

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



ADMINISTRACIÓN DE BASE DE DATOS

Integrantes:

Cajahuarina Samaniego Geyzon

Luis Ramirez Brayan ESnyder

Docente:

RAUL FERNANDEZ BEJARANO

Asignatura:

Base de datos II

HUANCAYO – PERÚ

2024

Administración de Base de Datos

Enunciado 01:

De acuerdo con la base de datos implementada (mínimo 100 registros), utilice los DBMS Microsoft SQL Server/MySQL, o un servidor de la nube como Microsoft Azure o Google FireBase. Explique qué problema soluciona su base de datos y responda las siguientes preguntas:

1) Implemente y explique un Script para crear una vista para crear utilizando tres tablas

```
GO
-- Paso 2: Crear la vista
CREATE VIEW vista_usuarios_detallada AS
SELECT
    u.[USUARIO],
    u.[GÉNERO],
    u.[FECHA NAC] AS FECHA_NACIMIENTO, -- Renombramos la columna para un formato más claro
    u.[EDAD],
    u.[NACIONALIDAD],
    u.[EDO# ORIGEN] AS ESTADO_ORIGEN, -- Corchetes para manejar el carácter especial '#'
    g.[FECHA DE INGRESO] AS FECHA_INGRESO, -- Renombramos la columna para consistencia
    g.[NIVEL DE PROCEDENCIA],
    g.[INGRESO],
    g.[NIVEL ACTUAL],
    g.[CATEGORÍA]
FROM
    [Univerdaddddd].[dbo].[Usuarios$] u
JOIN
    [Univerdaddddd].[dbo].[Grados$] g
ON
    u.[USUARIO] = g.[USUARIO]; -- Clave común para unir las tablas
GO
SELECT * FROM vista_usuarios_detallada;
```



	USUARIO	GÉNERO	FECHA_NACIMIENTO	EDAD	NACIONALIDAD	ESTADO_ORIGEN	FECHA_INGRESO	NIVEL DE PROCEDENCIA	INGRESO	NIVEL ACTUAL	CATEGORÍA
1	1	H	2004-10-16 00:00:00.000	17	MEXICANA	CHIAPAS	2022-08-05 00:00:00.000	PREPARATORIA	PROPEDEÚTICO	PROPEDEÚTICO	BECADO EXTERNO
2	1	H	2004-10-16 00:00:00.000	17	MEXICANA	CHIAPAS	2022-08-05 00:00:00.000	PREPARATORIA	PROPEDEÚTICO	PROPEDEÚTICO	BECADO EXTERNO
3	1	H	2004-10-16 00:00:00.000	17	MEXICANA	CHIAPAS	2022-08-05 00:00:00.000	PREPARATORIA	PROPEDEÚTICO	PROPEDEÚTICO	BECADO EXTERNO
4	1	H	2004-10-16 00:00:00.000	17	MEXICANA	CHIAPAS	2022-08-05 00:00:00.000	PREPARATORIA	PROPEDEÚTICO	PROPEDEÚTICO	BECADO EXTERNO
5	2	H	2001-02-02 00:00:00.000	21	MEXICANA	CAMPECHE	2022-08-07 00:00:00.000	PREPARATORIA	PROPEDEÚTICO	PROPEDEÚTICO	EXTERNO
6	2	H	2001-02-02 00:00:00.000	21	MEXICANA	CAMPECHE	2022-08-07 00:00:00.000	PREPARATORIA	PROPEDEÚTICO	PROPEDEÚTICO	EXTERNO
7	2	H	2001-02-02 00:00:00.000	21	MEXICANA	CAMPECHE	2022-08-07 00:00:00.000	PREPARATORIA	PROPEDEÚTICO	PROPEDEÚTICO	EXTERNO
8	2	H	2001-02-02 00:00:00.000	21	MEXICANA	CAMPECHE	2022-08-07 00:00:00.000	PREPARATORIA	PROPEDEÚTICO	PROPEDEÚTICO	EXTERNO
9	3	M	2007-01-02 00:00:00.000	15	MEXICANA	CHIAPAS	2022-08-01 00:00:00.000	SECUNDARIA	PREPARATORIA	MEDIO SUPERIOR	BECADO INTERNO
10	3	M	2007-01-02 00:00:00.000	15	MEXICANA	CHIAPAS	2022-08-01 00:00:00.000	SECUNDARIA	PREPARATORIA	MEDIO SUPERIOR	BECADO INTERNO
11	3	M	2007-01-02 00:00:00.000	15	MEXICANA	CHIAPAS	2022-08-01 00:00:00.000	SECUNDARIA	PREPARATORIA	MEDIO SUPERIOR	BECADO INTERNO
12	3	M	2007-01-02 00:00:00.000	15	MEXICANA	CHIAPAS	2022-08-01 00:00:00.000	SECUNDARIA	PREPARATORIA	MEDIO SUPERIOR	BECADO INTERNO
13	4	H	2000-03-07 00:00:00.000	22	MEXICANA	CAMPECHE	2019-09-06 00:00:00.000	LICENCIATURA	PROPEDEÚTICO	LICENCIATURA	EXTERNO
14	4	H	2000-03-07 00:00:00.000	22	MEXICANA	CAMPECHE	2019-09-06 00:00:00.000	LICENCIATURA	PROPEDEÚTICO	LICENCIATURA	EXTERNO
15	4	H	2000-03-07 00:00:00.000	22	MEXICANA	CAMPECHE	2019-09-06 00:00:00.000	LICENCIATURA	PROPEDEÚTICO	LICENCIATURA	EXTERNO
16	4	H	2000-03-07 00:00:00.000	22	MEXICANA	CAMPECHE	2019-09-06 00:00:00.000	LICENCIATURA	PROPEDEÚTICO	LICENCIATURA	EXTERNO
17	5	H	2007-08-02 00:00:00.000	15	MEXICANA	VERACRUZ	2022-08-02 00:00:00.000	SECUNDARIA	PREPARATORIA	MEDIO SUPERIOR	BECADO EXTERNO
18	5	H	2007-08-02 00:00:00.000	15	MEXICANA	VERACRUZ	2022-08-02 00:00:00.000	SECUNDARIA	PREPARATORIA	MEDIO SUPERIOR	BECADO EXTERNO
19	5	H	2007-08-02 00:00:00.000	15	MEXICANA	VERACRUZ	2022-08-02 00:00:00.000	SECUNDARIA	PREPARATORIA	MEDIO SUPERIOR	BECADO EXTERNO

2) Implemente y explique un Script para crear un procedimiento almacenado para insertar datos a su base de datos.

```
-- Ejecutar el procedimiento con otro set de datos
EXEC insertar_usuario_grado
    @USUARIO = 999999,          -- ID del usuario
    @GENERO = 'M',              -- Género (H/M)
    @FECHA_NAC = '1998-11-03',  -- Fecha de nacimiento (YYYY-MM-DD)
    @EDAD = 26,                 -- Edad
    @NACIONALIDAD = 'ARGENTINA', -- Nacionalidad
    @EDO_ORIGEN = 'BUENOS AIRES', -- Estado de origen
    @FECHA_INGRESO = '2021-06-15', -- Fecha de ingreso
    @NIVEL_PROC = 'BACHILLERATO', -- Nivel de procedencia
    @INGRESO = 'TÉCNICO',       -- Tipo de ingreso
    @NIVEL_ACTUAL = 'TÉCNICO',  -- Nivel actual
    @CATEGORIA = 'BECADO EXTERNO'; -- Categoría

SELECT * FROM [Univerdadddd].[dbo].[Usuarios$];
SELECT * FROM [Univerdadddd].[dbo].[Grados$];
```

100 %

Results Messages

	USUARIO	GÉNERO	FECHA NAC	EDAD	NACIONALIDAD	EDO# ORIGEN
805	3	H	2000-05-15 00:00:00.000	24	COLOMBIANA	ANTIOQUIA
806	4	M	1998-11-03 00:00:00.000	26	ARGENTINA	BUENOS AIR...
807	2	M	2001-02-02 00:00:00.000	21	MEXICANA	CAMPECHE
808	3	H	2000-05-15 00:00:00.000	24	COLOMBIANA	ANTIOQUIA
809	4	M	1998-11-03 00:00:00.000	26	ARGENTINA	BUENOS AIR...
810	2	M	2001-02-02 00:00:00.000	21	MEXICANA	CAMPECHE
811	3	H	2000-05-15 00:00:00.000	24	COLOMBIANA	ANTIOQUIA
812	999999	M	1998-11-03 00:00:00.000	26	ARGENTINA	BUENOS AIR...

	USUARIO	FECHA DE INGRESO	NIVEL DE PROCEDENCIA	INGRESO	NIVEL ACTUAL	CATEGORÍA
805	3	2023-01-12 00:00:00.000	SECUNDARIA	BACHILLERATO	BACHILLERATO	BECADO INTERNO
806	4	2021-06-15 00:00:00.000	BACHILLERATO	TÉCNICO	TÉCNICO	BECADO EXTERNO
807	2	2022-08-07 00:00:00.000	PREPARATORIA	PROPEDÉUTICO	PROPEDÉUTICO	BECADO EXTERNO
808	3	2023-01-12 00:00:00.000	SECUNDARIA	BACHILLERATO	BACHILLERATO	BECADO INTERNO
809	4	2021-06-15 00:00:00.000	BACHILLERATO	TÉCNICO	TÉCNICO	BECADO EXTERNO
810	2	2022-08-07 00:00:00.000	PREPARATORIA	PROPEDÉUTICO	PROPEDÉUTICO	BECADO EXTERNO
811	3	2023-01-12 00:00:00.000	SECUNDARIA	BACHILLERATO	BACHILLERATO	BECADO INTERNO
812	999999	2021-06-15 00:00:00.000	BACHILLERATO	TÉCNICO	TÉCNICO	BECADO EXTERNO

3) Implemente y explique un Script para crear un procedimiento almacenado para eliminar datos a su base de datos

```

SELECT *
FROM [Univerdadddd].[dbo].[Usuarios$]
WHERE [USUARIO] = 3;

```

```

SELECT *
FROM [Univerdadddd].[dbo].[Grados$]
WHERE [USUARIO] = 3;

```

100 %

Results Messages

USUARIO	GÉNERO	FECHA NAC	EDAD	NACIONALIDAD	EDO# ORIGEN

USUARIO	FECHA DE INGRESO	NIVEL DE PROCEDENCIA	INGRESO	NIVEL ACTUAL	CATEGORÍA

DELETE FROM [tabla] WHERE [USUARIO] = 3; elimina los registros donde el campo **USUARIO** tiene el valor 3.

Después de ejecutar el procedimiento, las consultas de verificación te confirmarán que no existen más registros con **USUARIO = 3**.

4) Implemente y explique un Script para crear un procedimiento almacenado para actualizar datos a su base de datos:

SQLQuery6.sql - LA_PC14\USER 17 (56) SQLQuery5.sql - LA_PC14\USER 17 (66)* SQLQuery1.sql - LA_PC14\USER 17 (86)

```

[INGRESO] = @INGRESO,
[NIVEL_ACTUAL] = @NIVEL_ACTUAL,
[CATEGORIA] = @CATEGORIA
WHERE [USUARIO] = @USUARIO;
END;
GO
-- Ejecutar el procedimiento para actualizar los datos del usuario con USUARIO = 2
EXEC actualizar_usuario_grado
@USUARIO = 2,
@GENERO = 'F',
@FECHA_NAC = '2000-05-10',
@EDAD = 24,
@NACIONALIDAD = 'MEXICANITA',
@EDO_ORIGEN = 'CAMPECHE',
@FECHA_INGRESO = '2022-08-01',
@NIVEL_PROC = 'PREPARATORIA',
@INGRESO = 'BÁSICO',
@NIVEL_ACTUAL = 'PREPARATORIA',
@CATEGORIA = 'EXTERNO';
SELECT *
FROM [Univerdadddd].[dbo].[Usuarios$]
WHERE [USUARIO] = 2;
SELECT *
FROM [Univerdadddd].[dbo].[Grados$]
WHERE [USUARIO] = 2;

```

100 %

Results Messages

	USUARIO	GÉNERO	FECHA NAC	EDAD	NACIONALIDAD	EDO#ORIGEN
1	2	F	2000-05-10 00:00:00.000	24	MEXICANITA	CAMPECHE
2	2	F	2000-05-10 00:00:00.000	24	MEXICANITA	CAMPECHE
3	2	F	2000-05-10 00:00:00.000	24	MEXICANITA	CAMPECHE
4	2	F	2000-05-10 00:00:00.000	24	MEXICANITA	CAMPECHE
5	2	F	2000-05-10 00:00:00.000	24	MEXICANITA	CAMPECHE
6	2	F	2000-05-10 00:00:00.000	24	MEXICANITA	CAMPECHE

	USUARIO	FECHA DE INGRESO	NIVEL DE PROCEDENCIA	INGRESO	NIVEL ACTUAL	CATEGORIA
1	2	2022-08-01 00:00:00.000	PREPARATORIA	BÁSICO	PREPARATORIA	EXTERNO
2	2	2022-08-01 00:00:00.000	PREPARATORIA	BÁSICO	PREPARATORIA	EXTERNO
3	2	2022-08-01 00:00:00.000	PREPARATORIA	BÁSICO	PREPARATORIA	EXTERNO
4	2	2022-08-01 00:00:00.000	PREPARATORIA	BÁSICO	PREPARATORIA	EXTERNO
5	2	2022-08-01 00:00:00.000	PREPARATORIA	BÁSICO	PREPARATORIA	EXTERNO
6	2	2022-08-01 00:00:00.000	PREPARATORIA	BÁSICO	PREPARATORIA	EXTERNO

Query executed successfully.

LAB04-PC14\MSSQLSERVERDEV (L... LAB04-PC14\USER 17 (66) Univerdadddd 00:00:00 12 rows

1. Verificar en la tabla **Usuarios\$**:

```

SELECT *
FROM [Univerdadddd].[dbo].[Usuarios$]
WHERE [USUARIO] = 2;

```

2. Verificar en la tabla **Grados\$**:

```

SELECT *
FROM [Univerdadddd].[dbo].[Grados$]
WHERE [USUARIO] = 2;

```

Ambas consultas deberían devolver los nuevos valores que has proporcionado al ejecutar el procedimiento.

5) Implemente y explique un Script para crear un procedimiento almacenado para realizar cálculos matemáticos de una columna de su base de datos.

```

-- Paso 1: Eliminar el procedimiento si ya existe
IF OBJECT_ID('calculos_usuario', 'P') IS NOT NULL
    DROP PROCEDURE calculos_usuario;
GO

-- Paso 2: Crear el procedimiento almacenado
CREATE PROCEDURE calculos_usuario
    @RESULTADO_PROMEDIO FLOAT OUTPUT, -- Resultado del promedio
    @RESULTADO_SUMATORIA INT OUTPUT, -- Resultado de la suma
    @RESULTADO_MAXIMO INT OUTPUT, -- Resultado del máximo
    @RESULTADO_MINIMO INT OUTPUT -- Resultado del mínimo
AS
BEGIN
    -- Paso 3: Calcular el promedio de la columna EDAD en Usuarios$
    SELECT @RESULTADO_PROMEDIO = AVG([EDAD])
    FROM [Univerdadddd].[dbo].[Usuarios$];

    -- Paso 4: Calcular la suma de la columna EDAD en Usuarios$
    SELECT @RESULTADO_SUMATORIA = SUM([EDAD])
    FROM [Univerdadddd].[dbo].[Usuarios$];

    -- Paso 5: Calcular el valor máximo de la columna EDAD en Usuarios$
    SELECT @RESULTADO_MAXIMO = MAX([EDAD])
    FROM [Univerdadddd].[dbo].[Usuarios$];

    -- Paso 6: Calcular el valor mínimo de la columna EDAD en Usuarios$
    SELECT @RESULTADO_MINIMO = MIN([EDAD])
    FROM [Univerdadddd].[dbo].[Usuarios$];
END;
GO

```

```

-- Paso 1: Declarar las variables para recibir los resultados
DECLARE @PROMEDIO FLOAT,
        @SUMATORIA INT,
        @MAXIMO INT,
        @MINIMO INT;

```

```

-- Paso 2: Ejecutar el procedimiento
EXEC calculos_usuario
    @RESULTADO_PROMEDIO = @PROMEDIO OUTPUT,
    @RESULTADO_SUMATORIA = @SUMATORIA OUTPUT,
    @RESULTADO_MAXIMO = @MAXIMO OUTPUT,
    @RESULTADO_MINIMO = @MINIMO OUTPUT;

```

```

-- Paso 3: Ver los resultados
SELECT

```

100 %

Results Messages

	Promedio	Suma	Maximo	Minimo
1	19.1439205955335	15430	32	14

```

-- Paso 3: Ver los resultados
SELECT
    @PROMEDIO AS Promedio,
    @SUMATORIA AS Suma,
    @MAXIMO AS Maximo,
    @MINIMO AS Minimo;

```

6) Implemente y explique un Script para crear un disparador para ingresar un registro automáticamente en una tabla de su base de datos.

1. Crear la Tabla de Auditoría

```
-- Crear la tabla de auditoría
CREATE TABLE [Univerdadddd].[dbo].[Auditoria$] (
    ID INT IDENTITY(1,1) PRIMARY KEY, -- ID auto-incremental
    USUARIO INT, -- ID del usuario
    FECHA_REGISTRO DATETIME, -- Fecha de registro del evento
    ACCION VARCHAR(50) -- Acción realizada (en este caso, INSERT)
);
```

100 %

Messages

Commands completed successfully.

Completion time: 2024-11-21T12:02:49.2187186-05:00

2. Crear el Disparador

```
--2.
-- Crear el disparador (trigger) para insertar automáticamente en la tabla de auditoría
CREATE TRIGGER InsertarAuditoriaUsuarios
ON [Univerdadddd].[dbo].[Usuarios$]
AFTER INSERT
AS
BEGIN
    -- Insertar un nuevo registro en la tabla de Auditoría
    INSERT INTO [Univerdadddd].[dbo].[Auditoria$] (USUARIO, FECHA_REGISTRO, ACCION)
    SELECT
        USUARIO, -- ID del usuario insertado
        GETDATE(), -- Fecha y hora del evento
        'INSERT' -- Descripción de la acción (en este caso, 'INSERT')
    FROM inserted; -- La tabla virtual "inserted" contiene los registros recién insertados
END;
```

%

Messages

Commands completed successfully.

Completion time: 2024-11-21T12:04:15.4515607-05:00

3. Probar el Disparador

```
--3.
-- Insertar un nuevo usuario
INSERT INTO [Univerdadddd].[dbo].[Usuarios$] ([USUARIO], [GÉNERO], [FECHA NAC], [EDAD], [NACIONALIDAD], [EDO# ORIGEN])
VALUES (5, 'F', '1995-10-25', 29, 'MEXICANA', 'YUCATÁN');
-- Verificar si el disparador insertó el registro en la tabla de auditoría
SELECT * FROM [Univerdadddd].[dbo].[Auditoria$];
```

ID	USUARIO	FECHA_REGISTRO	ACCION
1	5	2024-11-21 12:04:57.853	INSERT

7) Implemente y explique un Script para crear un disparador para elimine un registro automáticamente en una tabla de su base de datos.

```
-- Crear el disparador (trigger) para eliminar automáticamente en la tabla de auditoría
CREATE TRIGGER EliminarAuditoriaUsuarios
ON [Univerdadddd].[dbo].[Usuarios$]
AFTER DELETE
AS
BEGIN
    -- Eliminar los registros correspondientes en la tabla de Auditoría
    DELETE FROM [Univerdadddd].[dbo].[Auditoria$]
    WHERE USUARIO IN (SELECT USUARIO FROM deleted);
END;
GO
```

100 %

Messages

Commands completed successfully.

Completion time: 2024-11-21T12:09:47.4725975-05:00

Probar el Disparador

```
-- Eliminar un usuario de la tabla Usuarios$
DELETE FROM [Univerdadddd].[dbo].[Usuarios$]
WHERE USUARIO = 5;
-- Verificar si el registro fue eliminado de la tabla de auditoría
SELECT * FROM [Univerdadddd].[dbo].[Auditoria$];
```

ID	USUARIO	FECHA_REGISTRO	ACCION
----	---------	----------------	--------

8) Implemente y explique un Script para crear un disparador para actualice un registro automáticamente en una tabla de su base de datos.

```
-- Crear el disparador (trigger) para actualizar automáticamente en la tabla de auditoría
CREATE TRIGGER ActualizarAuditoriaUsuarios
ON [Univerdadddd].[dbo].[Usuarios$]
AFTER UPDATE
AS
BEGIN
    -- Actualizar el registro en la tabla de Auditoría cuando se actualice un usuario
    UPDATE a
    SET a.FECHA_REGISTRO = GETDATE() -- Actualizar la fecha de registro
    FROM [Univerdadddd].[dbo].[Auditoria$] a
    INNER JOIN inserted i ON a.USUARIO = i.USUARIO -- Relacionar con el usuario actualizado
    WHERE i.USUARIO IN (SELECT USUARIO FROM inserted);
END;
GO
```

100 %

Messages

Commands completed successfully.

Completion time: 2024-11-21T12:12:48.4002570-05:00

Probar el Disparador

```
-- Actualizar el usuario con USUARIO = 2
UPDATE [Univerdadddd].[dbo].[Usuarios$]
SET [GÉNERO] = 'F', [FECHA NAC] = '2000-03-05', [EDAD] = 24
WHERE [USUARIO] = 2;
-- Verificar si la fecha de registro en la tabla de auditoría fue actualizada
SELECT * FROM [Univerdadddd].[dbo].[Auditoria$]
WHERE USUARIO = 2;
```

%

Results Messages

ID	USUARIO	FECHA_REGISTRO	ACCION
----	---------	----------------	--------

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

```
CREATE TRIGGER trg_ValidarEdad
ON [Usuarios$]
FOR INSERT, UPDATE
AS
BEGIN
    -- Verifica que la edad esté en el rango permitido
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE EDAD < 0 OR EDAD > 120
    )
    BEGIN
        -- Si la edad está fuera de rango, cancela la operación y muestra un error
        RAISERROR ('La edad debe estar entre 0 y 120 años.', 16, 1);
        ROLLBACK TRANSACTION; -- Deshace la transacción
    END
END;

) %
Messages
Commands completed successfully.

Completion time: 2024-11-21T19:48:10.7124815-05:00
```

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

SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))

```
-- Crear logins a nivel de servidor
CREATE LOGIN Erick WITH PASSWORD = 'ContraseñaErick123!';
CREATE LOGIN Percy WITH PASSWORD = 'ContraseñaPercy456!';
CREATE LOGIN Brayan WITH PASSWORD = 'ContraseñaBrayan789!';

-- Usar la base de datos donde se crearán los usuarios
USE [exel base de datos];

-- Crear usuarios a nivel de base de datos asociados a los logins
CREATE USER Erick FOR LOGIN Erick;
CREATE USER Percy FOR LOGIN Percy;
CREATE USER Brayan FOR LOGIN Brayan;

-- (Opcional) Otorgar permisos básicos a los usuarios
ALTER ROLE db_datareader ADD MEMBER Erick;
ALTER ROLE db_datareader ADD MEMBER Percy;
ALTER ROLE db_datareader ADD MEMBER Brayan;

-- Verificar los usuarios creados
SELECT name AS Usuario, create_date AS FechaCreacion
FROM sys.database_principals
WHERE type = 'S' AND name IN ('Erick', 'Percy', 'Brayan');
```

100 %

Results Messages

	Usuario	FechaCreacion
1	Erick	2024-11-21 22:55:40.003
2	Percy	2024-11-21 22:55:40.003
3	Brayan	2024-11-21 22:55:40.003

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

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Realizar una copia de seguridad de la base de datos original
BACKUP DATABASE [exel base de datos]
TO DISK = 'C:\Backups\exel_base_datos.bak'
WITH FORMAT, MEDIANAME = 'SQLServerBackups', NAME = 'Full Backup of exel base de datos';

-- Restaurar la base de datos con un nuevo nombre
RESTORE DATABASE [exel_base_datos_copia]
FROM DISK = 'C:\Backups\exel_base_datos.bak'
WITH MOVE 'exel base de datos' TO 'C:\SQLData\exel_base_datos_copia.mdf',
MOVE 'exel base de datos_log' TO 'C:\SQLData\exel_base_datos_copia_log.ldf',
REPLACE;

USE [exel_base_datos_copia]; -- Base de datos copiada

-- Crear usuarios si no existen
CREATE USER Erick FOR LOGIN Erick;
CREATE USER Percy FOR LOGIN Percy;
CREATE USER Brayan FOR LOGIN Brayan;

-- Asignar permisos (por ejemplo, acceso de solo lectura)
ALTER ROLE db_datareader ADD MEMBER Erick;
ALTER ROLE db_datareader ADD MEMBER Percy;
ALTER ROLE db_datareader ADD MEMBER Brayan;
```

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

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Realizar una copia de seguridad de la base de datos
BACKUP DATABASE [exel base de datos]
TO DISK = 'C:\Backups\exel_base_datos.bak'
WITH FORMAT, NAME = 'Backup de exel base de datos';

100 %
Messages
Commands completed successfully.

Completion time: 2024-11-21T23:07:51.4746305-05:00
```

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

```

SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
USE [exel base de datos];
GO
-- 1. Crear una clave maestra
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'TuContraseñaSegura!';
GO
-- 2. Crear un certificado
CREATE CERTIFICATE CertificadoUsuarios
WITH SUBJECT = 'Certificado para encriptar datos de Usuarios';
GO
-- 3. Crear una clave simétrica usando el certificado
CREATE SYMMETRIC KEY ClaveSimetricaUsuarios
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE CertificadoUsuarios;
GO
-- 4. Abrir la clave simétrica para encriptar datos
OPEN SYMMETRIC KEY ClaveSimetricaUsuarios
DECRYPTION BY CERTIFICATE CertificadoUsuarios;
-- 5. Actualizar la columna 'EDAD' para encriptar los datos
UPDATE dbo.Usuarios$
SET EDAD = ENCRYPTBYKEY(KEY_GUID('ClaveSimetricaUsuarios'), CAST(EDAD AS NVARCHAR(10)));
GO
-- 6. Cerrar la clave simétrica
CLOSE SYMMETRIC KEY ClaveSimetricaUsuarios;
GO

```

14) Utilizando un script, aplique la seguridad a nivel de columna, restringiendo el acceso a la columna que contiene la clave primaria de una de las tablas de su base de datos

```

SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- 1. Crear la vista que excluye la columna de clave primaria (ID_USUARIO)
CREATE VIEW dbo.vw_Usuarios_Sin_ID
AS
SELECT
    USUARIO,
    GÉNERO,
    FECHA_NAC,
    EDAD,
    NACIONALIDAD
FROM dbo.Usuarios$;
GO
-- 2. Denegar acceso a la columna de clave primaria (ID_USUARIO) en la tabla original
DENY SELECT ON dbo.Usuarios$(ID_USUARIO) TO [UsuarioNoAutorizado];
GO
-- 3. Otorgar acceso a la vista para los usuarios no autorizados a ver la clave primaria
GRANT SELECT ON dbo.vw_Usuarios_Sin_ID TO [UsuarioNoAutorizado];
GO
-- 4. Otorgar acceso completo a la tabla original para los administradores (o usuarios autorizados)
GRANT SELECT ON dbo.Usuarios$ TO [Administrador];
GO

```

TABLE_NAME
Usuarios\$_xlnm#_FilterDatabase

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

SQLQuery9.sql - LA...PC13\USER 17 (56)* SQLQuery7.sql - LA...PC13\USER 17 (71) SQLQuery8.sql - LA...PC13\USER 17 (72)

```
-- Paso 1: Comprobar si la vista existe y eliminarla si es así
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'dbo.Vw_Usuarios_Seguro') AND type = 'V')
BEGIN
    DROP VIEW dbo.Vw_Usuarios_Seguro;
END
GO

-- Paso 2: Crear la Vista con las Columnas Permitidas (sin la columna 'EDAD')
CREATE VIEW dbo.Vw_Usuarios_Seguro AS
SELECT [USUARIO],
       [GÉNERO],
       [FECHA_NAC],
       [NACIONALIDAD]
FROM [exel base de datos].[dbo].[Usuarios$];
GO

-- Paso 3: Revocar acceso a la tabla original para el usuario
REVOKE SELECT ON [exel base de datos].[dbo].[Usuarios$] TO [Usuario_Con_Acceso];
GO

-- Paso 4: Otorgar acceso a la vista (que no tiene la columna 'EDAD') al usuario
GRANT SELECT ON dbo.Vw_Usuarios_Seguro TO [Usuario_Con_Acceso];
GO

-- Paso 5: Verificar los permisos otorgados al usuario
SELECT *
FROM sys.database_permissions
WHERE grantee_principal_id = USER_ID('Usuario_Con_Acceso');
GO
```

100 %

Results Messages

	class	class_desc	major_id	minor_id	grantee_principal_id	grantor_principal_id	type	permission_name	state	state_desc
1	0	DATABASE	0	0	6	1	CO	CONNECT	G	GRANT
2	1	OBJECT_OR_COLUMN	1317579732	0	6	1	SL	SELECT	G	GRANT

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

```
SQLQuery2.sql - D...3GC\USUARIO (75)) * X SQLQuery1.sql - D...3GC\USUARIO (80))
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'ContraseñaSegura123'; -- Cambia la contraseña
GO
-- Paso 2: Crear un certificado para el cifrado
CREATE CERTIFICATE TDE_Certificate
WITH SUBJECT = 'TDE Certificate';
GO
-- Paso 3: Crear la clave de cifrado de la base de datos
USE [exel base de datos]; -- Cambia al nombre de tu base de datos
GO
CREATE DATABASE ENCRYPTION KEY;
GO
-- Paso 4: Habilitar el cifrado de la base de datos
ALTER DATABASE [exel base de datos]
SET ENCRYPTION ON;
GO
-- Paso 5: Respalidar el certificado para futuras restauraciones
BACKUP CERTIFICATE TDE_Certificate
TO FILE = 'C:\Backups\TDE_Certificate.cer'
WITH PRIVATE KEY (
FILE = 'C:\Backups\TDE_CertificatePrivateKey.pvk',
ENCRYPTION BY PASSWORD = 'ContraseñaSegura123'); -- Cambia la contraseña
GO
-- Verificar si el cifrado está habilitado
SELECT
    database_id,
    encryption_state
FROM sys.database_encryption_keys;
GO
```

17) 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.

```
-- Asumiendo que el usuario ya existe, asignamos permisos
USE [exel base de datos]; -- Asegúrate de usar el nombre de tu base de datos
GO

-- Conceder permisos al usuario
GRANT SELECT, INSERT, UPDATE, DELETE ON DATABASE::[exel base de datos] TO [Brayan];
GO
```

18) Utilizando un Scripts realice la validación y filtración de entradas del usuario para evitar caracteres maliciosos (Ejemplo: ', --, ;')

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
CREATE FUNCTION dbo.fn_SanitizeInput (@InputString NVARCHAR(MAX))
RETURNS NVARCHAR(MAX)
AS
BEGIN
    -- Reemplazar caracteres maliciosos
    SET @InputString = REPLACE(@InputString, ''', '');
    SET @InputString = REPLACE(@InputString, '--', '');
    SET @InputString = REPLACE(@InputString, ';', '');
    SET @InputString = REPLACE(@InputString, '<', '');
    SET @InputString = REPLACE(@InputString, '>', '');

    RETURN @InputString;
END;
GO

100 %
Messages
Commands completed successfully.

Completion time: 2024-11-21T23:16:34.9630763-05:00
```

19) Realice un script que verifiquen que los datos ingresados cumplan con formatos esperados (ej.: números en lugar de texto, longitud máxima).

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Verificar si un número es válido
CREATE FUNCTION dbo.fn_ValidateNumericInput (@InputString NVARCHAR(MAX))
RETURNS BIT
AS
BEGIN
    -- Verificar si la entrada es un número
    IF (@InputString LIKE '%[^0-9]%')
        RETURN 0; -- No es un número válido
    ELSE
        RETURN 1; -- Es un número válido
END;
GO
```

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


```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Crear una auditoría
CREATE SERVER AUDIT Audit_Exel
TO FILE (FILEPATH = 'C:\Backups\AuditLogs\');
GO

-- Crear una especificación de auditoría
CREATE SERVER AUDIT SPECIFICATION AuditSpec_Exel
FOR SERVER AUDIT Audit_Exel
ADD (FAILED_LOGIN_GROUP),
ADD (SUCCESSFUL_LOGIN_GROUP)
WITH (STATE = ON);
GO
```

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

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Configurar la memoria máxima del servidor SQL
EXEC sp_configure 'max server memory (MB)', 4096; -- Ajusta a 4GB de memoria máxima
RECONFIGURE;
GO
```

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

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Generar copia de seguridad de la base de datos
BACKUP DATABASE [exel base de datos]
TO DISK = 'C:\Backups\exel_basedatos.bak'
WITH INIT, SKIP;
GO
```

23) Realice un script para programar backups automatizados de su base de datos

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Crear un trabajo de SQL Server Agent para programar backups automatizados
EXEC msdb.dbo.sp_add_job
    @job_name = 'Backup_Exel',
    @enabled = 1;

EXEC msdb.dbo.sp_add_jobstep
    @job_name = 'Backup_Exel',
    @step_name = 'BackupStep',
    @subsystem = 'TSQL',
    @command = 'BACKUP DATABASE [exel base de datos] TO DISK = ''C:\Backups\exel_basedatos.bak'';',
    @database_name = 'master';

EXEC msdb.dbo.sp_add_schedule
    @schedule_name = 'DailyBackupSchedule',
    @enabled = 1,
    @freq_type = 4, -- Diario
    @freq_interval = 1, -- Cada 1 día
    @active_start_time = 020000; -- A las 2 AM

EXEC msdb.dbo.sp_attach_schedule
    @job_name = 'Backup_Exel',
    @schedule_name = 'DailyBackupSchedule';

GO
```

100 %

Messages

Commands completed successfully.

Completion time: 2024-11-21T23:18:15.1322136-05:00

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

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Restaurar la base de datos desde un archivo .bak
RESTORE DATABASE [exel base de datos]
FROM DISK = 'C:\Backups\exel_basedatos.bak'
WITH REPLACE;
GO
```

25) Utilizando un script, cree un espejo de la base de datos

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Iniciar el espejo de la base de datos
ALTER DATABASE [exel base de datos]
SET PARTNER = 'TCP://ServidorEspejo:5022'; -- Cambia la dirección IP y el puerto
GO
```

26) Utilizando un script, para enviar datos a la base de datos espejo creada

Una vez que el espejo está configurado, los datos se sincronizan automáticamente entre la base de datos principal y la base de datos espejo.

27) Utilizando un script, de permiso a un usuario por un determinado tiempo

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO (80))
-- Crear un permiso temporal
CREATE LOGIN [UsuarioTemporal] WITH PASSWORD = 'ContraseñaSegura123';
GO

-- Dar permisos solo por un tiempo determinado
GRANT SELECT ON [exel base de datos] TO [UsuarioTemporal];
EXEC xp_logininfo 'UsuarioTemporal', 'permissions';
GO
```

28) Utilizando un script, realice la replicación de bases de datos

```
SQLQuery2.sql - D...3GC\USUARIO (75))* X SQLQuery1.sql - D...3GC\USUARIO
-- Crear publicación en la base de datos
EXEC sp_replicationdboption
    @dbname = 'exel base de datos',
    @optname = 'publish',
    @value = 'true';
GO
```

29) Explique que es Always On Availability Groups

Always On Availability Groups es una característica de SQL Server para alta disponibilidad y recuperación ante desastres. Permite la replicación de bases de datos entre servidores para asegurar la disponibilidad continua, incluso si un servidor falla.

- **Características clave:**
 1. Conmutación por error automática.
 2. Sincronización de datos entre servidores primarios y secundarios.
 3. Permite réplicas de solo lectura para mejorar el rendimiento.
 4. Requiere SQL Server Enterprise Edition.
- **Funcionamiento:**
 1. Se configura un grupo de disponibilidad con bases de datos replicadas entre el servidor primario y uno o más secundarios.
 2. Si el primario falla, uno de los secundarios toma el control automáticamente.

30) Explique que es Log Shipping

Log Shipping es una solución para mantener una copia de seguridad de la base de datos en un servidor secundario mediante la copia y restauración periódica de logs de transacciones.

- **Características clave:**

1. Copia de seguridad y restauración de logs de transacciones.
2. Conmutación por error manual (sin automatización).
3. Puede tener múltiples servidores secundarios.

- **Funcionamiento:**

1. Realiza una copia de seguridad de los logs en el servidor primario.
2. Copia los logs al servidor secundario.
3. Restaura los logs en el servidor secundario para mantener la base de datos actualizada.