



“Año De La Recuperación Y
Consolidación De La Economía Peruana”

UNIVERSIDAD PERUANA LOS ANDES

“FACULTAD DE INGENIERÍA”

ESCUELA PROFESIONAL “SISTEMAS Y
COMPUTACIÓN”

CÁTEDRA: Base de Datos II

CATEDRÁTICO: Ing. Fernandez Bejarano Raul Enrique

ESTUDIANTE: Vega Brañez Samuel Max

CICLO: V

SECCIÓN: B1

HUANCAYO PERÚ

2025

Manual de Seguridad y Control de Acceso

Base de Datos: TechGadget Store

El siguiente código SQL crea la base de datos `TechGadgetStore`, la tabla `Empleados`, y la rellena con 10 registros ficticios. La columna clave de seguridad es `PIN_Acceso`, que usa `HASHBYTES` para el **Cifrado y Protección de Datos**.

SQL

```
-- 1. CREACIÓN DE LA BASE DE DATOS Y TABLA
CREATE DATABASE TechGadgetStore;
GO
USE TechGadgetStore;
GO

CREATE TABLE Empleados (
    ID_Emppleado INT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Cargo VARCHAR(50) NOT NULL,
    Salario DECIMAL(10, 2) NOT NULL,
    PIN_Acceso VARBINARY(256), -- Almacena el hash SHA2_256 del PIN
    FechaContratacion DATE NOT NULL,
    Email VARCHAR(100) UNIQUE,
    ID_Supervisor INT,
    EstadoCuenta VARCHAR(10) DEFAULT 'Activo',
    UltimoLogin DATETIME -- Para monitoreo y auditoría
);
GO

-- 2. INSERCIÓN DE 10 DATOS FICTICIOS
INSERT INTO Empleados (ID_Emppleado, Nombre, Cargo, Salario,
PIN_Acceso, FechaContratacion, Email, ID_Supervisor, EstadoCuenta,
UltimoLogin) VALUES
(100, 'Ana Gómez', 'Gerente', 6500.00, HASHBYTES('SHA2_256',
'Gerente2025'), '2020-01-15', 'ana.g@tech.com', NULL, 'Activo',
GETDATE()),
(101, 'Beto Ruiz', 'Vendedor', 3200.00, HASHBYTES('SHA2_256',
'VentasBeto'), '2021-05-20', 'beto.r@tech.com', 100, 'Activo',
GETDATE()),
(102, 'Carla Diaz', 'Vendedor', 3150.00, HASHBYTES('SHA2_256',
'VentasCarla'), '2022-08-10', 'carla.d@tech.com', 100, 'Activo',
GETDATE()),
(103, 'David Soto', 'Almacén', 2800.00, HASHBYTES('SHA2_256',
'AlmacenD1'), '2023-03-01', 'david.s@tech.com', 105, 'Activo',
GETDATE()),
(104, 'Elena Paz', 'Soporte', 3500.00, HASHBYTES('SHA2_256',
'SoporteE5'), '2021-11-25', 'elena.p@tech.com', 100, 'Activo',
GETDATE()),
(105, 'Felipe Rey', 'Subgerente', 4800.00, HASHBYTES('SHA2_256',
'Subgerente'), '2019-10-01', 'felipe.r@tech.com', 100, 'Activo',
GETDATE()),
(106, 'Gaby Mora', 'Vendedor', 3000.00, HASHBYTES('SHA2_256',
'VentasGaby'), '2023-12-05', 'gaby.m@tech.com', 101, 'Activo',
GETDATE()),
```

```

(107, 'Hugo León', 'Almacén', 2750.00, HASHBYTES('SHA2_256',
'AlmacenH2'), '2024-01-18', 'hugo.l@tech.com', 105, 'Activo',
GETDATE()),
(108, 'Inés Cruz', 'Vendedor', 3100.00, HASHBYTES('SHA2_256',
'VentasInes'), '2022-04-01', 'ines.c@tech.com', 102, 'Activo',
GETDATE()),
(109, 'Juan Vidal', 'Soporte', 3450.00, HASHBYTES('SHA2_256',
'SoporteJ6'), '2023-07-07', 'juan.v@tech.com', 104, 'Inactivo',
GETDATE());

```

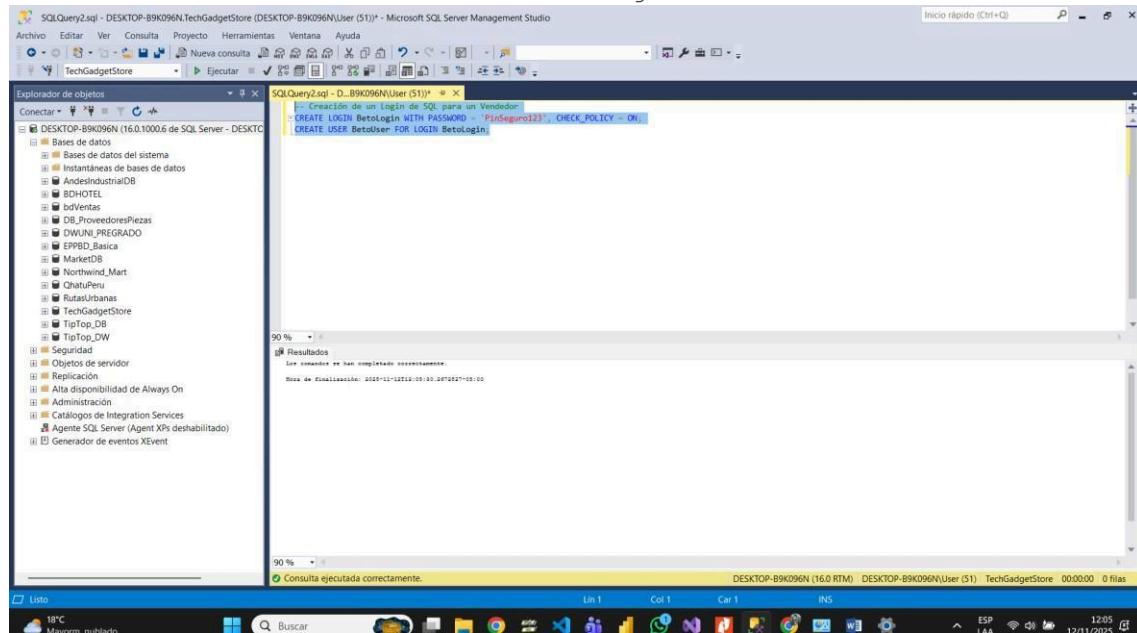
Aplicación de Seguridad y Control de Acceso (Semana 11)

1. Autenticación SQL y Windows (diferencias y prácticas seguras)

- Enunciado:** Se debe diferenciar el tipo de autenticación según el nivel de responsabilidad para mejorar la seguridad y la gestión de credenciales.
- Explicación:** Se recomienda usar la **Autenticación de Windows** (más segura y ligada al dominio) para Gerentes y Administradores (Ana Gómez). La **Autenticación de SQL Server** se reserva para usuarios de aplicaciones o de bajo privilegio (Beto Ruiz), ya que sus credenciales solo existen dentro del servidor SQL.
- Código Ejemplo:**

SQL

```
-- Creación de un Login de SQL para un Vendedor
CREATE LOGIN BetoLogin WITH PASSWORD = 'PinSeguro123', CHECK_POLICY
= ON;
CREATE USER BetoUser FOR LOGIN BetoLogin;
```



2. Cuentas de Servicio y configuración del Servidor

- **Enunciado:** Los servicios críticos de SQL Server deben ejecutarse con cuentas de mínimo privilegio para limitar el daño en caso de compromiso.
- **Explicación:** Es una **práctica segura** configurar el servicio del motor de base de datos (**MSSQLSERVER**) con una **Cuenta de Servicio Administrada (MSA)** de Windows, no con la cuenta `Local System` ni con una cuenta de dominio de administrador. Esto garantiza que si el servicio es atacado, el atacante no obtendrá acceso completo al sistema operativo.

3. Creación de roles fijos y personalizados

- **Enunciado:** Agrupar permisos en roles personalizados para mantener el **Principio del Mínimo Privilegio** y simplificar la administración.
- **Explicación:** Se crean roles específicos para cada función de la tienda. Los roles fijos de BD (`db_datareader`) son demasiado amplios. Los roles personalizados permiten una gestión precisa, donde un nuevo empleado hereda automáticamente los permisos de su rol.
- **Código Ejemplo:**

SQL

```
-- Rol Personalizado para Vendedores
CREATE ROLE Rol_Ventas_Tech;
ALTER ROLE Rol_Ventas_Tech ADD MEMBER BetoUser;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, there is a query window titled "SQLQuery2.sql - DESKTOP-B9K096N.TechGadgetStore (DESKTOP-B9K096N\user (S1))". The query contains the following T-SQL code:

```
-- Creación de un Login de SQL para un Vendedor
CREATE LOGIN BetoLogin WITH PASSWORD = 'PinSeguro123', CHECK_POLICY = ON;
CREATE USER BetoUser FOR LOGIN BetoLogin;

-- Rol Personalizado para Vendedores
CREATE ROLE Rol_Ventas_Tech;
ALTER ROLE Rol_Ventas_Tech ADD MEMBER BetoUser;
```

Below the query window, the status bar displays "Consulta ejecutada correctamente." and the execution time "00:00:00 0 filas".

4 Control de Acceso: GRANT, DENY y REVOKE

1. GRANT (Conceder Permiso)

El comando `GRANT` se utiliza para **otorgar** un permiso a un usuario o a un rol.

- Enunciado:** El rol de ventas necesita ver la información básica de todos los empleados para coordinar turnos y tareas (ID, Nombre, Cargo).
- Acción:** Conceder el permiso de lectura (`SELECT`) sobre la tabla `Empleados`.

SQL

```
-- Conceder el permiso SELECT (lectura) de la tabla Empleados al
-- Rol_Ventas_Tech
GRANT SELECT ON Empleados TO Rol_Ventas_Tech;
```

- Resultado:** Cualquier usuario asignado a `Rol_Ventas_Tech` (como Beto Ruiz) puede ejecutar `SELECT * FROM Empleados` y ver todas las columnas.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, there is a query window with the following SQL code:

```
-- Conceder el permiso SELECT (lectura) de la tabla Empleados al Rol_Ventas_Tech
GRANT SELECT ON Empleados TO Rol_Ventas_Tech;
```

Below the code, a message box displays: "Los comandos se han ejecutado correctamente." (The commands were executed successfully.) At the bottom of the screen, the status bar shows: "Consulta ejecutada correctamente." (Query executed successfully.)

2. DENY (Negar Permiso)

El comando `DENY` se utiliza para **denegar explícitamente** un permiso. Es crucial porque **DENY anula cualquier GRANT existente**.

- Enunciado:** La información de **Salario** es altamente sensible y solo debe ser visible para Gerentes y Subgerentes.
- Acción:** Negar el permiso de lectura (`SELECT`) específicamente sobre la columna `Salario` al rol de ventas.

SQL

```
-- Negar el permiso SELECT solo en la columna Salario para el
-- Rol_Ventas_Tech
-- Esto PREVALECE sobre el GRANT SELECT que se dio a toda la tabla.
DENY SELECT ON Empleados (Salario) TO Rol_Ventas_Tech;
```

- **Resultado:** Si Beto Ruiz intenta ejecutar `SELECT Salario FROM Empleados`, el sistema le arrojará un error de permisos. Si ejecuta `SELECT ID_Emppleado, Nombre, Cargo, Salario FROM Empleados`, verá un error o un valor nulo para la columna `Salario` (dependiendo de la configuración del entorno, pero el acceso será denegado).

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery2.sql - DESKTOP-B9K096N.TechGadgetStore (DESKTOP-B9K096N\User (51)) - Microsoft SQL Server Management Studio". The menu bar includes Archivo, Editar, Ver, Proyecto, Herramientas, Ventana, Ayuda. The toolbar has icons for New Query, New Script, Run, Stop, and Refresh. The status bar at the bottom right shows "Inicio rápido (Ctrl+Q)" and the connection details "DESKTOP-B9K096N (16.0 RTM) - DESKTOP-B9K096N\User (51) - TechGadgetStore 00:00:00 0 filas".

Explorador de objetos

SQLQuery2.sql - D_B9K096N\User (51) -

```
-- Creación de un Login de SQL para un Vendedor
CREATE LOGIN Betologin WITH PASSWORD = 'PinSeguro123', CHECK_POLICY = ON;
CREATE USER Betouser FOR LOGIN Betologin;
--consulta tres
-- Rol Personalizado para Vendedores
CREATE ROLE Rol_Ventas_Tech;
ALTER ROLE Rol_Ventas_Tech ADD MEMBER Betouser;
-- ultima consulta
-- Configuración para registrar todos los intentos de login fallidos
CREATE SERVER AUDIT SPECIFICATION ServerSpec_TechGadget
FOR SERVER AUDIT Audit_TechGadget
ADD FILELOGON_GROUP();
ALTER SERVER AUDIT SPECIFICATION ServerSpec_TechGadget WITH (STATE = ON);

-- Conceder el permiso SELECT (lectura) de la tabla Empleados al Rol_Ventas_Tech
GRANT SELECT ON Empleados TO Rol_Ventas_Tech;

-- Negar el permiso SELECT solo en la columna Salario para el Rol_Ventas_Tech
-- Esto PREVALECE sobre el GRANT SELECT que se dio a toda la tabla.
DENY SELECT ON Empleados (Salario) TO Rol_Ventas_Tech;
```

90 %

g) Resultados

Los comandos se han completado correctamente.

Hora de finalización: 2020-11-12T12:12:14.3987469-09:00

90 %

Consulta ejecutada correctamente.

3. REVOKE (Revocar/Eliminar Permiso)

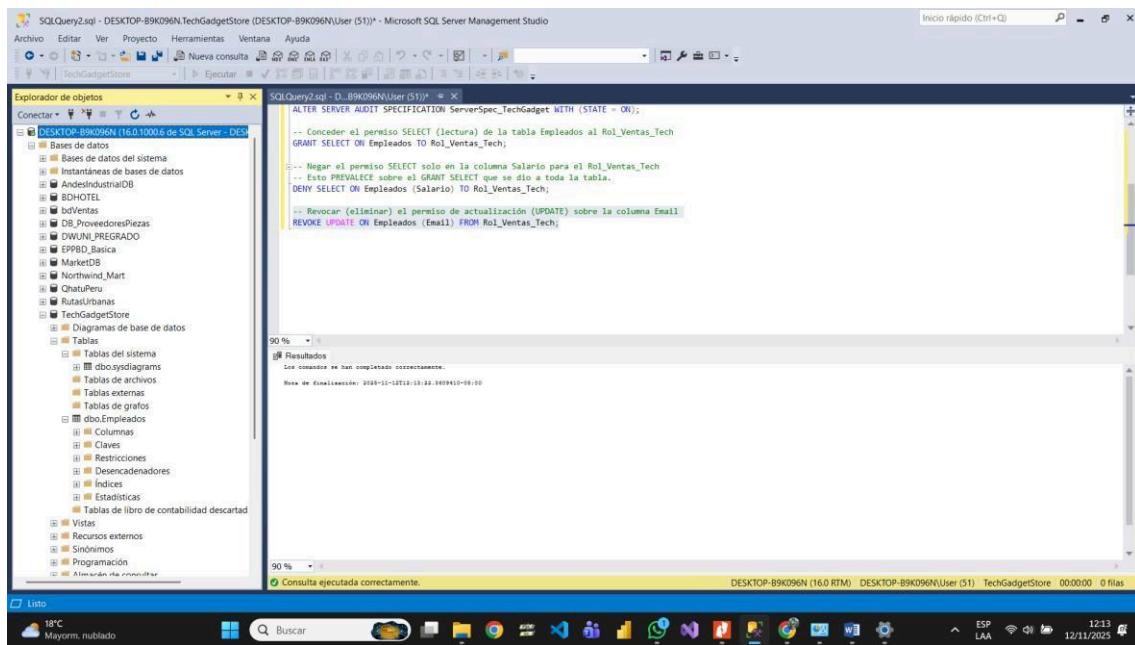
El comando `REVOKE` se utiliza para **eliminar un permiso previamente concedido (GRANT) o negado (DENY)**. Si se revoca un `GRANT`, el usuario vuelve a su estado de permiso predeterminado.

- **Enunciado:** Inicialmente, se le concedió a los Vendedores la capacidad de actualizar el Email de los empleados por error. Se necesita eliminar este permiso.
 - **Acción:** Revocar el permiso de modificación (UPDATE) sobre la columna Email.

SQL

```
-- Revocar (eliminar) el permiso de actualización (UPDATE) sobre la  
columna Email  
REVOKE UPDATE ON Empleados (Email) FROM Rol_Ventas_Tech;
```

- **Resultado:** Si el permiso UPDATE ON Empleados (Email) había sido concedido antes, ahora es eliminado. El Rol Ventas Tech ya no puede modificar esa columna.



En resumen:

- **GRANT** construye permisos.
- **DENY** bloquea permisos (y siempre gana).
- **REVOKE** elimina la concesión o la negación.

5 Cifrado y Protección de Datos

El objetivo principal de esta sección es **proteger la confidencialidad e integridad** de los datos sensibles de la empresa, tanto las credenciales de los empleados como cualquier información financiera o personal.

Enunciado del Caso Práctico

Se deben proteger las credenciales de acceso de los empleados (PIN_Acceso) utilizando técnicas de cifrado unidireccional (hashing) para asegurar que, en caso de una brecha de seguridad, las contraseñas reales no puedan ser reveladas.

Explicación y Aplicación

En la base de datos TechGadgetStore, aplicamos la técnica de **Hashing** durante la inserción de datos en la columna PIN_Acceso.

1. Hashing (Cifrado Unidireccional)

El *hashing* es el estándar de la industria para proteger contraseñas. Convierte la contraseña original (el PIN, en este caso) en una cadena de caracteres de longitud fija (el *hash*). Este proceso es **unidireccional**, lo que significa que es casi imposible revertir el *hash* para obtener el PIN original.

- **¿Por qué se usa?** Si un atacante roba la base de datos, solo obtendrá los *hashes*, no los PINs reales, haciendo que la información sea inútil para iniciar sesión.

2. Implementación en SQL

Usamos la función `HASHBYTES` con el algoritmo fuerte **SHA2_256** al momento de insertar o actualizar el PIN de un empleado.

- **Columna afectada:** `PIN_Acceso` (definida como `VARBINARY(256)` para almacenar el resultado del hash).
- **Función utilizada:** `HASHBYTES('SHA2_256', 'ContraseñaOriginal')`.

SQL

```
-- Ejemplo de la inserción de datos para el PIN de Ana Gómez
INSERT INTO Empleados (... , PIN_Acceso, ...) VALUES
(100, ... , HASHBYTES('SHA2_256', 'Gerente2025'), ...);
```

```
-- Ejemplo de cómo actualizar el PIN de un empleado:
UPDATE Empleados
SET PIN_Acceso = HASHBYTES('SHA2_256', 'NuevoPIN123')
WHERE ID_Emppleado = 101;
```

3. Cifrado Adicional (Recomendaciones)

Aunque el *hashing* protege las contraseñas, la **protección de datos** puede extenderse a otras áreas:

- **TDE (Transparent Data Encryption):** Cifra toda la base de datos en el disco. Esto protege los datos si los archivos físicos de la base de datos son robados del servidor.
- **Cifrado a Nivel de Columna:** Si existiera una columna de información financiera o personal altamente regulada (como una cuenta bancaria), se podría usar `ENCRYPTIONBYKEY` para cifrar solo esa columna, asegurando que solo las aplicaciones o usuarios con la clave de cifrado puedan acceder al dato.

Resumen de la Protección

Concepto	Aplicación en el Script	Beneficio de Seguridad
Hashing	Uso de <code>HASHBYTES('SHA2_256', ...)</code>	Protege credenciales. Evita que las contraseñas sean legibles incluso para administradores de bases de datos.
Tipo de Dato	<code>VARBINARY(256)</code>	Asegura que el <i>hash</i> (que no es texto) se almacene correctamente.

Aquí tienes el código para implementar **TDE (Transparent Data Encryption)** en la base de datos `TechGadgetStore`.

5 Cifrado de la Base de Datos con TDE

TDE cifra toda la base de datos en el disco (datos en reposo), protegiendo la información si alguien accede a los archivos físicos de la BD.

Paso 1: Crear la Clave Maestra de la Base de Datos (Master Key)

Esta clave protege las claves de cifrado que crearemos a continuación.

SQL

```
USE master;
-- Crear una Clave Maestra de Servicio (Service Master Key) si no existe.
-- Esta es la raíz de la jerarquía de cifrado.
-- Normalmente ya existe, pero se incluye por completitud.
-- ALTER SERVICE MASTER KEY FORCE REGENERATE;
GO
```

Paso 2: Crear un Certificado o Clave Asimétrica

Usaremos un certificado para proteger la llave de cifrado de la base de datos.

SQL

```
USE master;
-- Crear un certificado que se usará para cifrar la clave de la base de datos.
CREATE CERTIFICATE TechGadget_Cert
WITH SUBJECT = 'Certificado de Cifrado para TechGadgetStore';
GO
```

Paso 3: Crear la Clave de Cifrado de la Base de Datos (Database Encryption Key - DEK)

Esta es la clave simétrica que realmente cifra los datos y está protegida por el certificado que creamos en el paso anterior.

SQL

```
USE TechGadgetStore;
-- La clave de cifrado de la base de datos será protegida por TechGadget_Cert.
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE TechGadget_Cert;
GO
```

Paso 4: Habilitar TDE en la Base de Datos

Este comando inicia el proceso de cifrado de todos los archivos de datos (.mdf) y de registro (.ldf) de la base de datos TechGadgetStore.

SQL

```
USE master;
ALTER DATABASE TechGadgetStore
SET ENCRYPTION ON;
GO
```

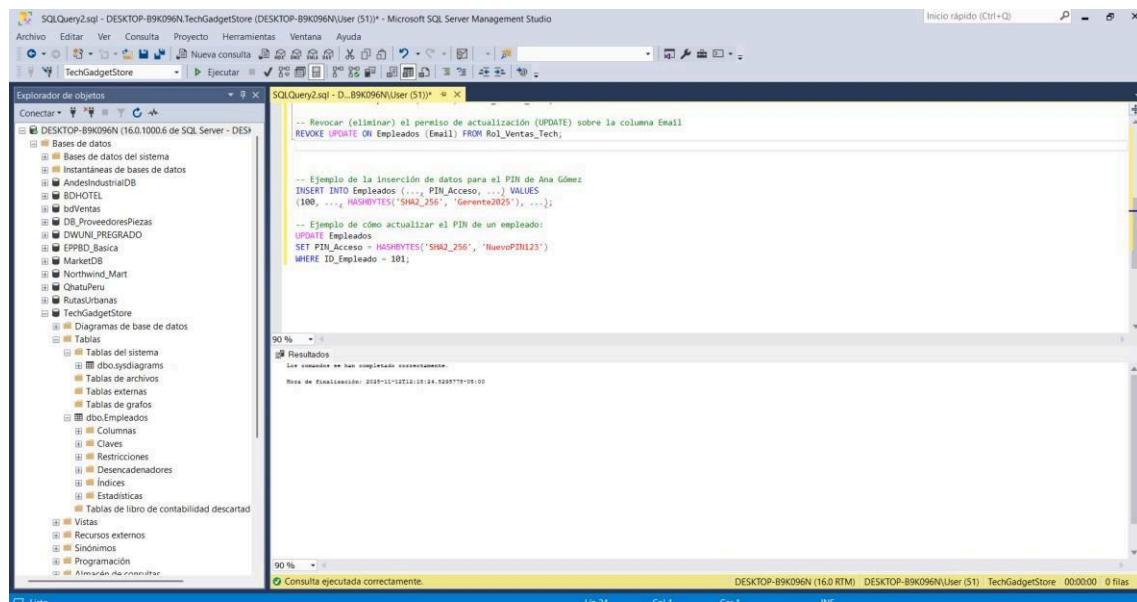
Paso Final: Verificación

Puedes verificar el estado de cifrado con la siguiente consulta:

SQL

```
SELECT
    name,
    is_encrypted
FROM sys.databases
WHERE name = 'TechGadgetStore';
```

- Resultado Esperado:** La columna `is_encrypted` mostrará 1 (Verdadero), indicando que el cifrado TDE está activo.



6. Auditoría y monitoreo de eventos

- Enunciado:** Mantener un registro completo de los eventos de seguridad y acceso para detectar actividades maliciosas.
- Explicación:** La columna `ultimoLogin` es un monitoreo básico. Para una auditoría real, se implementa el **SQL Server Audit**. Configuramos un grupo de auditoría específico (`FAILED_LOGIN_GROUP`) para registrar cada intento de inicio de sesión fallido. Esta herramienta es crucial para identificar **ataques de fuerza bruta** o cuentas comprometidas.
- Código Ejemplo:**

SQL

```
-- Configuración para registrar todos los intentos de login
fallidos
CREATE SERVER AUDIT SPECIFICATION ServerSpec_TechGadget
FOR SERVER AUDIT Audit_TechGadget
ADD (FAILED_LOGIN_GROUP);
ALTER SERVER AUDIT SPECIFICATION ServerSpec_TechGadget WITH (STATE
= ON);
```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery2.sql - DESKTOP-B9K096N\user (S1)' displays the T-SQL code provided above. The code creates a login 'Betologin' with password 'P@r@seguro123', a role 'Rol_Ventas_Tech', and a server audit specification 'ServerSpec_TechGadget' that audits failed logins. The execution progress is at 90%, and a green checkmark indicates the command was completed successfully. The system tray at the bottom shows the date and time as 12/11/2012 12:08.

```
-- Creación de un Login de SQL para un Vendedor
CREATE LOGIN Betologin WITH PASSWORD = 'P@r@seguro123', CHECK_POLICY = ON;
CREATE USER Betouser FOR LOGIN Betologin;
-- consulta tres
-- Rol Personalizado para Vendedores
CREATE ROLE Rol_Ventas_Tech;
ALTER ROLE Rol_Ventas_Tech ADD MEMBER Betouser;
-- Consulta una tabla
-- Configuración para registrar todos los intentos de login fallidos
CREATE SERVER AUDIT SPECIFICATION ServerSpec_TechGadget
FOR SERVER AUDIT Audit_TechGadget
ADD (FAILED_LOGIN_GROUP);
ALTER SERVER AUDIT SPECIFICATION ServerSpec_TechGadget WITH (STATE = ON);
```