UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE: INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE: Tecnologías de la Información

SISTEMAS DE BASES DE DATOS DISTRIBUIDOS

DOCENTE:

JOSE RUBEN CAIZA CAIZABUANO

AGOSTO 2025 - ENERO 2026

Tabla de contenido

INFORME DE TALLER DE TRANSACCIONES

I. PORTADA

Tema:	Taller Transacciones
Unidad de Organización Curricular:	PROFESIONAL
Nivel y Paralelo:	Quinto “A”
Alumnos participantes:	Aldas Jordan Wellington Ismael Caguasango Bayas Alex Patricio Gómez Llerena Luis Fernando Paredes Barrera Luis Enrique
Asignatura:	Sistemas de Base de Datos Distribuidas
Docente:	Ing. José Caiza, Mg.

II. INFORME DE TALLER DE TRANSACCIONES

2.1 Objetivos

2.1.1 Objetivo General:

Desarrollar y comprobar procesos en bases de datos distribuidas con SQL Server, utilizando transacciones, relaciones entre tablas y control de errores, con el fin de mantener la consistencia y seguridad de los datos.

2.1.2 Objetivos Específicos:

- Configurar transacciones que permitan ejecutar varias operaciones como una sola unidad, de manera que se confirmen en su totalidad o se cancelen si ocurre algún error.
- Asegurar la correcta relación entre tablas mediante la implementación de restricciones con claves primarias y foráneas.
- Aplicar mecanismos de manejo de excepciones con TRY-CATCH para controlar fallos durante la ejecución de operaciones sensibles en la base de datos.

2.2 Modalidad

Presencial

2.3 Tiempo de duración

Presenciales: 3

No presenciales: 0

2.4 Instrucciones

1. Conectarse a SQL Server Management Studio (SSMS).
2. Ejecutar scripts SQL proporcionados en la guía.
3. Documentar resultados y capturas de pantalla.
4. Validar cada paso con el docente antes de continuar

2.5 Listado de equipos, materiales y recursos

Listado de equipos y materiales generales empleados en la guía práctica:

- Computador con Windows/Linux/macOS.
- SQL Server Management Studio (SSMS) instalado.
- Acceso a la base de datos CentroMedicoDB.

TAC (Tecnologías para el Aprendizaje y Conocimiento) empleados en la guía práctica:

☒Plataformas educativas
☒Simuladores y laboratorios virtuales
☐Aplicaciones educativas
☒Recursos audiovisuales
☐Gamificación
☐Inteligencia Artificial
Otros (Especifique): _____

2.6 Actividades desarrolladas

2.6.1 CREACIÓN DE UNA NUEVA INSTANCIA DENTRO DE SQL SERVER

Para iniciar el diseño de la base de datos distribuida simulada, se creó una nueva instancia de SQL Server denominada **Centro_Medico**. Esta instancia es necesaria porque permite trabajar de forma independiente con los esquemas y tablas que representan cada sede.

Ejecución del instalador de SQL Server

Al observar la ilustración 1 presentada a continuación logramos observar que al ejecutar el centro de instalación de SQL Server debemos escoger una carpeta en la cual se realizara la creación de la instancia.
CONSEJO: Escoger la carpeta en la que se encuentren los archivos del programa.

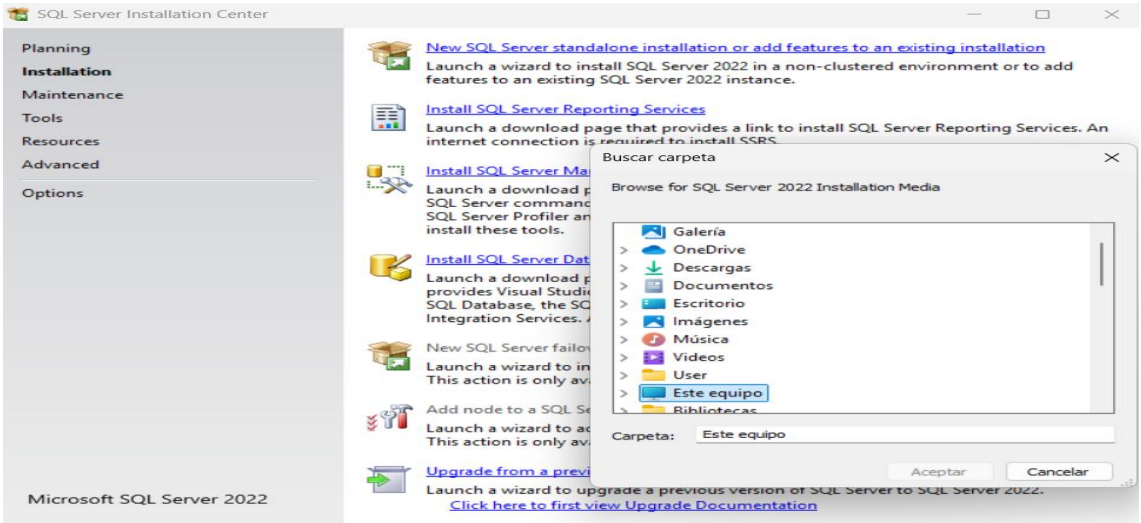


Ilustración 1 Selección de carpeta para la creación de la instancia

Configuración y Creación de la Instancia

Durante la configuración en el apartado de tipo de instalación se escoge la opción Realizar una nueva Instalación de SQL Server, como se puede observar en la ilustración 2.
Esto permitirá realizar la creación de la instancia.

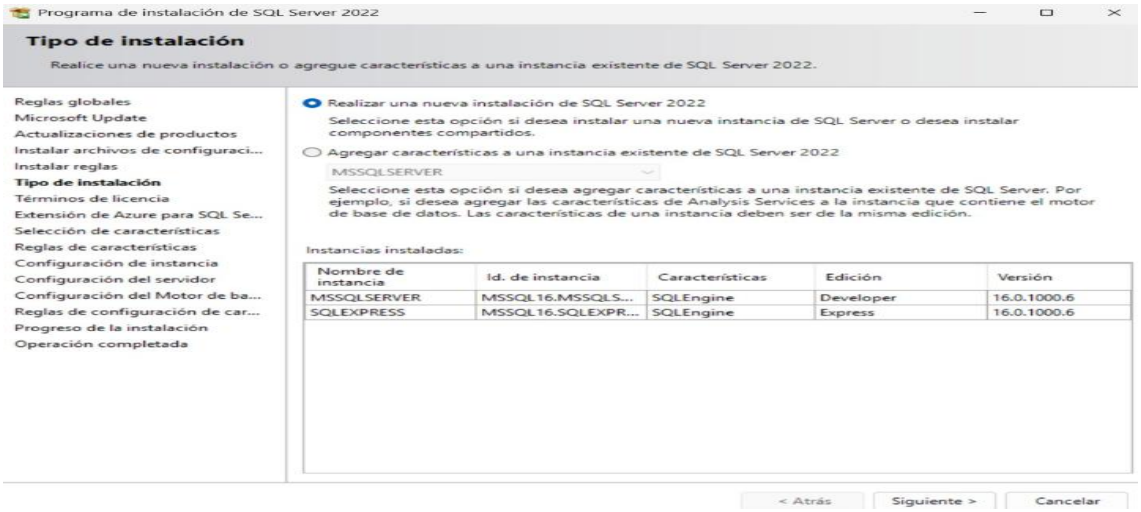


Ilustración 2 Tipo de instalación

En el apartado de **configuración de instancia** debemos seleccionar colocar un nombre propio a la nueva instancia y se colocara el nombre que se desee. En este caso SITIO_A, como se observa en la ilustración 3.

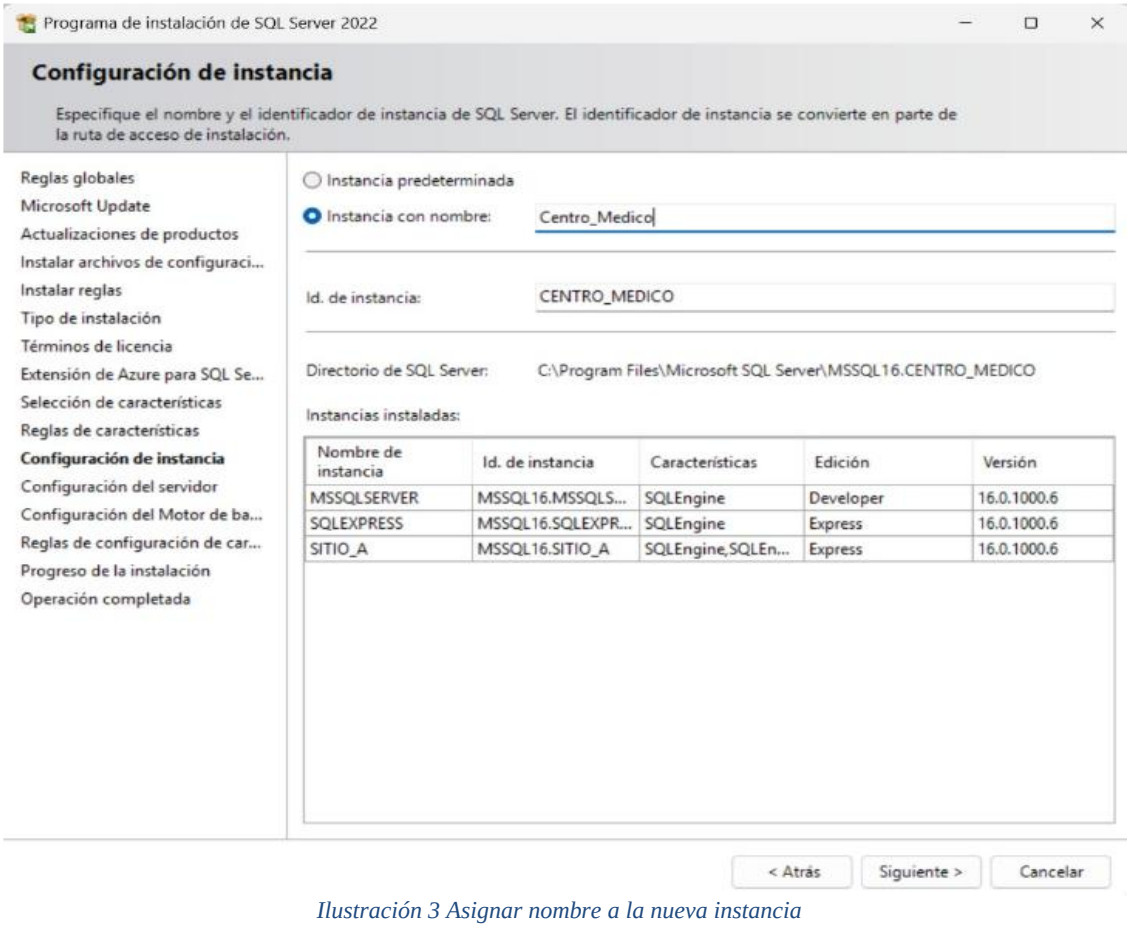


Ilustración 3 Asignar nombre a la nueva instancia

Una vez configurado todos los apartados se iniciará el proceso de instalación como se observa en la ilustración 4.

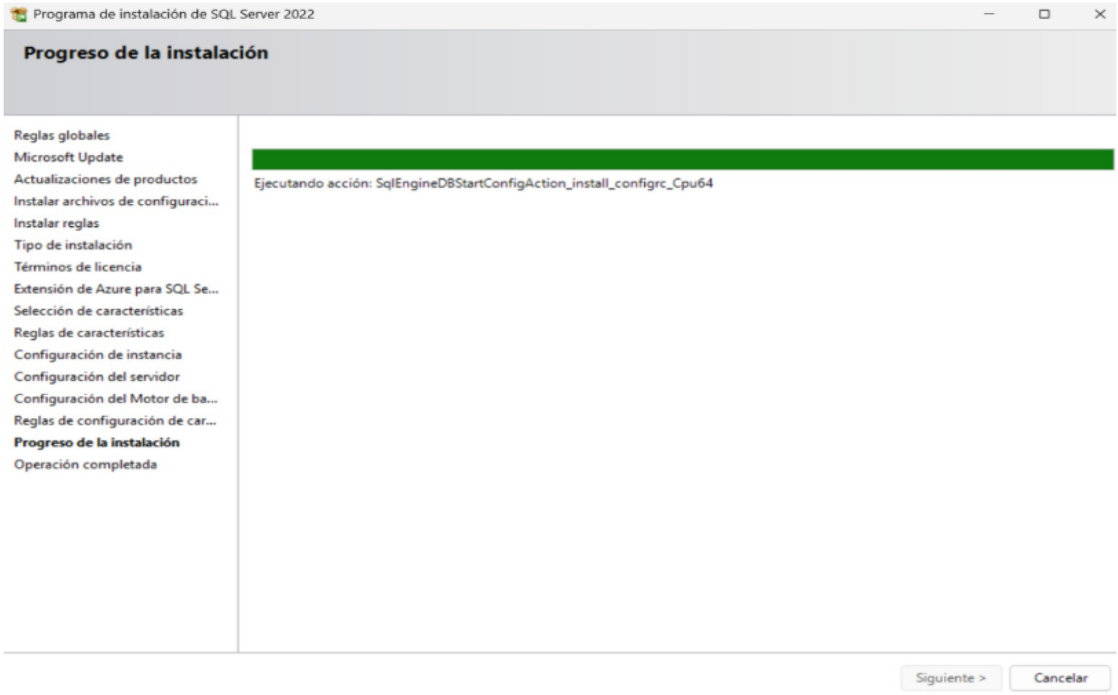


Ilustración 4 Progreso de instalación de la instancia

Como se puede observar en la Ilustración 5, la instancia ha sido creada correctamente y está lista para la posterior creación de la base de datos, esquemas y fragmentos que conformarán la vista global.

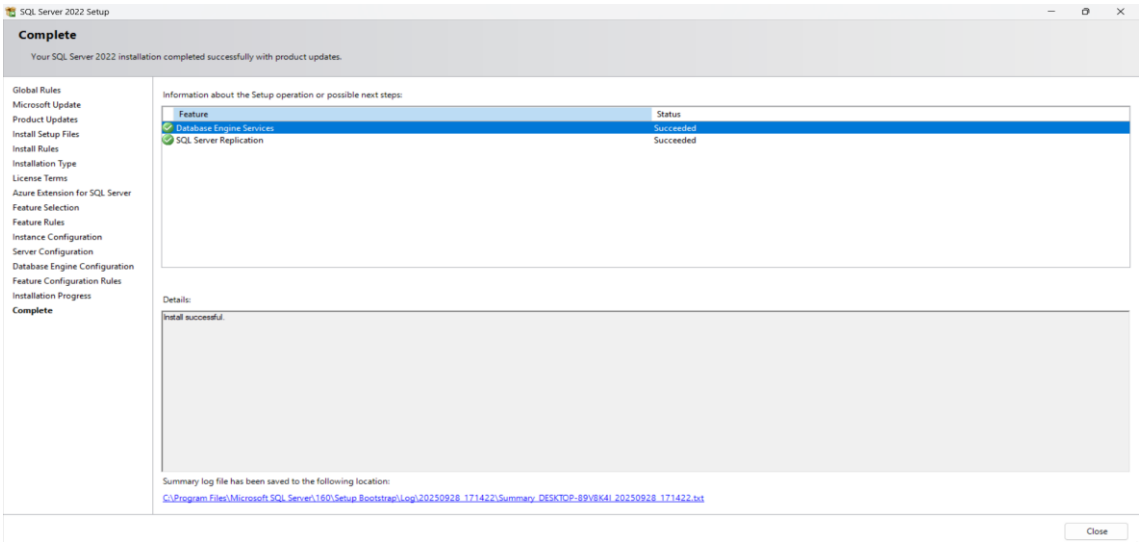


Ilustración 5 Creación Completa de la nueva Instancia

2.6.2 CONECTARSE A SQL SERVER MANAGEMENT STUDIO (SSMS).

¿Como ingresar a la instancia creada?

Para lograr ingresar en la instancia que creamos ejecutamos la aplicación de SQL Server y en la ventana en la cual debemos ingresar el nombre del servidor deberemos ingresar el nombre colocado a la instancia durante su creación, este nombre debe ir antecedido por .\

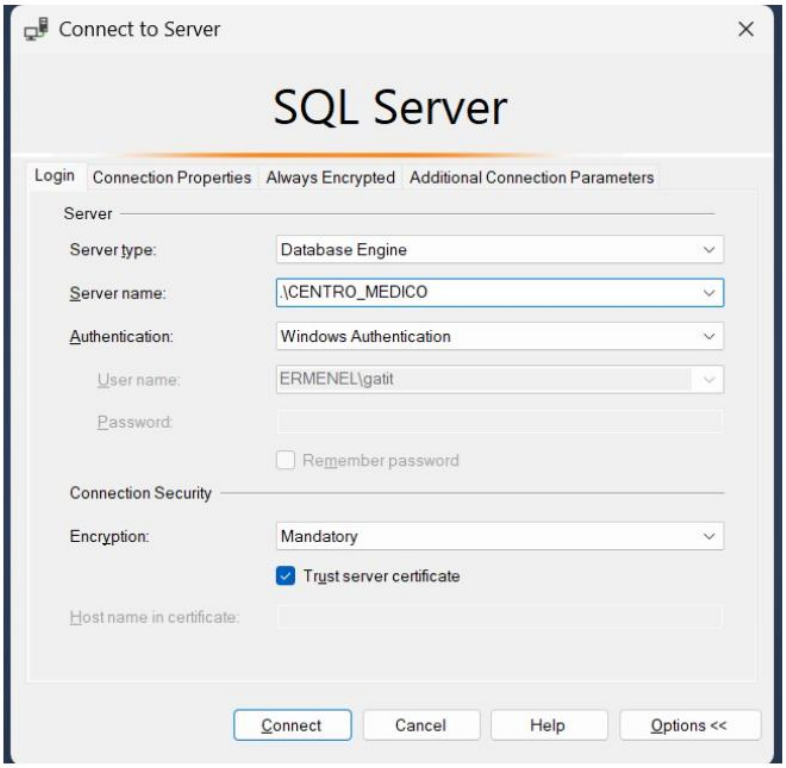


Ilustración 6 Ingreso a la Instancia creada

DISEÑO DE LA BASE DE DATOS

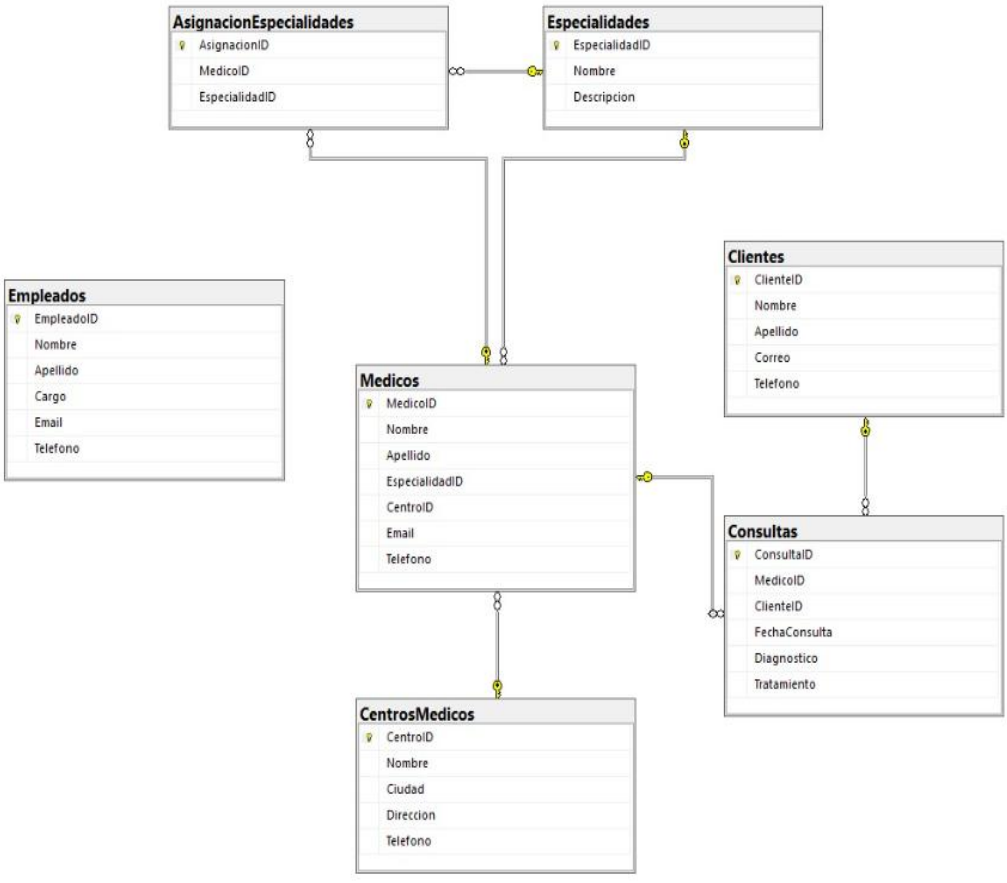


Ilustración 7 Diseño de la Base de Datos a Crear

2.6.3 IMPLEMENTACIÓN TÉCNICA (BD, tablas, datos)

Una vez dentro de la instancia procederemos a la creación de nuestra base de datos.

Creación de la Base de Datos dentro de la Instancia:

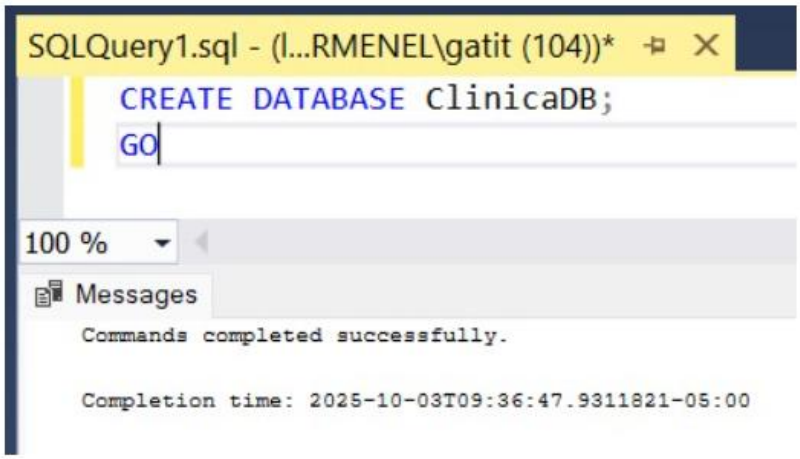


Ilustración 8 Creación de la Base de Datos dentro de SQL SERVER

Verificación de la correcta creación de la base de datos

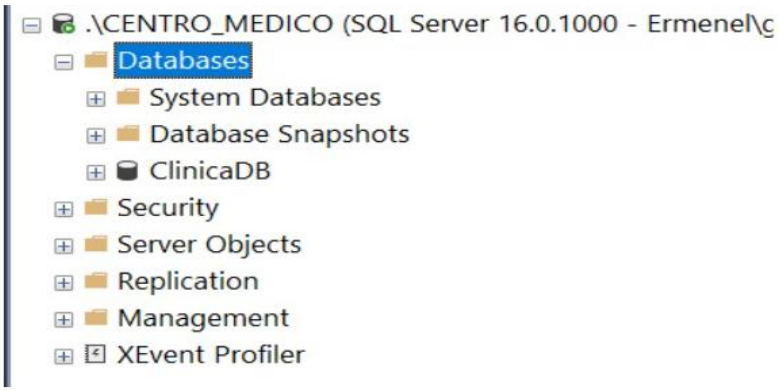


Ilustración 9 Verificar la base de datos creada

Una vez creada la base de datos procedemos a la creación de todas las tablas necesarias:

Creación de Tablas:

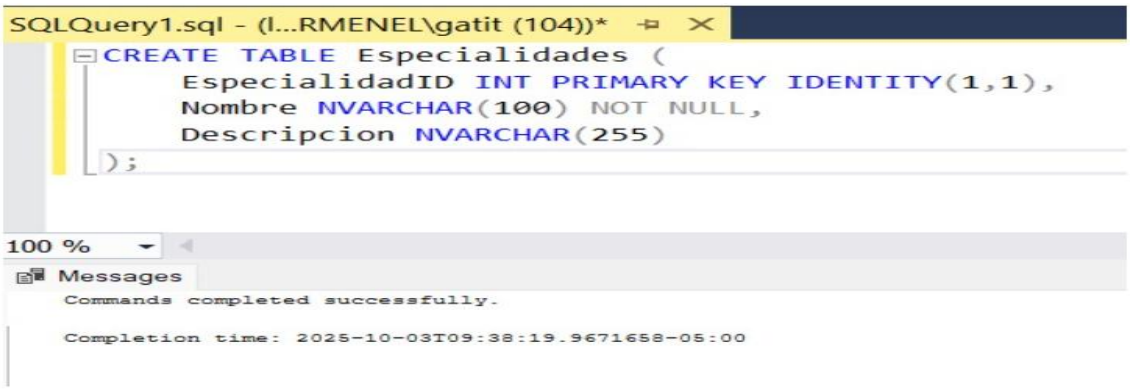


Ilustración 10 Creación de tabla Especialidades

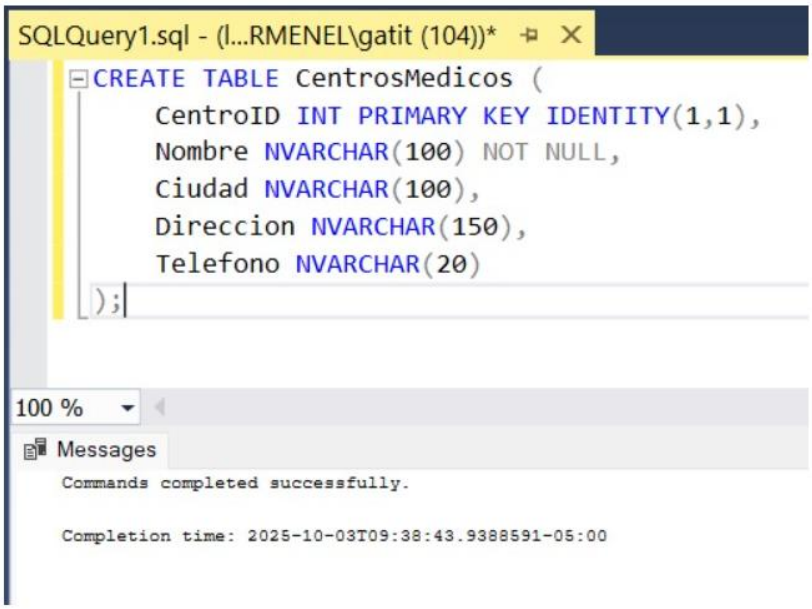


Ilustración 11 Creación de la tabla CentrosMedicos

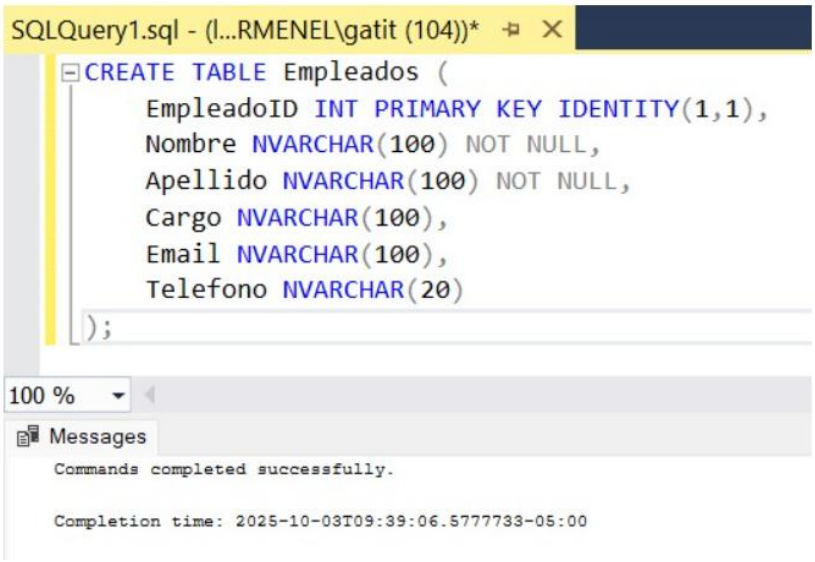


Ilustración 12 Creación de la tabla Empleados

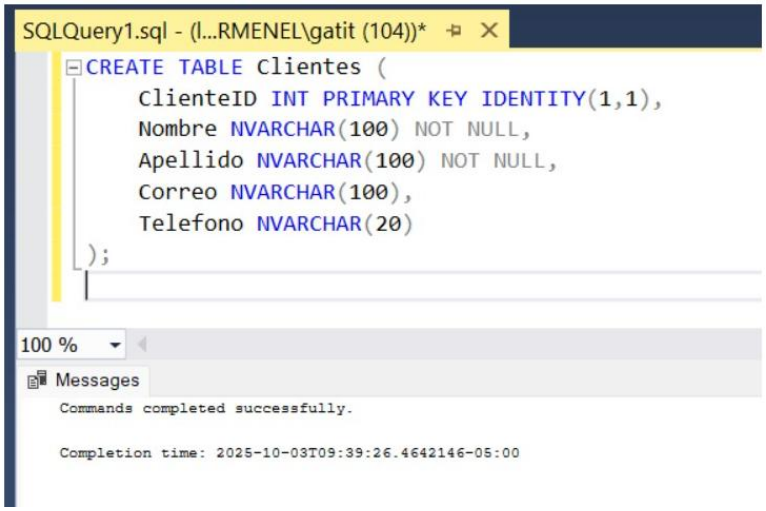


Ilustración 13 Creación de la tabla Clientes

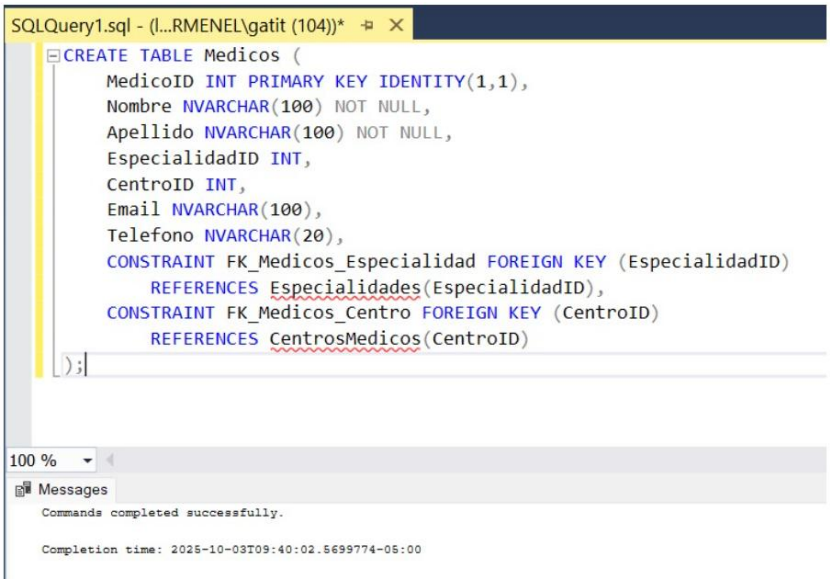


Ilustración 14 Creación de la tabla Medicos

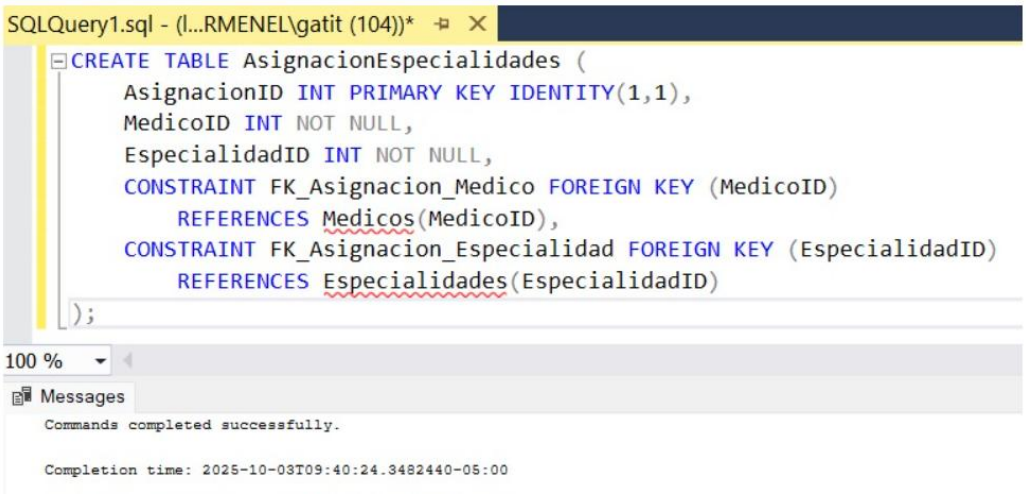


Ilustración 15 Creación de la tabla AsignacionEspecialidades

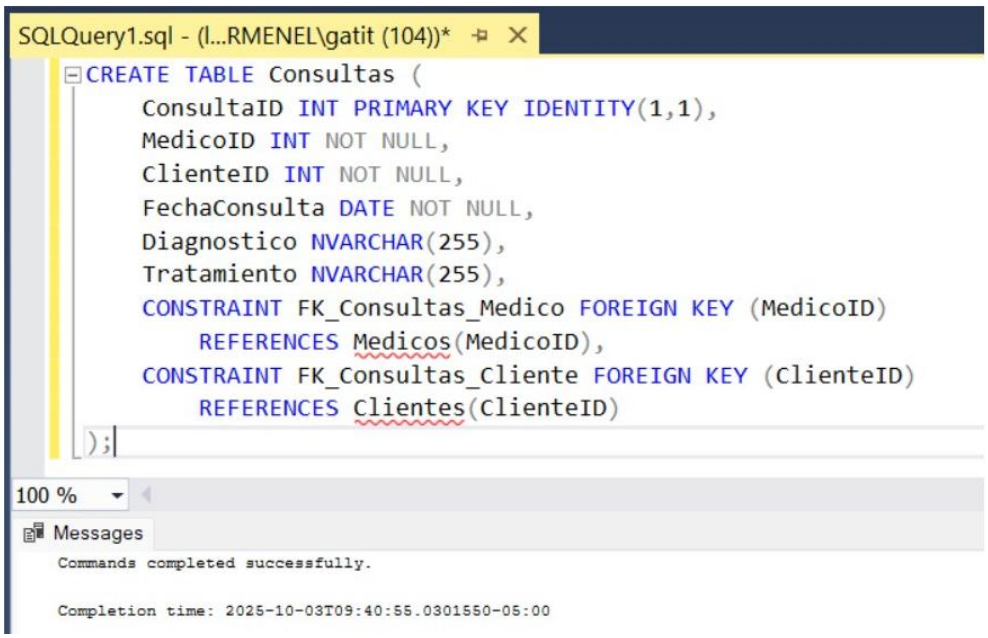


Ilustración 16 Creación de la tabla Consultas

Una vez creada las tablas procedimos a la verificación de su correcta creación:

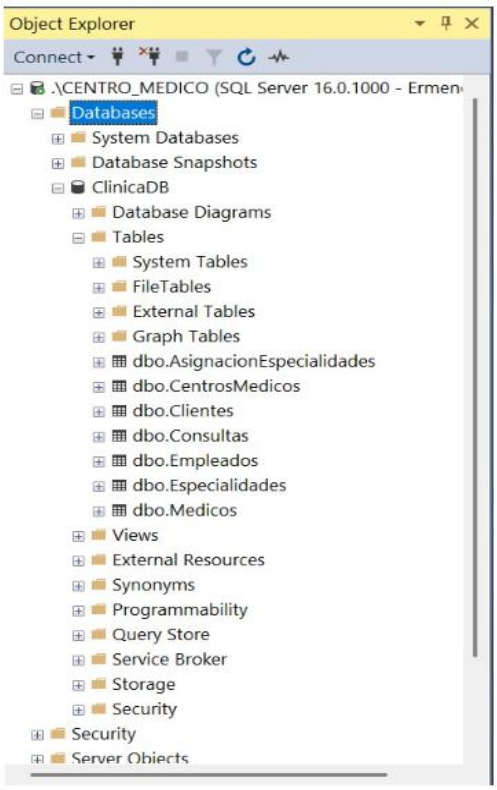


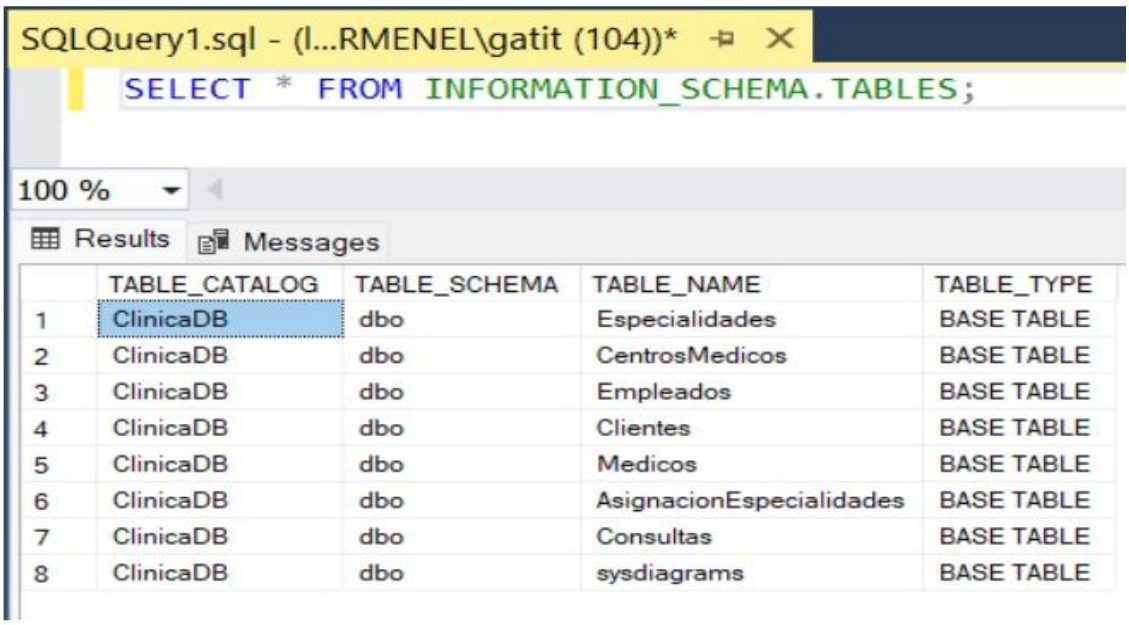
Ilustración 17 Verificación de tablas creadas

2.6.4 EJECUTAR SCRIPTS SQL PROPORCIONADOS EN LA GUÍA.

VERIFICAR CONEXIÓN Y ESTRUCTURA DE LA BASE DE DATOS

Verificamos las tablas existentes ejecutando el comando:

SELECT * FROM INFORMATION_SCHEMA.TABLES;



	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
1	ClinicaDB	dbo	Especialidades	BASE TABLE
2	ClinicaDB	dbo	CentrosMedicos	BASE TABLE
3	ClinicaDB	dbo	Empleados	BASE TABLE
4	ClinicaDB	dbo	Clientes	BASE TABLE
5	ClinicaDB	dbo	Medicos	BASE TABLE
6	ClinicaDB	dbo	AsignacionEspecialidades	BASE TABLE
7	ClinicaDB	dbo	Consultas	BASE TABLE
8	ClinicaDB	dbo	sysdiagrams	BASE TABLE

Ilustración 18 Verificar conexión y estructura de la base de datos

2.6.5 COMPROBAR LA INTEGRIDAD REFERENCIAL CON DATOS EXISTENTES

Ejemplo1: Ingresar datos existentes en las diferentes tablas:

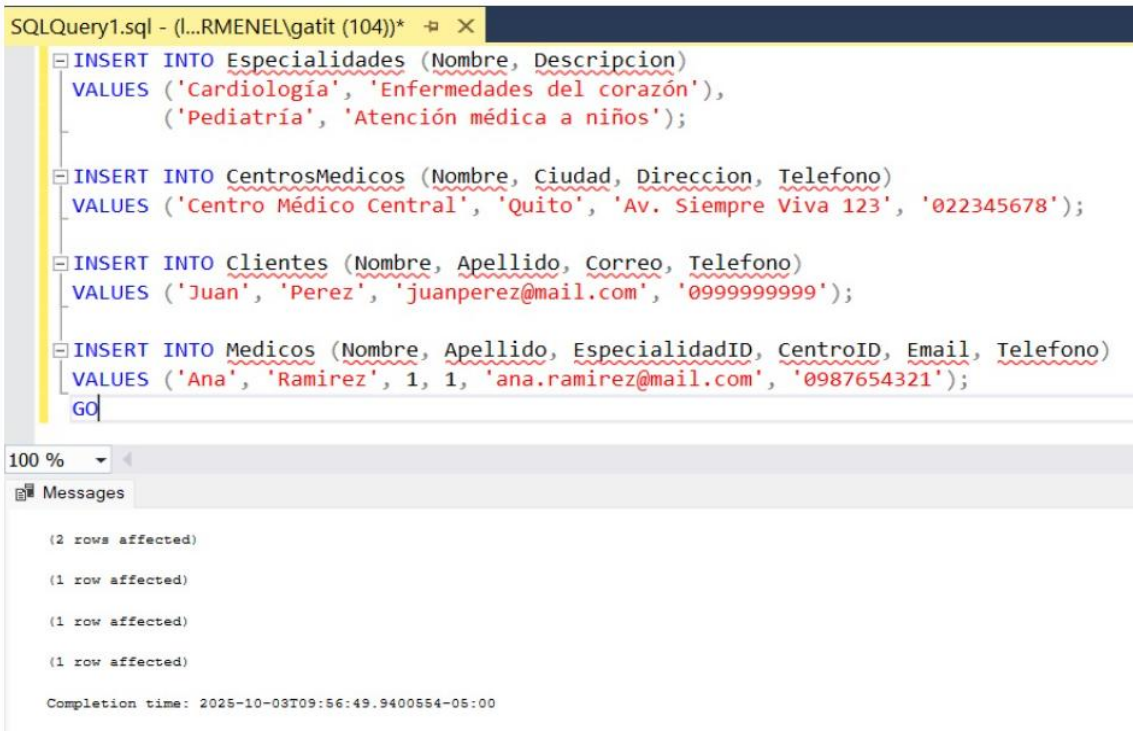


Ilustración 19 Ingresar datos existentes en diferentes tablas

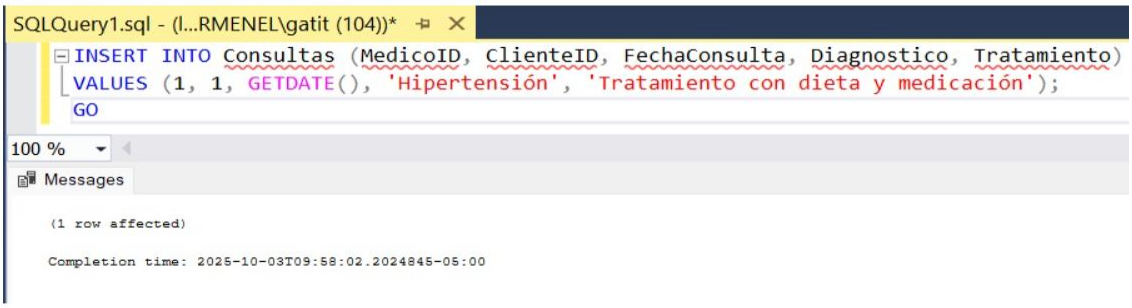


Ilustración 20 Ingresar datos existentes en diferentes tablas

Como podemos observar en las ilustraciones 19 y 20 todas las ejecuciones fueron correctas por lo que las relaciones dentro de nuestra base se encuentran correctamente.

Ejemplo 2: Intentar eliminar una especialidad en uso (Pediatría, ID=1);

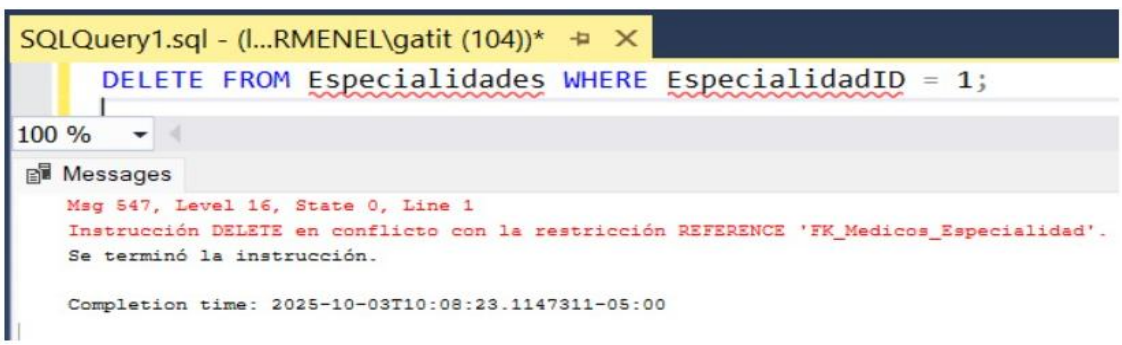


Ilustración 21 Intento de eliminación de especialidad en uso

2.6.6 TRANSACCIONES CON ATOMICIDAD (COMMIT Y ROLLBACK)

Transacción exitosa: Crear un centro médico y asignar un médico:

```
SQLQuery1.sql - (l...RMENEL\gatit (104))*
BEGIN TRANSACTION;

BEGIN TRY
    -- Insertamos un cliente válido
    INSERT INTO Clientes (Nombre, Apellido, Correo, Telefono)
    VALUES ('Maria', 'Gonzalez', 'maria.g@mail.com', '0911111111');

    -- Insertamos un médico válido (referencia a EspecialidadID = 1 y CentroID = 1)
    INSERT INTO Medicos (Nombre, Apellido, EspecialidadID, CentroID, Email, Telefono)
    VALUES ('Luis', 'Martinez', 1, 1, 'luis.martinez@mail.com', '0922222222');

    -- Si todo va bien, confirmamos los cambios
    COMMIT TRANSACTION;
    PRINT 'Transacción exitosa → cambios confirmados (COMMIT).';
END TRY
BEGIN CATCH
    -- Si ocurre error, revertimos
    ROLLBACK TRANSACTION;
    PRINT 'Error en transacción → cambios revertidos (ROLLBACK).';
    PRINT ERROR_MESSAGE();
END CATCH;

100 %
Messages

(1 row affected)

(1 row affected)
Transacción exitosa ? cambios confirmados (COMMIT).

Completion time: 2025-10-03T10:10:59.9376276-05:00
```

Ilustración 22 Transacción Exitosa

Verificación de la correcta ejecución de la transacción:

```
SQLQuery1.sql - (l...RMENEL\gatit (104))*
SELECT * FROM Clientes WHERE Nombre = 'Maria';
SELECT * FROM Medicos WHERE Nombre = 'Luis';

100 %
Results  Messages
```

	ClienteID	Nombre	Apellido	Correo	Telefono
1	2	Maria	Gonzalez	maria.g@mail.com	0911111111

	MedicoID	Nombre	Apellido	EspecialidadID	CentroID	Email	Telefono
1	5	Luis	Martinez	1	1	luis.martinez@mail.com	0922222222

Ilustración 23 Verificación de Transacción Exitosa

Transacción fallida: Intentar asignar un médico a un centro inexistente:

```
SQLQuery1.sql - (l...RMENEL\gatit (104))*
BEGIN TRANSACTION;

BEGIN TRY
    -- Insertamos un cliente válido
    INSERT INTO Clientes (Nombre, Apellido, Correo, Telefono)
    VALUES ('Pedro', 'Suarez', 'pedro.s@mail.com', '0933333333');

    -- Intentamos insertar un médico con EspecialidadID inexistente (99)
    INSERT INTO Medicos (Nombre, Apellido, EspecialidadID, CentroID, Email, Telefono)
    VALUES ('ErrorTest', 'Medico', 99, 1, 'errortest@mail.com', '0944444444');

    -- No llegará aquí porque fallará la FK
    COMMIT TRANSACTION;
    PRINT 'Transacción exitosa (esto NO debería mostrarse en caso de error).';
END TRY
BEGIN CATCH
    -- Si ocurre error, se revierten todos los cambios
    ROLLBACK TRANSACTION;
    PRINT 'Transacción fallida → cambios revertidos (ROLLBACK).';
    PRINT ERROR_MESSAGE();
END CATCH;

100 %
Messages

(1 row affected)

(0 rows affected)
Transacción fallida ? cambios revertidos (ROLLBACK).
Instrucción INSERT en conflicto con la restricción FOREIGN KEY 'FK_Medicos_Especialidad'. El conflicto ha aparecido en la base de datos

Completion time: 2025-10-03T10:12:36.8902347-05:00
```

Ilustración 24 Transacción fallida

Verificación de la Transacción Fallida:

```
SQLQuery1.sql - (l...RMENEL\gatit (104))*
SELECT * FROM Clientes WHERE Nombre = 'Pedro';

100 %
Results  Messages
ClienteID  Nombre  Apellido  Correo  Telefono
```

Ilustración 25 Verificación de Transacción Fallida

2.6.7 PRUEBAS CONCURRENTES

Sesión 1 (Actualización):

Mantener la transacción abierta (sin COMMIT)

```
SQLQuery1.sql - (l...RMENEL\gatit (104))*
BEGIN TRANSACTION;

UPDATE Clientes
SET Telefono = '0999999999'
WHERE ClienteID = 1;

100 %
Messages

(1 row affected)

Completion time: 2025-10-03T10:16:27.5251506-05:00
```

Ilustración 26 Pruebas Concurrentes (ACTUALIZACIÓN)

Sesión 2 (Lectura):

Bloqueado hasta que Sesión 1 finalice

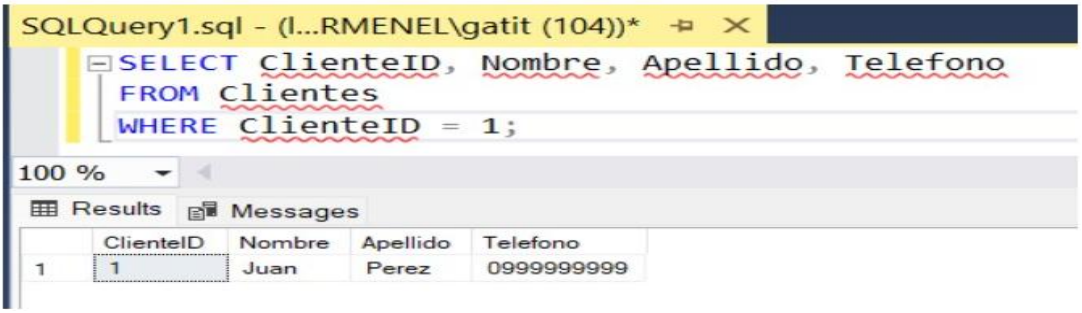


Ilustración 27 Pruebas Concurrentes (LECTURA)

2.6.8 MANEJO DE ERRORES CON TRY…CATCH

Ejemplo 1: Insertar un cliente con un ClienteID existente.

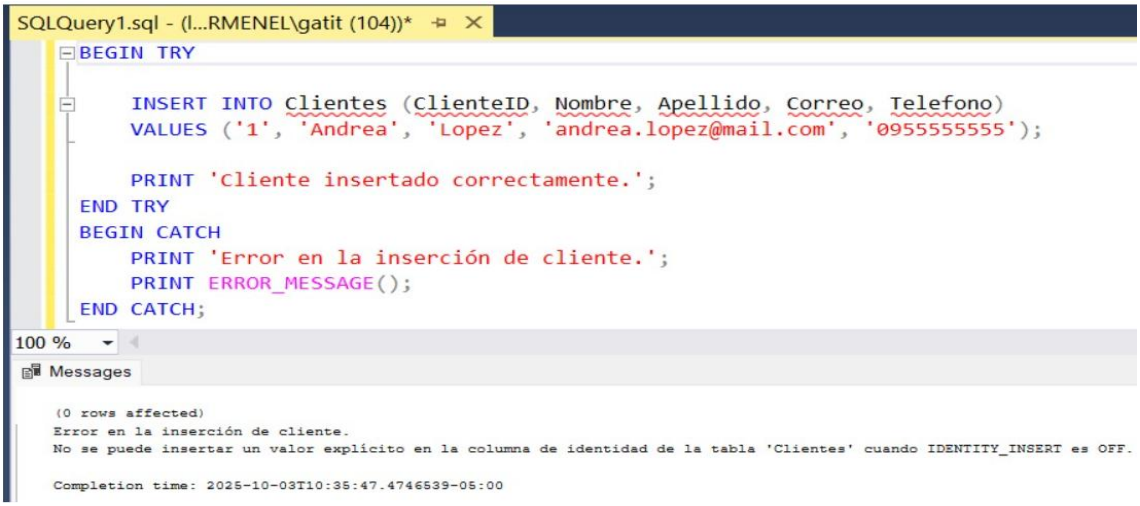


Ilustración 28 Manejo de errores - ID existente

Ejemplo 2: Error forzado por violación FK

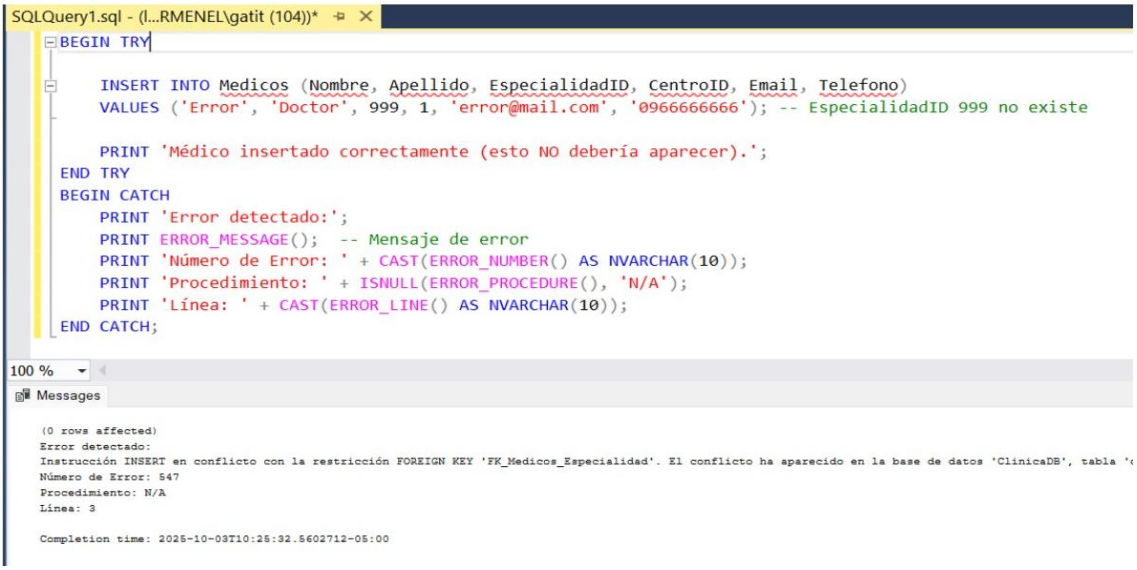


Ilustración 29 Manejo de errores - Violación FK

2.6.9 VERIFICACIÓN FINAL

Consultar datos nuevos:

Ejemplo 1: Verificar los clientes nuevos

```
SELECT ClienteID, Nombre, Apellido, Correo, Telefono
FROM Clientes
ORDER BY ClienteID DESC;
```

100 %

Results | Messages

	ClienteID	Nombre	Apellido	Correo	Telefono
1	5	Andrea	Lopez	andrea.lopez@mail.com	0955555555
2	4	Andrea	Lopez	andrea.lopez@mail.com	0955555555
3	2	Maria	Gonzalez	maria.g@mail.com	0911111111
4	1	Juan	Perez	juanperez@mail.com	0999999999

Ilustración 30 Verificación Final - Clientes

Ejemplo 2: Consultar Médicos creados -- Incluye la especialidad y el centro médico

```
SELECT m.MedicoID, m.Nombre, m.Apellido, e.Nombre AS Especialidad,
       c.Nombre AS CentroMedico, m.Email, m.Telefono
FROM Medicos m
LEFT JOIN Especialidades e ON m.EspecialidadID = e.EspecialidadID
LEFT JOIN CentrosMedicos c ON m.CentroID = c.CentroID
ORDER BY m.MedicoID DESC;
```

100 %

Results | Messages

	MedicoID	Nombre	Apellido	Especialidad	CentroMedico	Email	Telefono
1	5	Luis	Martinez	Cardiología	Centro Médico Central	luis.martinez@mail.com	0922222222
2	3	Carlos	Lopez	Cardiología	Centro Médico Central	carlos.lopez@mail.com	0977777777
3	1	Ana	Ramirez	Cardiología	Centro Médico Central	ana.ramirez@mail.com	0987654321

Ilustración 31 Verificación Final - Médicos

Ejemplo 3: Consultar Consultas registradas -- Con detalle de médico y cliente

```
SELECT con.ConsultaID,
       con.FechaConsulta,
       m.Nombre + ' ' + m.Apellido AS Medico,
       cl.Nombre + ' ' + cl.Apellido AS Cliente,
       con.Diagnostico,
       con.Tratamiento
FROM Consultas con
INNER JOIN Medicos m ON con.MedicoID = m.MedicoID
INNER JOIN Clientes cl ON con.ClienteID = cl.ClienteID
ORDER BY con.ConsultaID DESC;
```

100 %

Results | Messages

	ConsultaID	FechaConsulta	Medico	Cliente	Diagnostico	Tratamiento
1	1	2025-10-03	Ana Ramirez	Juan Perez	Hipertensión	Tratamiento con dieta y medicación

Ilustración 32 Verificación Final - Consultas

2.7 Resultados obtenidos

La realización de la practica permitió la comprensión de la correcta implementación de las transacciones en SQL Server, comprobando la atomicidad con operaciones exitosas y fallidas mediante COMMIT y ROLLBACK. Se verificó la integridad referencial a través de claves primarias y foráneas, evitando eliminaciones o inserciones inválidas, y se comprobó el bloqueo en escenarios de concurrencia con múltiples sesiones. Además, el manejo de errores con TRY-CATCH permitió controlar excepciones sin afectar la consistencia de los datos. En conjunto, las pruebas demostraron que el diseño de la base de datos distribuida mantiene la coherencia, confiabilidad y transparencia en el acceso a la información.

2.8 Habilidades blandas empleadas en la práctica

☐ Liderazgo
☐ Trabajo en equipo
☐ Comunicación asertiva
☐ La empatía
☒ Pensamiento crítico
☐ Flexibilidad
☒ La resolución de conflictos
☒ Adaptabilidad
☒ Responsabilidad

Durante el desarrollo de la práctica se aplicaron distintas habilidades blandas que facilitaron el trabajo en equipo y la correcta implementación de la base de datos.

El pensamiento crítico fue clave para analizar las transacciones, detectar posibles inconsistencias y aplicar medidas que aseguren la coherencia de los datos en el sistema.

La resolución de conflictos fue necesaria al enfrentar errores en la configuración de las tablas y relaciones, permitiendo encontrar soluciones rápidas sin detener el avance.

La adaptabilidad se puso en práctica al ajustarse a imprevistos técnicos y a los cambios requeridos por el docente durante la validación de los ejercicios.

La responsabilidad se reflejó en el cumplimiento de cada paso de la guía y en la verificación de resultados para garantizar la calidad del trabajo.

2.9 Conclusiones

- El uso de transacciones permitió comprobar que múltiples operaciones pueden ejecutarse de manera confiable como una sola unidad, asegurando que en caso de fallo los cambios se reviertan sin afectar la base de datos.

- Las pruebas de integridad referencial confirmaron que las claves primarias y foráneas cumplen su función al evitar eliminaciones o inserciones que rompan la coherencia entre tablas relacionadas.

- La implementación de TRY-CATCH evidenció su utilidad para manejar errores en tiempo de ejecución, ya que permitió capturar excepciones y responder adecuadamente sin comprometer la información almacenada.

2.10 Recomendaciones

- Mantener siempre el uso de transacciones en operaciones críticas, ya que permiten garantizar la atomicidad y evitar inconsistencias en la base de datos.

- Documentar claramente las relaciones entre tablas y las reglas de integridad referencial, lo que facilitará el mantenimiento y la detección de errores en futuros desarrollos.

2.11 Anexos

Resultado Final de la Base de Datos Implementada dentro de la Instancia Creada.