Nombre,
SUM(V.Total) AS InversionTotal
FROM Clientes C
JOIN Ventas V ON C.ClienteID = V.ClienteID
GROUP BY C.Nombre
ORDER BY InversionTotal DESC;

```

```

SQLQuery1.sql - DESKTOP-B9K096N.EnterpriseSales (DESKTOP-B9K096N\User (67)) - Microsoft SQL Server Management Studio

Explorador de objetos
Conectar a > SQL Server - DESKTOP-B9K096N (16.0.1000.6 de SQL Server - DESKTOP-B9K096N) > EnterpriseSales

SQLQuery1.sql - D_B9K096N\User (67)* - x
[...]
-- 2. CCREACIÓN Y CONFIGURACIÓN
CREATE DATABASE EnterpriseSales;
GO

ALTER DATABASE EnterpriseSales SET RECOVERY FULL;
GO

USE EnterpriseSales;
GO

-- 3. ESTRUCTURA (Aseguramos que Ciudad esté presente para evitar el error de columna)
CREATE TABLE Clientes (
    ClienteID INT PRIMARY KEY IDENTITY(1,1),
    Nombre VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Ciudad VARCHAR(50) -- Columna necesaria para los inserts de abajo
    FechaRegistro DATETIME DEFAULT GETDATE()
);

-- Consulta ejecutada correctamente.

```

ClienteID	Nombre	Email	Ciudad	FechaRegistro
1	Juan Pérez	juan.perez@email.com	Lima	2025-12-18 09:34:12.620
2	Compradora X	contacto@corpx.com	Traillo	2025-12-18 09:34:12.620

VentaID	ClienteID	FechaVenta	Total	Estado
1	1	2025-12-18 09:34:12.620	1225.00	Completo

Table	Total
Categorías	52
Clientes	52
DetalleVentas	52
Productos	52
Ventas	51

DESCRIPCIÓN DE LOS DATOS EN LA TABLA VENTAS:

VentaID	Cliente	Categoría	NombreProducto	Cantidad	PrecioUnitario	Subtotal	FechaVenta	
1	49	Cliente Prueba 34	Categoría Especializada 20	Producto Industrial 17	3	804.82	2414.76	2025-10-30 09:34:12.743
2	49	Cliente Prueba 34	Computación	Laptop Pro 15	1	1200.00	1200.00	2025-10-31 09:34:12.743
3	48	Cliente Prueba 30	Categoría Especializada 32	Producto Industrial 46	1	449.79	449.79	2025-11-01 09:34:12.743
4	48	Cliente Prueba 30	Categoría Especializada 11	Producto Industrial 50	5	380.83	1904.15	2025-11-01 09:34:12.743
5	49	Cliente Prueba 49	Categoría Especializada 99	Producto Industrial 97	9	777.00	7092.00	2025-11-01 09:34:12.743

3. Explicación Técnica

Eliminación Forzada: El uso de `SET SINGLE_USER WITH ROLLBACK IMMEDIATE` es una técnica avanzada para garantizar que la base de datos se elimine incluso si hay administradores o aplicaciones conectadas, evitando el error de "Database in use".

Modelo Relacional: Se implementan llaves primarias (`PRIMARY KEY`) e incrementales (`IDENTITY`) para asegurar la unicidad. Las llaves foráneas (`FOREIGN KEY`) garantizan que no existan ventas de productos que no existen o para clientes inexistentes.

Lógica de Negocio en SQL: Se utiliza la función `NEWID()` dentro de un `ORDER BY` para seleccionar registros aleatorios de otras tablas durante la carga de datos, simulando un comportamiento de compra real.

Integridad de Cálculo: El script incluye un `UPDATE` posterior a la inserción en `DetalleVentas` para asegurar que el campo `Total` de la tabla `Ventas` coincida con la suma de sus detalles.

4. Justificación

Este script se justifica por las siguientes necesidades empresariales:

Entornos de Prueba (Sandboxing): Permite crear en segundos un entorno con datos realistas para que los desarrolladores prueben aplicaciones sin usar datos reales y sensibles de clientes.

Verificación de Alta Disponibilidad: Al configurar el `RECOVERY FULL`, el script está preparado para ser utilizado en escenarios de **Log Shipping** o **Always On**, permitiendo verificar que los datos se replican correctamente tras un failover.

Toma de Decisiones: Los reportes finales (Top clientes y bajo stock) justifican la estructura de la base de datos al transformar datos crudos en información estratégica para reabastecimiento de almacén y fidelización de clientes.

```

USE EnterpriseSales;
GO

-- 1. REPORTE GENERAL DE VENTAS (Muestra el flujo completo del negocio)
-- Este reporte es ideal para verificar la consistencia de datos tras un Failover.
SELECT
    V.VentaID,
    C.Nombre AS Cliente,
    CAT.NombreCategoria AS Categoria,
    P.NombreProducto,
    DV.PrecioAplicado,
    DV.PrecioAplicado,
    (DV.Cantidad * DV.PrecioAplicado) AS Subtotal,
    V.FechaVenta
FROM Ventas V
INNER JOIN Clientes C ON V.ClienteID = C.ClienteID
INNER JOIN DetalleVentas DV ON V.VentaID = DV.VentaID

```

Results:

	Cantidad	
1	Clientes	52
2	Categorías	52
3	Productos	52
4	Ventas	51
5	Detalle Ventas	52

NombreProducto	StockActual
1 Producto Industrial 33	0
2 Producto Industrial 37	2
3 Producto Industrial 32	2
4 Producto Industrial 41	3

Consulta ejecutada correctamente.

2. Implementación de Plan de Respaldo y Recuperación (DR)

Enunciado: Configurar una estrategia que garantice un RPO (Objetivo de Punto de Recuperación) mínimo.

Script de Backup Integral

SQL

```

-- 4. SOLUCIÓN AL ERROR DE RUTA: Asegúrate de crear la carpeta C:\Backups antes
-- Primero realizamos el Backup Completo (Obligatorio antes que cualquier otro) BACKUP
DATABASE EnterpriseSales
TO DISK = 'C:\Backups\EnterpriseSales_Full.bak'
WITH FORMAT, NAME = 'Full Backup EnterpriseSales'; GO

-- Ahora sí permitiría el diferencial o el de log
BACKUP DATABASE EnterpriseSales
TO DISK = 'C:\Backups\EnterpriseSales_Diff.bak' WITH DIFFERENTIAL;
GO

BACKUP LOG EnterpriseSales
TO DISK = 'C:\Backups\EnterpriseSales_Log.trn';
GO

```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. On the left, the Object Explorer displays the database structure for 'DESKTOP-B9K096N' (16.0.1000.6 for SQL Server - DES). The 'EnterpriseSales' database is selected. The 'Script' tab is active, showing T-SQL code for backing up the database.

```
-- 4. SOLUCION AL ERROR DE RUTA: Asegurate de crear la carpeta C:\backups antes
--   Primero realizamos el Backup Completo (Obligatorio antes que cualquier otro)
BACKUP DATABASE EnterpriseSales
TO DISK = "C:\Backups\EnterpriseSales_Full.bak"
WITH FORMAT, NAME = 'Full Backup EnterpriseSales',
GO

-- Ahora si permitiria el diferencial o el de Log
BACKUP DATABASE EnterpriseSales
TO DISK = "C:\Backups\EnterpriseSales_DIFF.bak" WITH DIFFERENTIAL,
GO

BACKUP LOG EnterpriseSales
TO DISK = "C:\Backups\EnterpriseSales_Log.trn";
GO
```

Below the script, the 'Messages' pane shows the execution results:

```
Procesadas 531 páginas para archivo 'EnterpriseSales', file 'EnterpriseSales', en página 1.
Procesadas 118 páginas para archivo 'EnterpriseSales_log', file 'EnterpriseSales_log', en página 1.
BACKUP DATABASE EnterpriseSales procesó 550 páginas en 0.123 segundos (4.757 KB/sec).
Procesadas 118 páginas para archivo 'EnterpriseSales', file 'EnterpriseSales', en página 1.
Procesadas 118 páginas para archivo 'EnterpriseSales_log', file 'EnterpriseSales_log', en página 1.
BACKUP DATABASE EnterpriseSales con DIFERENCIAL procesó 136 páginas en 0.111 segundos (4.928 KB/sec).
Procesadas 118 páginas para archivo 'EnterpriseSales_log', file 'EnterpriseSales_log', en página 1.
BACKUP LOG EnterpriseSales procesó 1 páginas en 0.000 segundos (0.000 KB/sec).

Hora de finalización: 2020-12-18T05:08:31.211244+00:00
```

3. Configuración de Alta Disponibilidad (HA)

Para cumplir con el objetivo de **AlwaysOn Availability Groups** mencionado en tu temario, se requiere una arquitectura de red, pero a nivel de SQL Scripting, el monitoreo de la sincronización se justifica así:

Explicación de las Tecnologías:

Log Shipping: Ideal para recuperación ante desastres a bajo costo. Envía archivos de log de un servidor a otro.

AlwaysOn: Es la solución "Premium". Permite tener múltiples bases de datos secundarias que pueden usarse incluso para "Solo Lectura", optimizando el rendimiento del servidor principal.

4. Simulación de Failover y Restauración

Escenario: El servidor principal falla. Debemos restaurar la operación en el servidor de contingencia.

Script de Recuperación (Manual)

SQL

-- 1. LIMPIEZA TOTAL Y CIERRE DE CONEXIONES

USE master;

GO

```
IF EXISTS (SELECT name FROM sys.databases WHERE name = 'EnterpriseSales') BEGIN
    -- Expulsa a cualquier usuario conectado para evitar el error "in use"
    ALTER DATABASE EnterpriseSales SET SINGLE_USER WITH ROLLBACK IMMEDIATE;      DROP
DATABASE EnterpriseSales;
END
GO
```

-- 2. CREATIÓN DE BASE DE DATOS Y CONFIGURACIÓN HA

CREATE DATABASE EnterpriseSales;

GO

```
ALTER DATABASE EnterpriseSales SET RECOVERY FULL; -- Obligatorio para Alta  
Disponibilidad
```

```
GO
```

```
USE EnterpriseSales;
GO
```

```
-- 3. ESTRUCTURA EMPRESARIAL
```

```
CREATE TABLE Categorias (CategoriaID INT PRIMARY KEY IDENTITY(1,1),
NombreCategoria VARCHAR(50));
CREATE TABLE Clientes (ClienteID INT PRIMARY KEY IDENTITY(1,1), Nombre
VARCHAR(100), Email VARCHAR(100), Ciudad VARCHAR(50));
CREATE TABLE Productos (ProductoID INT PRIMARY KEY IDENTITY(1,1), NombreProducto
VARCHAR(100), CategoriaID INT FOREIGN KEY REFERENCES Categorias(CategoriaID),
PrecioUnitario DECIMAL(18,2), StockActual INT);
CREATE TABLE Ventas (VentaID INT PRIMARY KEY IDENTITY(1,1), ClienteID INT FOREIGN
KEY REFERENCES Clientes(ClienteID), FechaVenta DATETIME DEFAULT GETDATE(), Total
DECIMAL(18,2));
CREATE TABLE DetalleVentas (DetalleID INT PRIMARY KEY IDENTITY(1,1), VentaID INT
FOREIGN KEY REFERENCES Ventas(VentaID), ProductoID INT FOREIGN KEY REFERENCES
Productos(ProductoID), Cantidad INT, PrecioAplicado DECIMAL(18,2));
GO
```

```
-- 4. CARGA MASIVA DE 50 REGISTROS (SIMULACIÓN EMPRESARIAL)
```

```
DECLARE @i INT = 1;
WHILE @i <= 50
BEGIN
    INSERT INTO Categorias VALUES ('Categoria ' + CAST(@i AS VARCHAR(5)));
    INSERT INTO Clientes VALUES ('Cliente '+CAST(@i AS VARCHAR(5)),
'user'+CAST(@i AS VARCHAR(5))+ '@mail.com', 'Ciudad '+CAST(@i AS VARCHAR(5)));
    INSERT INTO Productos VALUES ('Producto '+CAST(@i AS VARCHAR(5)), (SELECT TOP
1 CategoriaID FROM Categorias ORDER BY NEWID()), RAND()*100, 50);
    INSERT INTO Ventas (ClienteID, Total) VALUES ((SELECT TOP 1 ClienteID FROM
Clientes ORDER BY NEWID()), 0);

    DECLARE @vID INT = SCOPE_IDENTITY();
    DECLARE @pID INT = (SELECT TOP 1 ProductoID FROM Productos ORDER BY NEWID());
    INSERT INTO DetalleVentas VALUES (@vID, @pID, 2, 50.00);
    UPDATE Ventas SET Total = 100.00 WHERE VentaID = @vID;
    SET @i = @i + 1;
END
GO
```

```
-- 5. RESPALDOS (SOLUCIÓN AL ERROR DE RUTA)
```

```
-- Nota: Asegúrate de haber creado C:\Backups en tu Windows
BACKUP DATABASE EnterpriseSales TO DISK = 'C:\Backups\EnterpriseSales_Full.bak'
WITH FORMAT, NAME = 'Full Backup';
BACKUP LOG EnterpriseSales TO DISK = 'C:\Backups\EnterpriseSales_Log.trn' WITH NAME =
'Log Backup';
GO
```

-- 6. VERIFICACIÓN FINAL

```
SELECT 'EXITO' AS Estado, COUNT(*) AS VentasRegistradas FROM Ventas;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure of 'EnterpriseSales'. In the center, a query window titled 'SQLQuery1.sql - DESKTOP-B9K096N.EnterpriseSales (DESKTOP-B9K096N\User (67)) - Microsoft SQL Server Management Studio' contains the following T-SQL script:

```
-- 1. LIMPIEZA TOTAL Y CIERRE DE CONEXIONES
USE master;
GO

IF EXISTS (SELECT name FROM sys.databases WHERE name = 'EnterpriseSales')
BEGIN
    -- Evita que el script sea cancelado por un usuario conectado para volver el error "In use"
    ALTER DATABASE EnterpriseSales SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE EnterpriseSales;
END
GO

-- 2. CREACION DE BASE DE DATOS Y CONFIGURACION HA
CREATE DATABASE EnterpriseSales;
GO

ALTER DATABASE EnterpriseSales SET RECOVERY FULL; -- Obligatorio para Alta Disponibilidad
GO
```

The results pane shows a single row of data:

Estado	VentasRegistradas
EXITO	50

At the bottom, a status bar indicates: Consulta ejecutada correctamente.

Justificación del Procedimiento

Se utiliza NORECOVERY para permitir la aplicación de todos los cambios pendientes antes de abrir la base de datos al público.

Esto garantiza que **no haya pérdida de datos** entre el último backup y el momento de la caída.

Código Optimizado: Índice Non-Clustered Compuesto

En lugar de indexar solo la fecha, vamos a incluir el ClienteID y el Estado. Esto es mucho más útil para reportes reales de ventas por cliente y periodo.

SQL

```
USE EnterpriseSales;
```

```
GO
```

```
-- 1. Eliminamos el índice anterior si existe para evitar duplicidad
IF EXISTS (SELECT name FROM sys.indices WHERE name =
'IX_Ventas_ReporteGerencial')
    DROP INDEX IX_Ventas_ReporteGerencial ON Ventas;
```

```
GO
```

```
-- 2. Creamos un índice optimizado para consultas de alta frecuencia
```

```
CREATE NONCLUSTERED INDEX IX_Ventas_ReporteGerencial
ON Ventas (FechaVenta, ClienteID) -- Columnas de búsqueda (Key Columns)
INCLUDE (Total, Estado) -- Columnas de cobertura
(Covering Columns)
WITH (FILLFACTOR = 80, ONLINE = ON);
GO
```

```

-- 6. VERIFICACIÓN FINAL
--SELECT 'EXITO' AS Estado, COUNT(*) AS VentasRegistradas FROM Ventas;

-- tercera parte de la actividad de la semana 16

-- 3. LIMPIEZA TOTAL (Cierra conexiones para evitar errores de Captura 1 y 3)
USE master;
GO

IF EXISTS (SELECT name FROM sys.databases WHERE name = 'EnterpriseSales')
BEGIN
    ALTER DATABASE EnterpriseSales SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE EnterpriseSales;
END
GO

```

90 % 4 Messages

Processed 352 pages for database 'EnterpriseSales', file 'EnterpriseSales' on file 1.
Processed 2 pages for database 'EnterpriseSales', file 'EnterpriseSales_log' on file 1.
ALTER DATABASE EnterpriseSales SET SINGLE_USER WITH ROLLBACK IMMEDIATE
Processed 2 pages for database 'EnterpriseSales', file 'EnterpriseSales_log' on file 1.
BACKUP LOG successfully processed 2 pages in 0:000 seconds (7.712 MB/sec).

Hora de finalización: 2025-12-18T05:46:14.7840000+01:00

Consulta ejecutada correctamente.

¿Por qué esta versión es mejor? (Justificación para tu Trabajo)

Índice de Cobertura (Covering Index): Al incluir (`INCLUDE`) las columnas `Total` y `Estado`, el motor de SQL Server puede responder a la consulta directamente desde el índice sin tener que ir a buscar datos a la tabla principal (operación conocida como Key Lookup).

FILLFACTOR = 80: Esta propiedad deja un 20% de espacio libre en las páginas del índice. En una base de datos con 50 o más registros que crecen constantemente, esto evita la fragmentación del disco y reduce la carga en el `Log Shipping` o `AlwaysOn`, ya que hay menos divisiones de página (Page Splits) que replicar.

ONLINE = ON: Permite que la base de datos siga operativa mientras se crea el índice. Esto es un requisito fundamental en entornos de Alta Disponibilidad (HA) donde el tiempo de inactividad debe ser cero.

Ejemplo de Consulta que se beneficia de este índice

Con este índice, la siguiente consulta (típica de un tablero de control empresarial) será instantánea:

SQL

```

SELECT ClienteID, SUM(Total) as VentaTotal
FROM Ventas
WHERE FechaVenta BETWEEN '2025-01-01' AND '2025-12-31'
AND Estado = 'Completado'
GROUP BY ClienteID;

```

Monitoreo de Fragmentación (Tema 3 del Sílabo)

Para tu entrega de la Semana 16, puedes agregar este script de monitoreo para justificar la administración del índice:

SQL

```

SELECT
    index_id, name, avg_fragmentation_in_percent
FROM sys.dm_db_index_physical_stats (DB_ID('EnterpriseSales'),
OBJECT_ID('Ventas'), NULL, NULL, NULL) AS a
JOIN sys.indexes AS b ON a.object_id = b.object_id AND a.index_id =

```

b.index_id;

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left is the Object Explorer pane, which lists several databases including 'EnterpriseSales'. In the center is a query window titled 'SQLQuery1.sql - DESKTOP-B9K096N\User (67)*'. The query script contains T-SQL commands for backing up the 'EnterpriseSales' database and its log, and a SELECT statement to calculate total sales by client. Below the script is a results pane showing index statistics for the 'Ventas' table. At the bottom, a status bar indicates the query was executed successfully.

```
-- 5. BACKUPS (Corrigiendo Captura 2 - Asegúrate que C:\Backups existe)
BACKUP DATABASE EnterpriseSales
TO DISK = 'C:\Backups\EnterpriseSales_Full.bak' WITH FORMAT;
BACKUP LOG EnterpriseSales
TO DISK = 'C:\Backups\EnterpriseSales_Log.trn' WITH FORMAT;
GO

--SELECT ClienteID, SUM(Total) as VentaTotal
FROM Ventas
WHERE FechaVenta BETWEEN '2025-01-01' AND '2025-12-31'
AND Estado = 'Completado'
GROUP BY ClienteID;

-- 1. LIMPIEZA Y CIERRE DE CONEXIONES (Corrige errores de Captura 2 y 4)

-- 2. ESTADÍSTICAS DE LOS ÍNDICES
SELECT index_id, name, avg_fragmentation_in_percent
FROM sys.dm_db_index_physical_stats(DB_ID('EnterpriseSales'), 1, 1, 1, 'DETAILED')
```

index_id	name	avg_fragmentation_in_percent
1	PK_Ventas_5B41514C15C19F0B	0
2	IX_Ventas_ReporteGerencial	0

Consulta ejecutada correctamente.

Consultas de Análisis Empresarial (Queries)

Copia este código al final de tu script para generar los reportes de rendimiento:

SQL

```
USE EnterpriseSales;
GO
```

```
-- 1. REPORTE GERENCIAL DE VENTAS (Uso de JOINS y el Índice
Optimizado)
-- Este reporte muestra el detalle de qué cliente compró qué producto y
en qué categoría.
SELECT
    V.VentaID,
    C.Nombre AS Cliente,
    CAT.NombreCategoria AS Categoria,
    P.NombreProducto,
    DV.Cantidad,
    DV.PrecioAplicado,
    (DV.Cantidad * DV.PrecioAplicado) AS Subtotal,
    V.FechaVenta,
    V.Estado
FROM Ventas V
INNER JOIN Clientes C ON V.ClienteID = C.ClienteID
INNER JOIN DetalleVentas DV ON V.VentaID = DV.VentaID
INNER JOIN Productos P ON DV.ProductoID = P.ProductoID
INNER JOIN Categorias CAT ON P.CategoriaID = CAT.CategoriaID
WHERE V.Estado = 'Completado'
ORDER BY V.FechaVenta DESC;
```

```

-- 2. TOP 5 CLIENTES CON MAYOR VOLUMEN DE COMPRA
-- Identifica a los clientes más importantes para la empresa.
SELECT TOP 5
    C.Nombre,
    C.Ciudad,
    COUNT(V.VentaID) AS TotalTransacciones,
    SUM(V.Total) AS InversionTotal
FROM Clientes C
JOIN Ventas V ON C.ClienteID = V.ClienteID
GROUP BY C.Nombre, C.Ciudad
ORDER BY InversionTotal DESC;

-- 3. RENDIMIENTO POR CATEGORÍA DE PRODUCTO -- Muestra qué categorías generan más ingresos.
SELECT
    CAT.NombreCategoria,
    COUNT(DV.ProductoID) AS ProductosVendidos,
    SUM(DV.Cantidad * DV.PrecioAplicado) AS TotalIngresos
FROM Categorias CAT
LEFT JOIN Productos P ON CAT.CategoríaID = P.CategoríaID
LEFT JOIN DetalleVentas DV ON P.ProductoID = DV.ProductoID
GROUP BY CAT.NombreCategoria
ORDER BY TotalIngresos DESC;

-- 4. CONTROL DE INVENTARIO (PRODUCTOS CRÍTICOS)
-- Lista productos que tienen menos de 10 unidades en stock.
SELECT
    NombreProducto,
    StockActual,
    PrecioUnitario
FROM Productos
WHERE StockActual < 10
ORDER BY StockActual ASC;

-- 5. RESUMEN DE INTEGRIDAD DE LA BASE DE DATOS
-- Consulta técnica para confirmar que los datos se cargaron correctamente por tabla.
SELECT 'Clientes' AS Tabla, COUNT(*) AS Cantidad FROM Clientes
UNION ALL
SELECT 'Categorías', COUNT(*) FROM Categorias
UNION ALL
SELECT 'Productos', COUNT(*) FROM Productos
UNION ALL
SELECT 'Ventas', COUNT(*) FROM Ventas
UNION ALL
SELECT 'Detalle Ventas', COUNT(*) FROM DetalleVentas;

```

Explicación y Justificación Técnica

Optimización de Consultas: La consulta #1 está diseñada para utilizar el índice IX_Ventas_ReporteGerencial que creamos anteriormente. Al filtrar por Estado y FechaVenta, el motor de SQL Server no necesita leer toda la tabla, lo que garantiza velocidad incluso con miles de datos.

Administración de Disponibilidad: En un escenario de *AlwaysOn*, estas consultas de reporte se ejecutarían en el **Nodo Secundario (Read-Only)**, liberando al servidor principal de la carga de procesamiento.

Consistencia de Datos: El reporte #5 es fundamental tras un proceso de *Failover o Restauración*. Sirve para verificar rápidamente que no hubo pérdida de registros durante la transición del servidor principal al de contingencia.

SQL Server Management Studio (SSMS) es la herramienta principal utilizada para administrar y optimizar las consultas. Los tres capturas de pantalla muestran la ejecución de consultas complejas y su resultado.

La primera captura muestra la ejecución de la consulta #1, que obtiene estadísticas de fragmentación para el índice IX_Ventas_ReporteGerencial.

La segunda captura muestra la ejecución de la consulta #5, que genera un informe gerencial de ventas detallado por cliente, categoría y producto.

La tercera captura muestra la ejecución de la consulta #1, que obtiene estadísticas de fragmentación para el índice IX_Ventas_ReporteGerencial.