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: ROJAS PANDURO GIAN JHERSON Código: R01125C
Ciclo: 5TO Salón: A1

1) Explicar qué problema soluciona su base de datos:

La base de datos está diseñada para gestionar la información de una institución educativa. En particular, resuelve el problema de almacenar y organizar los datos de los estudiantes, sus calificaciones, asignaturas y profesores, permitiendo el acceso y la actualización de información de manera eficiente y segura.

---

2) Script para crear una vista utilizando tres tablas:

Supongamos que tenemos las siguientes tablas: `Estudiantes`, `Asignaturas` y `Calificaciones`.

```
sql
Copiar código
CREATE VIEW Vista_Resultado AS
SELECT
    E.NombreEstudiante,
    A.NombreAsignatura,
    C.Calificacion
FROM
    Estudiantes E
JOIN
    Calificaciones C ON E.ID_Estudiante = C.ID_Estudiante
JOIN
    Asignaturas A ON A.ID_Asignatura = C.ID_Asignatura;
```

---

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

Este procedimiento almacenado asegura que se ingresen los datos en una secuencia específica.

```
sql
Copiar código
CREATE PROCEDURE InsertarCalificacion
    @ID_Estudiante INT,
    @ID_Asignatura INT,
    @Calificacion DECIMAL(5,2)
```

```
AS
BEGIN
    -- Verificar si el estudiante ya tiene una calificación para la
asignatura
    IF NOT EXISTS (SELECT * FROM Calificaciones WHERE ID_Estudiante =
@ID_Estudiante AND ID_Asignatura = @ID_Asignatura)
    BEGIN
        INSERT INTO Calificaciones (ID_Estudiante, ID_Asignatura,
Calificacion)
        VALUES (@ID_Estudiante, @ID_Asignatura, @Calificacion);
    END
    ELSE
    BEGIN
        UPDATE Calificaciones
        SET Calificacion = @Calificacion
        WHERE ID_Estudiante = @ID_Estudiante AND ID_Asignatura =
@ID_Asignatura;
    END
END;
```


---

4) Script para crear un disparador que verifique el control de datos (nota entre 0 y 20):

```
sql
Copiar código
CREATE TRIGGER VerificarNota
ON Calificaciones
AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @Nota DECIMAL(5,2);

    SELECT @Nota = Calificacion FROM inserted;

    IF @Nota < 0 OR @Nota > 20
    BEGIN
        RAISERROR('La nota debe estar entre 0 y 20.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
```

---

5) Script para crear 03 usuarios con nombres de sus compañeros y uno suyo:

```
sql
Copiar código
CREATE LOGIN Usuario1 WITH PASSWORD = 'ContraseñaSegura1';
CREATE USER Usuario1 FOR LOGIN Usuario1;

CREATE LOGIN Usuario2 WITH PASSWORD = 'ContraseñaSegura2';
CREATE USER Usuario2 FOR LOGIN Usuario2;

CREATE LOGIN Usuario3 WITH PASSWORD = 'ContraseñaSegura3';
CREATE USER Usuario3 FOR LOGIN Usuario3;

CREATE LOGIN MiUsuario WITH PASSWORD = 'MiContraseñaSegura';
```

```
CREATE USER MiUsuario FOR LOGIN MiUsuario;
```

---

6) Script para copiar la base de datos y compartir en cada uno de los usuarios:

```
sql
Copiar código
-- Copiar base de datos
BACKUP DATABASE MiBaseDatos TO DISK = 'C:\Backup\MiBaseDatos.bak';

-- Restaurar base de datos para cada usuario
RESTORE DATABASE MiBaseDatosCopia FROM DISK =
'C:\Backup\MiBaseDatos.bak';

-- Otorgar permisos a los usuarios
GRANT SELECT, INSERT, UPDATE, DELETE ON DATABASE::MiBaseDatosCopia TO
Usuario1;
GRANT SELECT, INSERT, UPDATE, DELETE ON DATABASE::MiBaseDatosCopia TO
Usuario2;
GRANT SELECT, INSERT, UPDATE, DELETE ON DATABASE::MiBaseDatosCopia TO
Usuario3;
GRANT SELECT, INSERT, UPDATE, DELETE ON DATABASE::MiBaseDatosCopia TO
MiUsuario;
```


---

7) Script para generar una copia de seguridad de la base de datos y compartir a cada uno de los usuarios:

```
sql
Copiar código
BACKUP DATABASE MiBaseDatos TO DISK = 'C:\Backup\MiBaseDatos.bak';

-- Otorgar acceso al backup
GRANT SELECT ON FILE::'C:\Backup\MiBaseDatos.bak' TO Usuario1;
GRANT SELECT ON FILE::'C:\Backup\MiBaseDatos.bak' TO Usuario2;
GRANT SELECT ON FILE::'C:\Backup\MiBaseDatos.bak' TO Usuario3;
GRANT SELECT ON FILE::'C:\Backup\MiBaseDatos.bak' TO MiUsuario;
```

---

8) Script para encriptar una tabla:

```
sql
Copiar código
CREATE DATABASE EncriptadoDB;

USE EncriptadoDB;

CREATE TABLE Empleados (
    ID INT PRIMARY KEY,
    Nombre VARCHAR(100),
    Salario DECIMAL(10, 2)
);

-- Crear una clave para la encriptación
CREATE SYMMETRIC KEY MiClaveSimetrica
    WITH ALGORITHM = AES_256
    ENCRYPTION BY PASSWORD = 'ContraseñaSegura';
```

```
-- Encriptar los datos al insertarlos
OPEN SYMMETRIC KEY MiClaveSimetrica DECRYPTION BY PASSWORD =
'contraseñaSegura';
INSERT INTO Empleados (ID, Nombre, Salario)
VALUES (1, 'Juan Perez', EncryptByKey(Key_GUID('MiClaveSimetrica'),
'5000'));

CLOSE SYMMETRIC KEY MiClaveSimetrica;
```


---

9) Script para aplicar seguridad a nivel de columna restringiendo el acceso a la columna DNI:

```
sql
Copiar código
DENY SELECT ON COLUMN Empleados.DNI TO Usuario2;
```

---

10) Script para aplicar seguridad a nivel de columna restringiendo el acceso a una columna:

```
sql
Copiar código
DENY SELECT ON COLUMN Empleados.Salario TO Usuario3;
```

---

11) Script para realizar el cifrado transparente de datos (TDE):

```
sql
Copiar código
CREATE DATABASE MiBaseDeDatosEncriptada;
GO
USE master;
GO

-- Crear una clave maestra
CREATE DATABASE ENCRYPTION KEY;
GO

-- Activar TDE en la base de datos
ALTER DATABASE MiBaseDeDatosEncriptada
SET ENCRYPTION ON;
```

---

12) Script para otorgar permisos de SELECT, INSERT, UPDATE y DELETE a un usuario:

```
sql
Copiar código
GRANT SELECT, INSERT, UPDATE, DELETE ON DATABASE::MiBaseDatos TO
Usuario1;
```

---

13) Script para configurar la auditoría para el seguimiento y registro de acciones:

```
sql
Copiar código
CREATE SERVER AUDIT MiAuditoria
TO FILE (FILEPATH = 'C:\Auditoria\');
ALTER SERVER AUDIT MiAuditoria
WITH (STATE = ON);

CREATE SERVER AUDIT SPECIFICATION MiEspecificacionAuditoria
FOR SERVER AUDIT MiAuditoria
ADD (SUCCESSFUL_LOGIN_GROUP),
ADD (FAILED_LOGIN_GROUP)
WITH (STATE = ON);
```

---

14) Script para configurar la memoria y el disco duro:

```
sql
Copiar código
-- Establecer configuración de memoria
EXEC sp_configure 'max server memory', 2048;  -- 2GB de memoria
RECONFIGURE;

-- Establecer configuración de disco duro
ALTER DATABASE MiBaseDatos
MODIFY FILE (NAME = MiArchivoDeDatos, SIZE = 10GB);
```

---

15) Script para generar una copia de seguridad de la base de datos:

```
sql
Copiar código
BACKUP DATABASE MiBaseDatos TO DISK = 'C:\Backup\MiBaseDatos.bak';
```

---

16) Script para generar la restauración de la base de datos:

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

---

17) Script para crear un espejo de la base de datos:

```
sql
Copiar código
-- Configuración de la base de datos espejo
ALTER DATABASE MiBaseDatos
SET PARTNER = 'TCP://ServidorSecundario:5022';
```

---

18) Script para realizar la replicación de bases de datos:

```
sql
Copiar código
-- Configuración de publicación de datos
EXEC sp_addpublication
    @publication = 'MiPublicacion',
    @publisher = 'ServidorPrimario',
    @publication_type = 0, -- Transaccional
    @description = 'Replicación de datos';

-- Configuración de suscripción
EXEC sp_addsubscription
    @publication = 'MiPublicacion',
    @subscriber = 'ServidorSecundario',
    @subscription_type = 0; -- Solo lectura
```


---

19) ¿Qué es Always On Availability Groups?

Always On Availability Groups es una característica de alta disponibilidad y recuperación ante desastres en SQL Server. Permite la creación de grupos de bases de datos que pueden estar disponibles en múltiples servidores (nodos), proporcionando redundancia, failover automático y protección de datos.

---

20) ¿Qué es Log Shipping?

Log Shipping es una técnica de alta disponibilidad que permite la copia continua de los registros de transacciones de una base de datos principal a una o más bases de datos secundarias. Es útil para recuperar datos tras una falla en el servidor principal.