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: ___GONZALO AGUILERA LUIS ANGEL_____ Código: _R01067C___
Ciclo: _____V_____ Salón: _____A1_____

Enunciado 01:

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

1) Explique que problema soluciona su base de datos
2) Implemente un Script para crear una **vista** para crear utilizando tres tablas
3) Implemente un Script para crear un **procedimiento almacenado** para modificar el ingreso de datos en forma secuencial
4) Implemente un Script para crear un **disparador** para verificar el control de datos (Ejemplo: que la nota ingresada este entre 0 y 20)
5) Utilizando Script Crear 03 usuarios con nombres de sus compañeros y uno suyo
6) Utilizando un script, copiar la base de datos (creada anteriormente) y compartir en cada uno de los usuarios
7) Utilizando un script, generar una copia de seguridad de la base de datos y compartir a cada uno de los usuarios
8) Utilizando un script, encriptar una de las tablas para que no se puedan ver los datos
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
10) Utilizando un script, implementé seguridad a nivel de columna restringiendo el acceso a una de las columnas de una tabla.
11) Utilizando un script, realice el cifrado transparente de datos (TDE) para una las tablas.
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.
13) Utilizando un script, configure la auditoría para el seguimiento y registro de acciones en la base de datos
14) Utilizando un script, configure de la memoria y el disco duro
15) Utilizando un script, genere una copia de seguridad de la base de datos
16) Utilizando un script, genere la restauración de la base de datos
17) Utilizando un script, cree un espejo de la base de datos
18) Utilizando un script, realice la replicación de bases de datos
19) Explique que es Always On Availability Groups
20) Explique que es Log Shipping

1. **Problema que soluciona la base de datos:** La base de datos desarrollada en Microsoft SQL Server está diseñada para centralizar y gestionar datos de manera eficiente, asegurando integridad, seguridad, y disponibilidad. Por ejemplo, puede resolver problemas como la redundancia de datos, dificultad en la obtención de informes y manejo inseguro de datos sensibles.

---

2. **Script para crear una vista utilizando tres tablas:**

```
sql
Copiar código
CREATE VIEW VistaEjemplo AS
SELECT
    e.Nombre AS Empleado,
    d.Nombre AS Departamento,
    p.Nombre AS Proyecto
FROM
    Empleados e
JOIN
    Departamentos d ON e.DepartamentoID = d.DepartamentoID
JOIN
    Proyectos p ON d.ProyectoID = p.ProyectoID;
```

---

3. **Script para un procedimiento almacenado que modifique datos en forma secuencial:**

```
sql
Copiar código
CREATE PROCEDURE ActualizarSalarioSecuencial
@EmpleadoID INT,
@NuevoSalario DECIMAL(10, 2)
AS
BEGIN
    UPDATE Empleados
    SET Salario = @NuevoSalario
    WHERE EmpleadoID = @EmpleadoID;
END;
```

---

4. **Script para un disparador que verifique que una nota esté entre 0 y 20:**

```
sql
Copiar código
CREATE TRIGGER VerificarNota
```

```
ON Notas
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM Inserted
        WHERE Nota < 0 OR Nota > 20
    )
    BEGIN
        ROLLBACK TRANSACTION;
        RAISERROR ('La nota debe estar entre 0 y 20.', 16, 1);
    END
END;
```


---

5. **Script para crear 3 usuarios y asignarles roles:**

```
sql
Copiar código
CREATE LOGIN Usuario1 WITH PASSWORD = 'Password123';
CREATE LOGIN Usuario2 WITH PASSWORD = 'Password123';
CREATE LOGIN Usuario3 WITH PASSWORD = 'Password123';

CREATE USER Usuario1 FOR LOGIN Usuario1;
CREATE USER Usuario2 FOR LOGIN Usuario2;
CREATE USER Usuario3 FOR LOGIN Usuario3;
```

Punto 6: Script para la creación de un índice en una tabla

Los índices son fundamentales para mejorar el rendimiento de las consultas.

```
sql
Copiar código
CREATE NONCLUSTERED INDEX IDX_NombreEmpleado
ON Empleados (Nombre);
```

Punto 7: Script para la implementación de cifrado

Este script muestra cómo implementar el cifrado de datos sensibles, como números de tarjetas de crédito.

1. **Habilitar el cifrado en SQL Server:**

```
sql
Copiar código
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'ClaveSegura123!';
CREATE CERTIFICATE CertificadoSeguridad
WITH SUBJECT = 'Cifrado de datos';
CREATE SYMMETRIC KEY LlaveSimetrica
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE CertificadoSeguridad;
```

2. **Cifrar datos en una tabla:**

```
sql
Copiar código
OPEN SYMMETRIC KEY LlaveSimetrica
DECRYPTION BY CERTIFICATE CertificadoSeguridad;

UPDATE TarjetasCredito
SET NumeroTarjetaCifrada =
ENCRYPTBYKEY(KEY_GUID('LlaveSimetrica'), NumeroTarjeta);

CLOSE SYMMETRIC KEY LlaveSimetrica;
```


3. **Consultar datos cifrados:**

```
sql
Copiar código
OPEN SYMMETRIC KEY LlaveSimetrica
DECRYPTION BY CERTIFICATE CertificadoSeguridad;

SELECT
    TarjetaID,
    CONVERT(VARCHAR, DECRYPTBYKEY(NumeroTarjetaCifrada)) AS
NumeroTarjeta
FROM TarjetasCredito;

CLOSE SYMMETRIC KEY LlaveSimetrica;
```

Punto 8: Seguridad a nivel de columna

Aquí se deniega acceso a columnas específicas según el rol del usuario.

1. **Crear un rol y denegar acceso:**

```
sql
Copiar código
CREATE ROLE LecturaLimitada;
DENY SELECT ON Empleados(Salario) TO LecturaLimitada;
```

2. **Asignar el rol al usuario:**

```
sql
Copiar código

EXEC sp_addrolemember 'LecturaLimitada', 'Usuario1';
```

Punto 9: Replicación de base de datos

SQL Server permite replicación transaccional, de mezcla o instantánea. Aquí muestro cómo configurar una replicación transaccional básica.

1. **Habilitar el servidor como publicador:**

```
sql
Copiar código
EXEC sp_replicationdboption
    @dbname = 'MiBaseDatos',
    @optname = 'publish',
    @value = 'true';
```


2. **Crear una publicación:**

```
sql
Copiar código
EXEC sp_addpublication
    @publication = 'PublicacionEjemplo',
    @status = 'active';
```

3. **Agregar un artículo (tabla) a la publicación:**

```
sql
Copiar código
EXEC sp_addarticle
    @publication = 'PublicacionEjemplo',
    @article = 'Empleados',
    @source_object = 'Empleados';
```

4. **Configurar el suscriptor:**

```
sql
Copiar código
EXEC sp_addsubscription
    @publication = 'PublicacionEjemplo',
    @subscriber = 'ServidorSuscriptor',
    @destination_db = 'BaseDatosSuscriptor';
```

Punto 10: Recuperación y respaldo

1. **Crear un respaldo de la base de datos:**

```
sql
Copiar código
BACKUP DATABASE MiBaseDatos
TO DISK = 'C:\Respaldo\MiBaseDatos.bak'
WITH FORMAT, MEDIANAME = 'RespaldoMedia', NAME =
'RespaldoCompleto';
```

2. **Restaurar una base de datos desde el respaldo:**

```
sql
Copiar código
RESTORE DATABASE MiBaseDatosRecuperada
FROM DISK = 'C:\Respaldo\MiBaseDatos.bak'
```

```
    WITH MOVE 'MiBaseDatos' TO
    'C:\Archivos\MiBaseDatosRecuperada.mdf',
        MOVE 'MiBaseDatos_log' TO
    'C:\Archivos\MiBaseDatosRecuperada.ldf';
```


Punto 11: Crear una tabla particionada

El particionado permite distribuir los datos de una tabla en diferentes archivos para mejorar el rendimiento.

1. **Crear una función de partición:**

   ```
   sql
   Copiar código
   CREATE PARTITION FUNCTION FuncionParticion (INT)
   AS RANGE LEFT FOR VALUES (1000, 2000, 3000);
   ```

2. **Crear un esquema de partición:**

   ```
   sql
   Copiar código
   CREATE PARTITION SCHEME EsquemaParticion
   AS PARTITION FuncionParticion
   TO (GrupoArchivo1, GrupoArchivo2, GrupoArchivo3, GrupoArchivo4);
   ```

3. **Crear la tabla particionada:**

   ```
   sql
   Copiar código
   CREATE TABLE Facturas
   (
       FacturaID INT NOT NULL,
       Monto DECIMAL(10, 2),
       Fecha DATETIME
   )
   ON EsquemaParticion(FacturaID);
   ```

Punto 12: Consultas con CTE (Common Table Expressions)

Un CTE es útil para consultas complejas y recursivas.

```
sql
Copiar código
WITH EmpleadosCTE AS (
    SELECT Nombre, Apellido, Salario
    FROM Empleados
    WHERE Salario > 5000
)
SELECT *
FROM EmpleadosCTE
WHERE Salario < 10000;
```

Punto 13: Creación de una vista

Las vistas son útiles para encapsular consultas complejas.

```
sql
Copiar código
CREATE VIEW VistaEmpleadosAltosSalarios AS
SELECT Nombre, Apellido, Salario
FROM Empleados
WHERE Salario > 10000;
```

Punto 14: Procedimiento almacenado con parámetros

Un procedimiento para consultar empleados por departamento.

```
sql
Copiar código
CREATE PROCEDURE ObtenerEmpleadosPorDepartamento
    @DepartamentoID INT
AS
BEGIN
    SELECT Nombre, Apellido
    FROM Empleados
    WHERE DepartamentoID = @DepartamentoID;
END;
```

Punto 15: Transacciones para garantizar consistencia

Las transacciones aseguran que las operaciones se realicen de forma atómica.

```
sql
Copiar código
BEGIN TRANSACTION;

UPDATE Cuentas
SET Saldo = Saldo - 100
WHERE CuentaID = 1;

UPDATE Cuentas
SET Saldo = Saldo + 100
WHERE CuentaID = 2;

IF @@ERROR <> 0
    ROLLBACK TRANSACTION;
ELSE
    COMMIT TRANSACTION;
```

Punto 16: Implementación de triggers

Los triggers automatizan tareas después de eventos en la base de datos.

```
sql
Copiar código
CREATE TRIGGER TriggerAuditoria
ON Empleados
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    INSERT INTO Auditoria (Accion, Fecha, Usuario)
    VALUES (EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]',
'NVARCHAR(50)'), GETDATE(), SYSTEM_USER);
END;
```

Punto 17: Optimización de consultas

Para optimizar, utilizamos índices y estadísticas.

1. **Revisar planes de ejecución:**

   ```
   sql
   Copiar código
   SET STATISTICS TIME ON;
   SELECT * FROM Empleados WHERE Salario > 5000;
   SET STATISTICS TIME OFF;
   ```

2. **Crear un índice en una consulta frecuentemente usada:**

   ```
   sql
   Copiar código
   CREATE INDEX IDX_DepartamentoID ON Empleados (DepartamentoID);
   ```

Punto 18: Backup diferencial

Un respaldo diferencial almacena solo los cambios desde el último respaldo completo.

```
sql
Copiar código
BACKUP DATABASE MiBaseDatos
TO DISK = 'C:\Respaldo\MiBaseDatos_Diferencial.bak'
WITH DIFFERENTIAL;
```

Punto 19: Uso de funciones definidas por el usuario

Las funciones encapsulan operaciones que retornan valores.

1. **Función escalar:**

   ```
   sql
   Copiar código
   CREATE FUNCTION CalcularImpuesto (@Monto DECIMAL(10, 2))
   RETURNS DECIMAL(10, 2)
   AS
   ```

```
BEGIN
    RETURN @Monto * 0.18;
END;
```

Uso:

```
sql
Copiar código
SELECT Nombre, CalcularImpuesto(Salario) AS Impuesto
FROM Empleados;
```


2. **Función de tabla:**

```
sql
Copiar código
CREATE FUNCTION ObtenerEmpleadosPorEdad (@Edad INT)
RETURNS TABLE
AS
RETURN
(
    SELECT Nombre, Apellido
    FROM Empleados
    WHERE Edad = @Edad
);
```

Uso:

```
sql
Copiar código
SELECT * FROM ObtenerEmpleadosPorEdad(30);
```

Punto 20: Manejo de concurrencia

Configurar niveles de aislamiento para evitar problemas como bloqueos.

1. **Ejemplo con aislamiento READ COMMITTED:**

```
sql
Copiar código
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

BEGIN TRANSACTION;
SELECT * FROM Empleados WITH (ROWLOCK) WHERE DepartamentoID = 1;
UPDATE Empleados SET Salario = Salario + 100 WHERE
DepartamentoID = 1;
COMMIT TRANSACTION;
```

2. **Usar hints para control de bloqueo:**

```
sql
Copiar código
SELECT * FROM Empleados WITH (NOLOCK);
```