



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO



Proyecto Final ABD

Alumno: Alonso Palafox Miguel Ángel

Núm. De control: 22070280

Carrera: Ingeniería en Sistemas Computacionales

Docente: Fernando Manzanares González

Materia: Administración de Base de Datos

HORA: 16:00 – 17:00 HRS

PERIODO: ENERO – JUNIO 2025

Introducción

Este documento presenta la guía paso a paso para la administración básica de una base de datos en SQL Server destinada al registro de expedientes clínicos digitales, conforme a los lineamientos establecidos en la Norma Oficial Mexicana NOM-0024-SSA3-2012.

El objetivo principal es diseñar y crear un esquema de base de datos que permita almacenar y gestionar la información clínica de manera estructurada y segura, así como implementar mecanismos de control de acceso mediante la creación de usuarios, roles y permisos específicos.

Además, se abordará la automatización de tareas administrativas como la creación de respaldos automáticos y la implementación de planes de mantenimiento que garanticen la integridad y disponibilidad de la base de datos.

Finalmente, se realizarán modificaciones estructurales y pruebas de restauración para asegurar la correcta gestión de la base de datos a lo largo de su ciclo de vida.

Este documento incluye la descripción detallada de cada paso, los comandos SQL empleados, así como las evidencias visuales de la ejecución y los resultados obtenidos.

1. Creación de la Base de Datos

Para el desarrollo del sistema de registro de expedientes clínicos digitales, se diseñó y creó la base de datos denominada ExpedienteClinicoDB. Esta base fue estructurada conforme a los lineamientos establecidos en la Norma Oficial Mexicana NOM-024-SSA3-2012, poniendo especial énfasis en los capítulos relacionados con las generalidades del expediente clínico y los requisitos específicos para consultas generales y de especialidad. El esquema garantiza el manejo adecuado de la información médica, cumpliendo con las disposiciones de confidencialidad, integridad y disponibilidad requeridas.

Query para crear la base de datos ExpedienteClinicoDB:

```
-- 1. Crear la base de datos para el expediente clínico digital
CREATE DATABASE ExpedienteClinicoDB;
GO

-- 2. Usar la base de datos creada
USE ExpedienteClinicoDB;
GO
```

-- 3. Crear tabla establecimiento
-- 5.4.1 Identificación del establecimiento

```
CREATE TABLE establecimiento (  
    establecimiento_id INT PRIMARY KEY IDENTITY(1,1),  
    nombre NVARCHAR(100) NOT NULL,  
    tipo NVARCHAR(50) NOT NULL,  
    domicilio NVARCHAR(255) NOT NULL,  
    institucion_perteneciente NVARCHAR(100),  
    razon_social NVARCHAR(100),  
    activo BIT DEFAULT 1  
);  
GO
```

-- 4. Crear tabla paciente
-- 6.2 Datos generales del paciente

```
CREATE TABLE paciente (  
    paciente_id INT PRIMARY KEY IDENTITY(1,1),  
    nombre NVARCHAR(50) NOT NULL,  
    apellido_paterno NVARCHAR(50) NOT NULL,  
    apellido_materno NVARCHAR(50),  
    fecha_nacimiento DATE NOT NULL,  
    sexo CHAR(1) NOT NULL CHECK (sexo IN ('H', 'M')),  
    domicilio NVARCHAR(255),  
    telefono NVARCHAR(10),  
    email NVARCHAR(100),  
    fecha_registro DATETIME DEFAULT GETDATE(),  
    responsable_registro NVARCHAR(100)  
);  
GO
```

-- 5. Crear tabla expediente_clinico
-- 6.3 Control y registro del expediente clínico

```
CREATE TABLE expediente_clinico (  
    expediente_id INT PRIMARY KEY IDENTITY(1,1),  
    paciente_id INT NOT NULL,  
    establecimiento_id INT NOT NULL,  
    numero_expediente NVARCHAR(20) NOT NULL UNIQUE,  
    fecha_creacion DATETIME NOT NULL DEFAULT GETDATE(),  
    estatus NVARCHAR(20) NOT NULL CHECK (estatus IN ('Activo', 'Inactivo', 'Archivado')),  
    fecha_ultima_actualizacion DATETIME,  
    usuario_ultima_modificacion NVARCHAR(100),  
  
    FOREIGN KEY (paciente_id) REFERENCES paciente(paciente_id),  
    FOREIGN KEY (establecimiento_id) REFERENCES establecimiento(establecimiento_id)  
);  
GO
```

```

-- 6. Crear tabla personal_salud
-- 5.5 Identificación y acreditación del personal de salud
CREATE TABLE personal_salud (
    personal_id INT PRIMARY KEY IDENTITY(1,1),
    nombre NVARCHAR(50) NOT NULL,
    apellido_paterno NVARCHAR(50) NOT NULL,
    apellido_materno NVARCHAR(50),
    cedula_profesional NVARCHAR(20) NOT NULL UNIQUE,
    especialidad NVARCHAR(100),
    tipo NVARCHAR(50) CHECK (tipo IN ('Médico', 'Enfermero', 'Psicólogo', 'Nutriólogo',
'Odontólogo')),
    activo BIT DEFAULT 1
);
GO

```

```

-- 7. Crear tabla historia_clinica
-- 6.5 Historia clínica y documentación clínica
CREATE TABLE historia_clinica (
    historia_id INT PRIMARY KEY IDENTITY(1,1),
    expediente_id INT NOT NULL,
    personal_id INT NOT NULL,
    fecha_elaboracion DATETIME NOT NULL DEFAULT GETDATE(),
    grupo_etnico NVARCHAR(50),
    antecedentes_familiares NVARCHAR(500),
    antecedentes_personales NVARCHAR(500),
    padecimiento_actual NVARCHAR(500) NOT NULL,
    signos_vitales NVARCHAR(200),
    peso DECIMAL(5,2),
    talla DECIMAL(5,2),
    exploracion_fisica NVARCHAR(1000),
    diagnostico_presuntivo NVARCHAR(500),
    codigo_cie10 NVARCHAR(10),
    pronostico NVARCHAR(200),
    firma_digital NVARCHAR(MAX),

    FOREIGN KEY (expediente_id) REFERENCES expediente_clinico(expediente_id),
    FOREIGN KEY (personal_id) REFERENCES personal_salud(personal_id)
);
GO

```

-- 8. Crear tabla nota_evolucion

-- 6.6 Notas de evolución clínica

```
CREATE TABLE nota_evolucion (  
    nota_id INT PRIMARY KEY IDENTITY(1,1),  
    expediente_id INT NOT NULL,  
    personal_id INT NOT NULL,  
    fecha_hora DATETIME NOT NULL DEFAULT GETDATE(),  
    evolucion_clinica NVARCHAR(1000) NOT NULL,  
    signos_vitales NVARCHAR(200),  
    resultados_estudios NVARCHAR(500),  
    diagnostico_actual NVARCHAR(500),  
    pronostico NVARCHAR(200),  
    tratamiento NVARCHAR(1000),  
    nombre_completo NVARCHAR(150) NOT NULL,  
    firma_digital NVARCHAR(MAX),  
  
    FOREIGN KEY (expediente_id) REFERENCES expediente_clinico(expediente_id),  
    FOREIGN KEY (personal_id) REFERENCES personal_salud(personal_id)  
);  
GO
```

-- 9. Crear tabla acceso_expediente

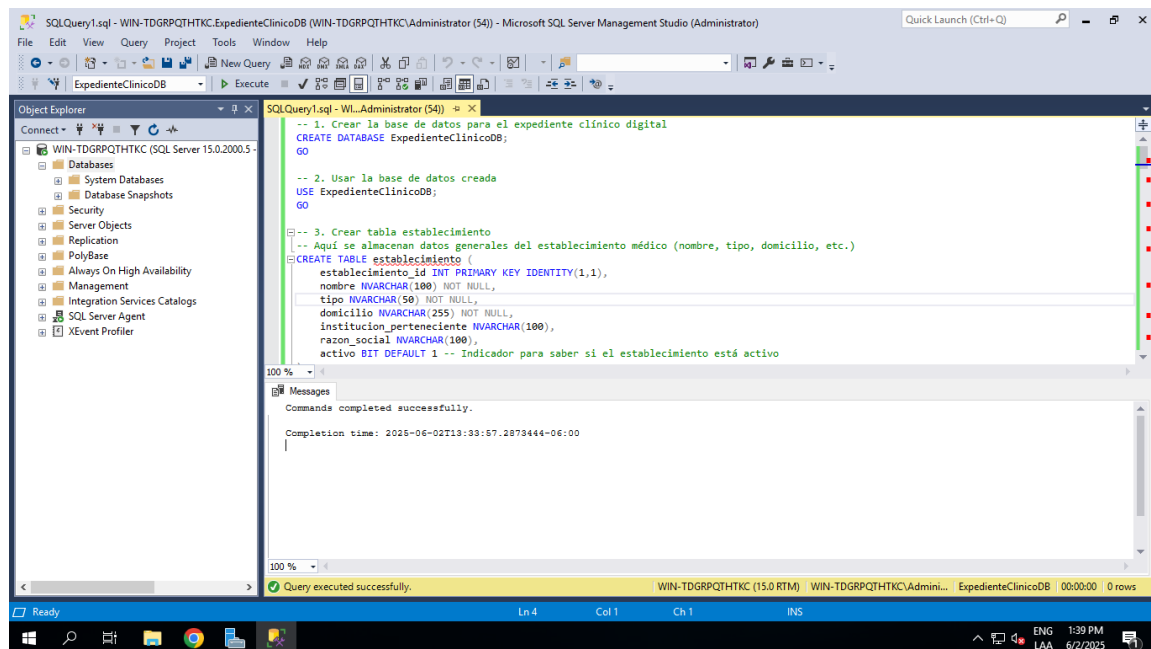
-- 5.7 Confidencialidad y control de acceso

```
CREATE TABLE acceso_expediente (  
    acceso_id INT PRIMARY KEY IDENTITY(1,1),  
    expediente_id INT NOT NULL,  
    personal_id INT NOT NULL,  
    fecha_acceso DATETIME NOT NULL DEFAULT GETDATE(),  
    motivo NVARCHAR(200) NOT NULL,  
    nombre_completo NVARCHAR(150) NOT NULL,  
  
    FOREIGN KEY (expediente_id) REFERENCES expediente_clinico(expediente_id),  
    FOREIGN KEY (personal_id) REFERENCES personal_salud(personal_id)  
);  
GO
```

- 10. Crear tabla autorizacion_acceso
- 5.8 Autorización para acceso a expedientes clínicos

```
CREATE TABLE autorizacion_acceso (  
    autorizacion_id INT PRIMARY KEY IDENTITY(1,1),  
    expediente_id INT NOT NULL,  
    paciente_id INT NOT NULL,  
    autorizado_por NVARCHAR(100) NOT NULL,  
    nombre_autorizador NVARCHAR(150) NOT NULL,  
    fecha_autorizacion DATE NOT NULL,  
    motivo NVARCHAR(200) NOT NULL,  
    vigencia DATE,  
  
    FOREIGN KEY (expediente_id) REFERENCES expediente_clinico(expediente_id),  
    FOREIGN KEY (paciente_id) REFERENCES paciente(paciente_id)  
);  
GO
```

Evidencia

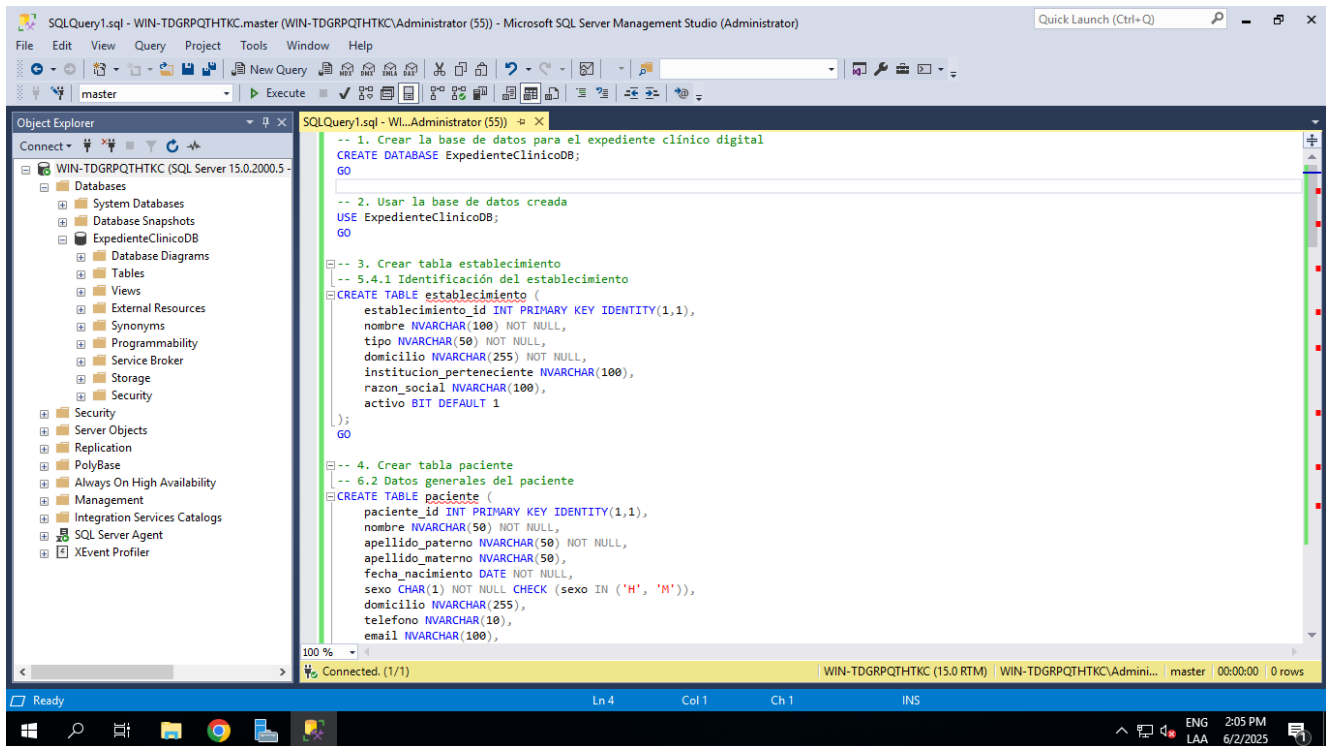


Creada completamente sin ningún problema.

La base de datos ExpedienteClinicoDB fue diseñada y creada exitosamente siguiendo los lineamientos y requerimientos especificados en la NOM-024-SSA3-2012. Todas las tablas fueron implementadas con las relaciones adecuadas, llaves primarias y foráneas, así como las restricciones necesarias para garantizar la integridad de los datos.

Se incluyeron campos específicos para el control y seguimiento de expedientes clínicos digitales, asegurando cumplimiento con las normativas de confidencialidad y seguridad. Además, se establecieron tipos de datos adecuados para cada campo y se aplicaron validaciones mediante CHECK constraints para garantizar la calidad de la información capturada.

La creación se realizó sin errores, y la estructura quedó preparada para la carga y manejo de datos clínicos conforme a las necesidades del sistema.



The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the connection is to 'SQLQuery1.sql - WIN-TDGRPQTHTKC.master (WIN-TDGRPQTHTKC\Administrator (55)) - Microsoft SQL Server Management Studio (Administrator)'. The 'Object Explorer' on the left shows the server structure, including 'Databases' and 'ExpedienteClinicoDB'. The 'Query Editor' on the right contains a SQL script with the following content:

```
-- 1. Crear la base de datos para el expediente clínico digital
CREATE DATABASE ExpedienteClinicoDB;
GO

-- 2. Usar la base de datos creada
USE ExpedienteClinicoDB;
GO

-- 3. Crear tabla establecimiento
-- 5.4.1 Identificación del establecimiento
CREATE TABLE establecimiento (
    establecimiento_id INT PRIMARY KEY IDENTITY(1,1),
    nombre NVARCHAR(100) NOT NULL,
    tipo NVARCHAR(50) NOT NULL,
    domicilio NVARCHAR(255) NOT NULL,
    institucion_perteneiente NVARCHAR(100),
    razon_social NVARCHAR(100),
    activo BIT DEFAULT 1
);
GO

-- 4. Crear tabla paciente
-- 6.2 Datos generales del paciente
CREATE TABLE paciente (
    paciente_id INT PRIMARY KEY IDENTITY(1,1),
    nombre NVARCHAR(50) NOT NULL,
    apellido_paterno NVARCHAR(50) NOT NULL,
    apellido_materno NVARCHAR(50),
    fecha_nacimiento DATE NOT NULL,
    sexo CHAR(1) NOT NULL CHECK (sexo IN ('H', 'M')),
    domicilio NVARCHAR(255),
    telefono NVARCHAR(10),
    email NVARCHAR(100),
);
```

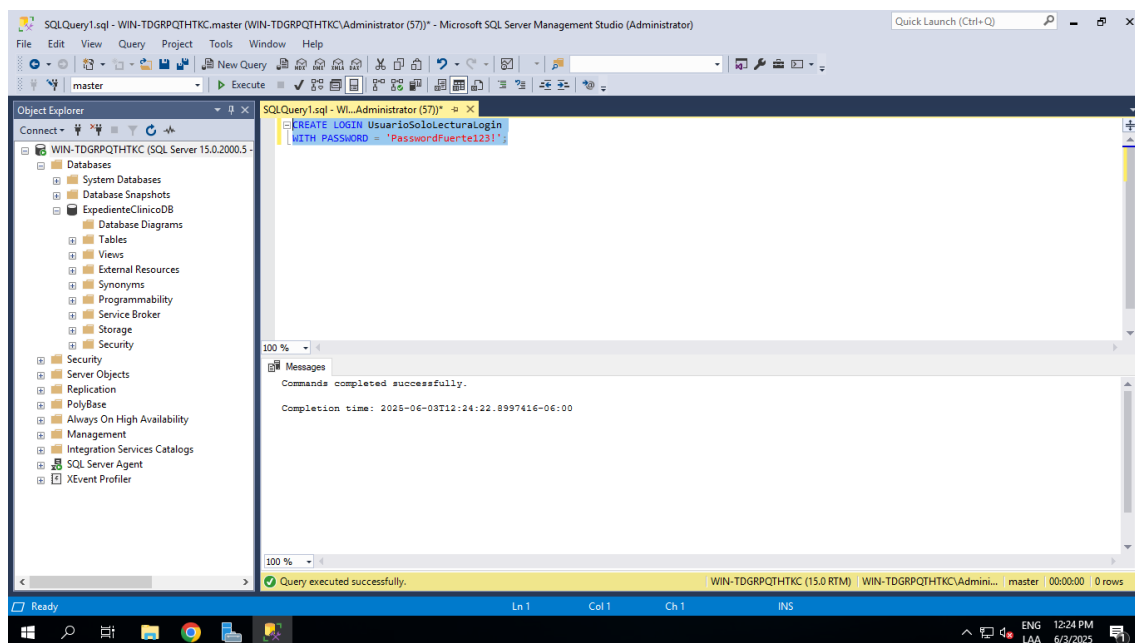
The status bar at the bottom shows 'Connected. (1/1)' and 'WIN-TDGRPQTHTKC (15.0 RTM) | WIN-TDGRPQTHTKC\Admini... | master | 00:00:00 | 0 rows'.

2. Crear Inicios de sesión, usuarios, roles y asignación de permisos.

Primer paso: Crear un nuevo inicio de sesión y un usuario en la base de datos con rol de solo lectura.

1.1 Crear el login a nivel de servidor:

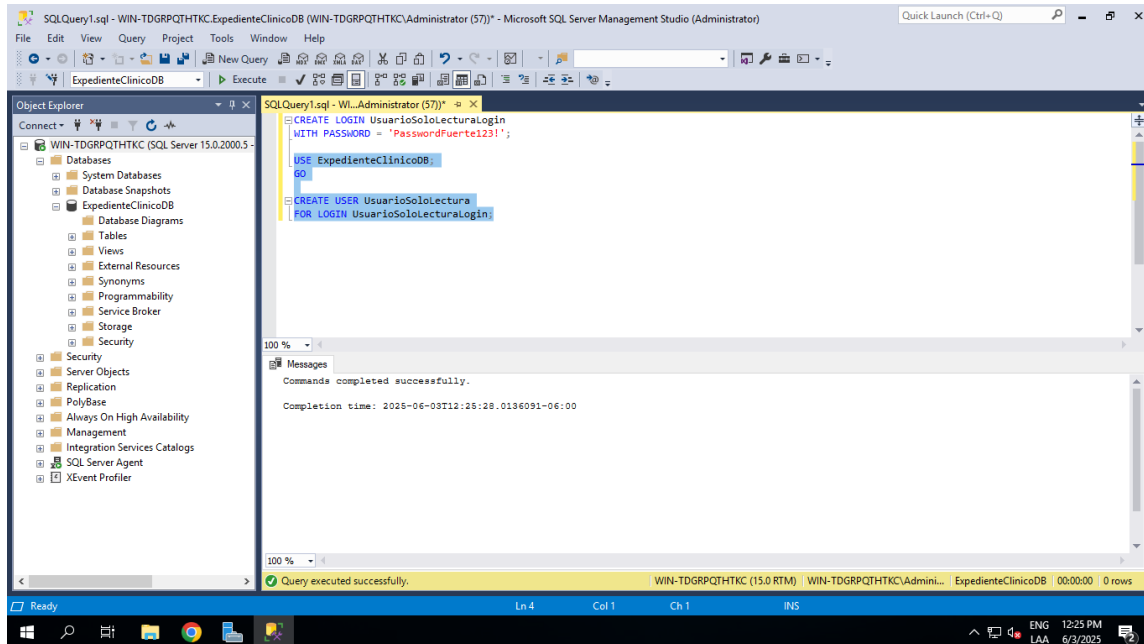
```
CREATE LOGIN UsuarioSoloLecturaLogin  
WITH PASSWORD = 'PasswordFuerte123!';
```



1.2 Crear el usuario dentro de la base de datos:

```
USE ExpedienteClinicoDB;  
GO
```

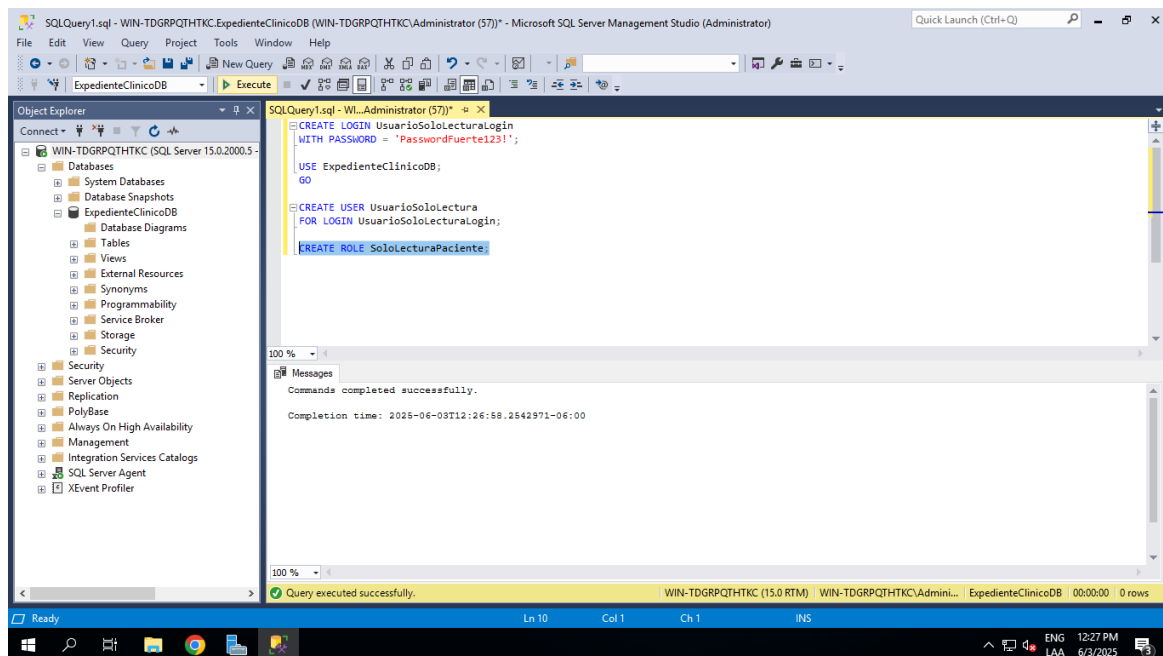
```
CREATE USER UsuarioSoloLectura  
FOR LOGIN UsuarioSoloLecturaLogin;
```

Segundo paso: Crear un rol personalizado y asignarle permisos de solo lectura sobre una tabla.

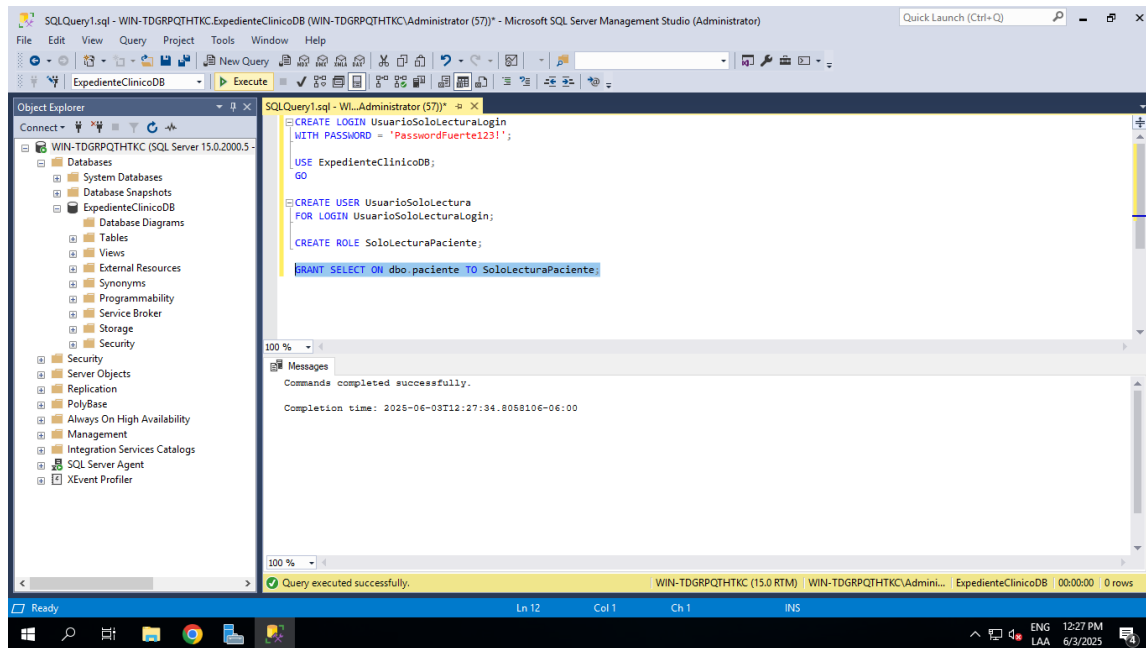
2.1 Crear rol en la base de datos:

CREATE ROLE SoloLecturaPaciente;



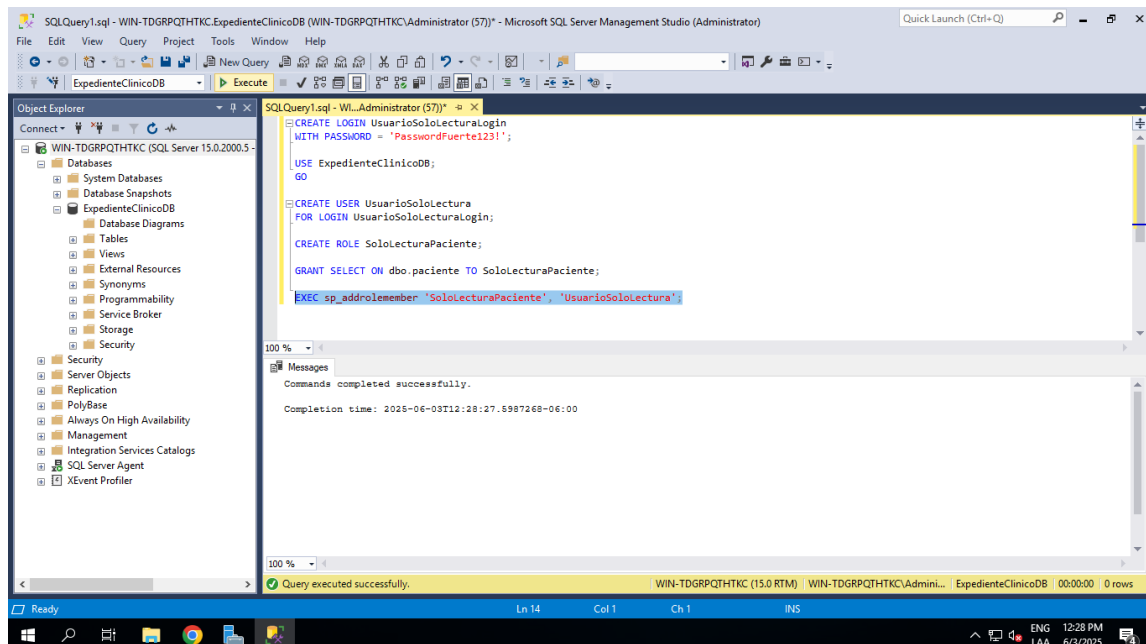
2.2 Conceder permiso SELECT sobre la tabla paciente:

GRANT SELECT ON dbo.paciente TO SoloLecturaPaciente;



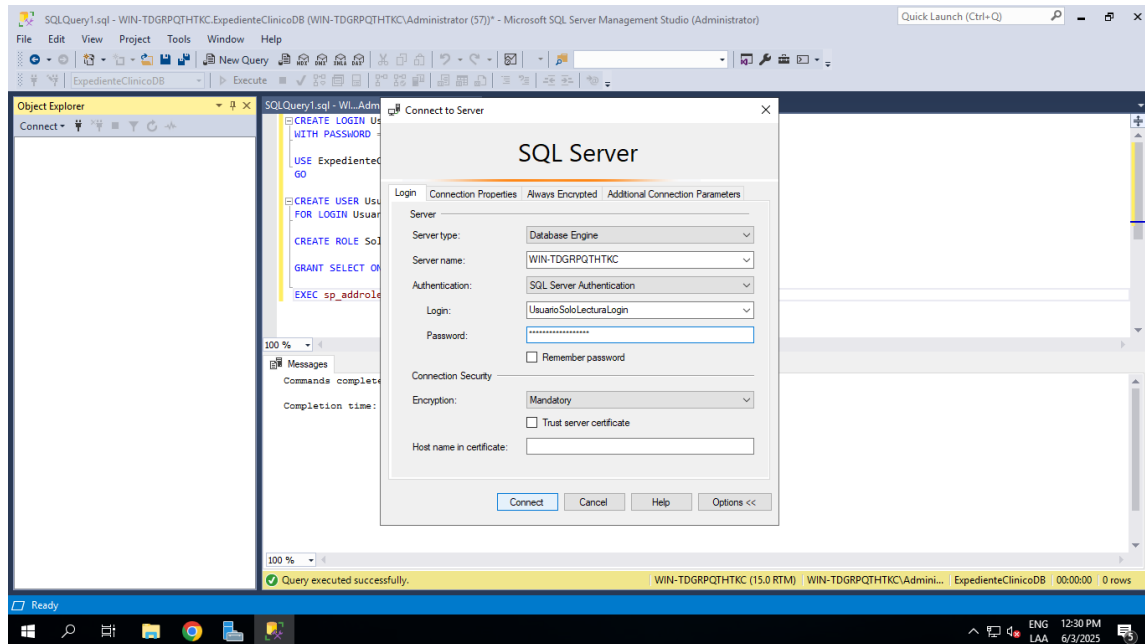
2.3 Asignar el usuario al rol:

EXEC sp_addrolemember 'SoloLecturaPaciente', 'UsuarioSoloLectura';



Tercer paso: Verificar comportamiento con permisos.

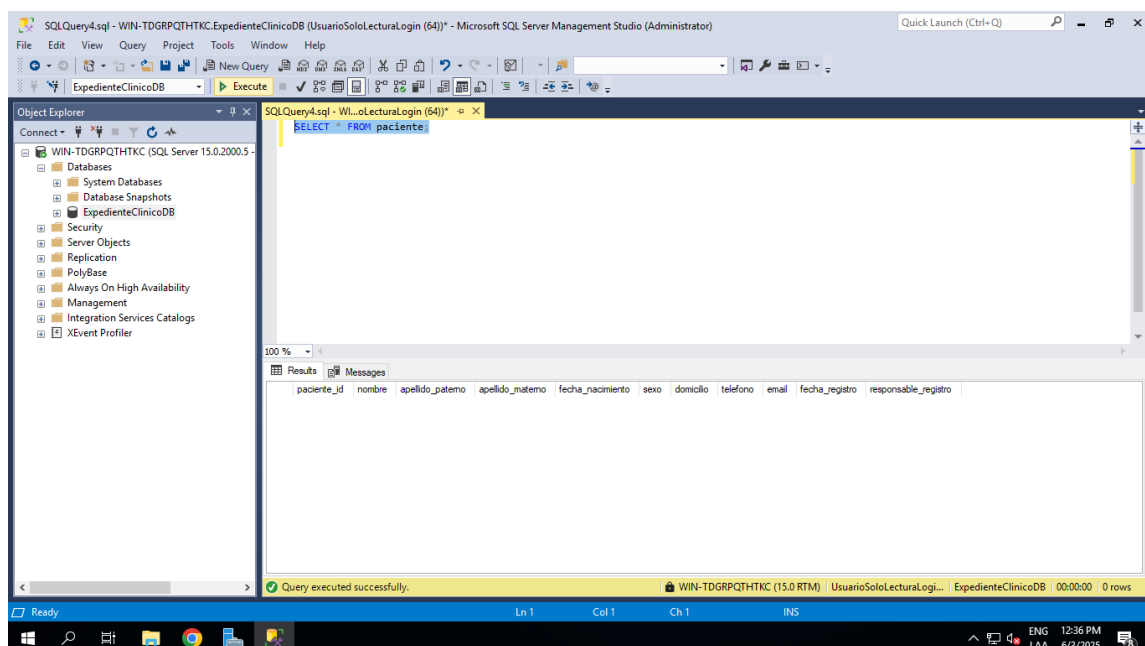
Antes de probar, hay que iniciar sesión como el usuario UsuarioSoloLecturaLogin con la Autenticación de SQL Server



3.1 Ejecutar consultas como el usuario con rol de solo lectura:

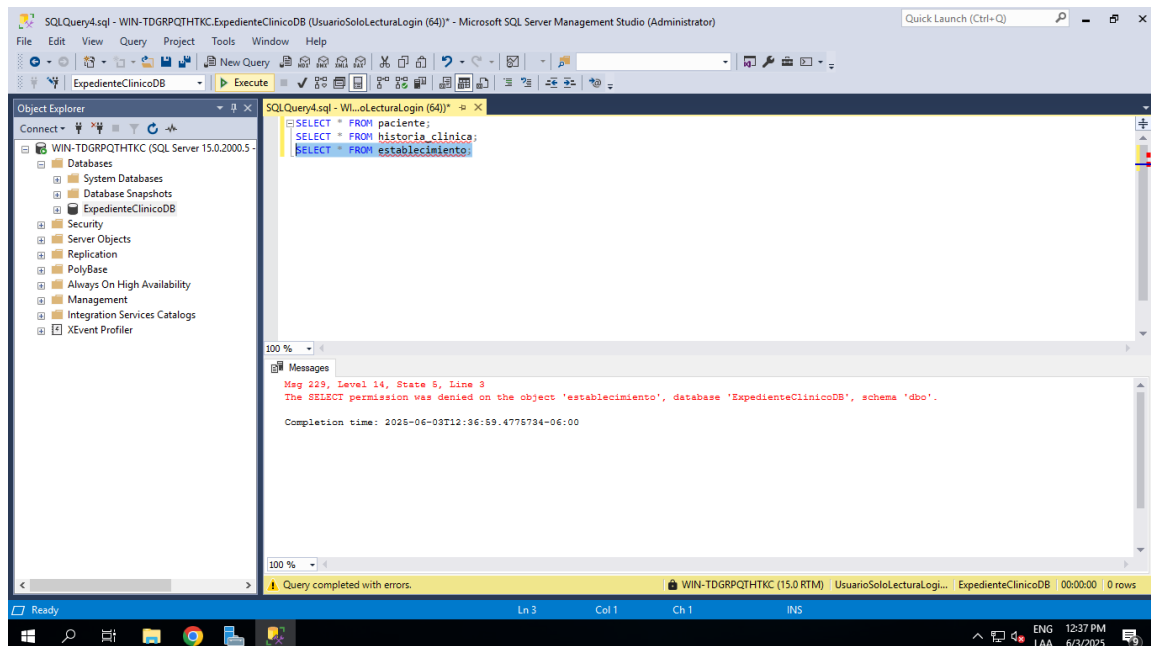
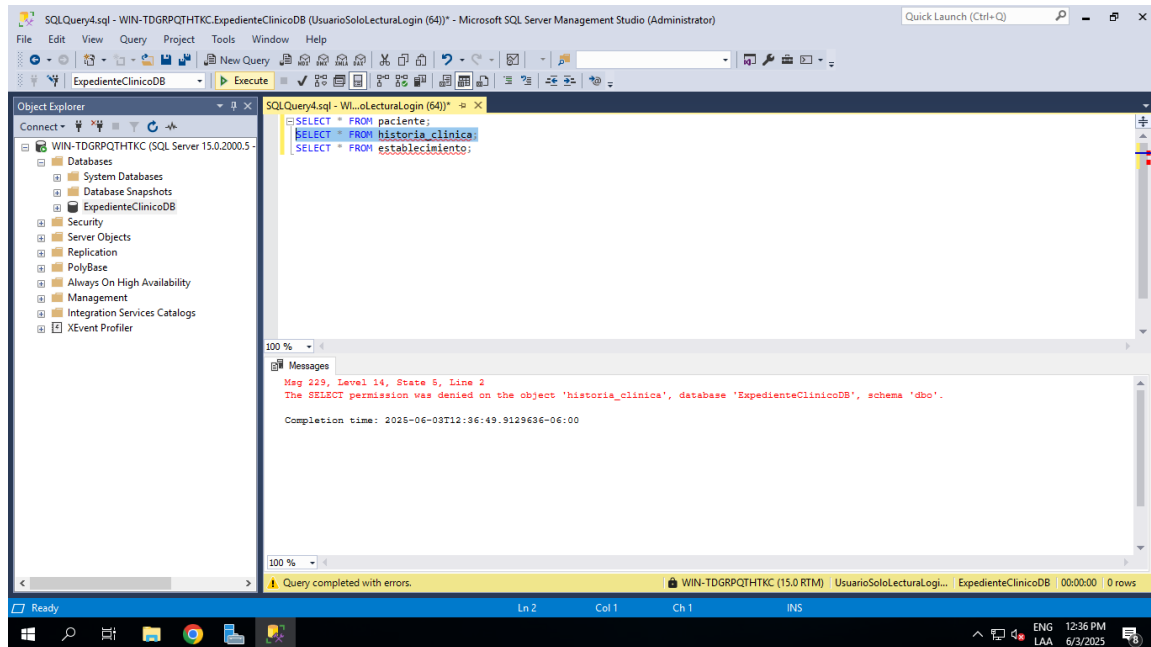
-- Esta consulta debe funcionar:

SELECT * FROM paciente;



-- Estas deben fallar:

```
SELECT * FROM historia_clinica;  
SELECT * FROM establecimiento;
```



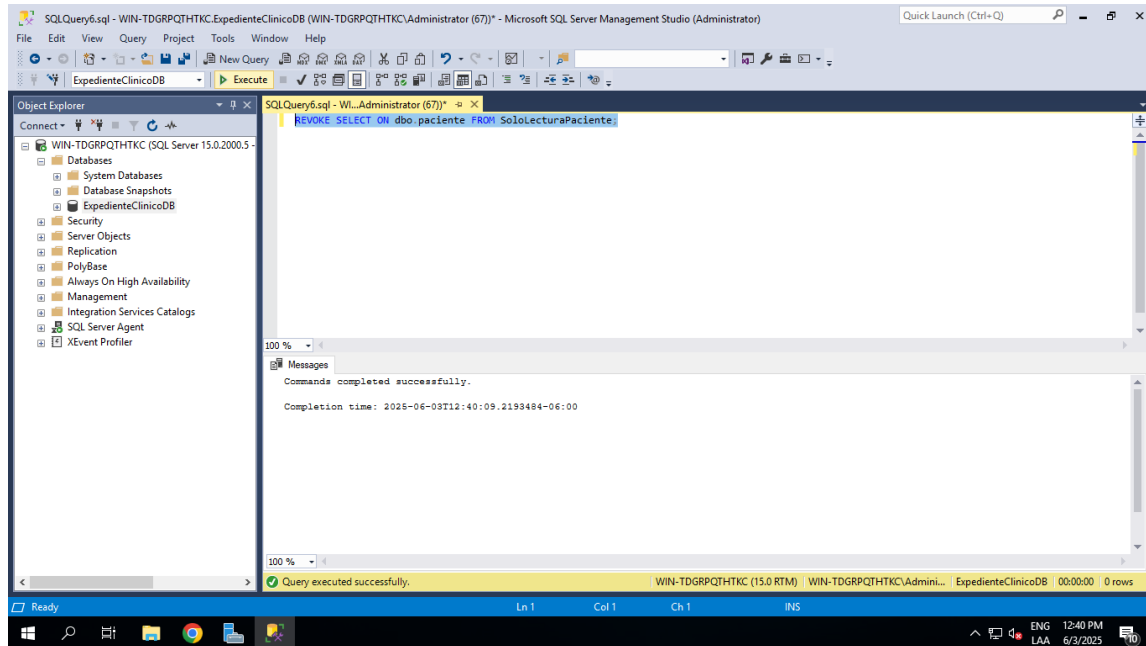
Resultado esperado:

El usuario **solo podrá consultar la tabla paciente**, y las demás generarán errores por permisos insuficientes.

Cuarto paso: ¿Cómo se podrían revocar estos permisos?

1.- Revocar permiso SELECT sobre una tabla:

REVOKE SELECT ON dbo.paciente FROM SoloLecturaPaciente;



2.- Eliminar usuario del rol:

EXEC sp_droprolemember 'SoloLecturaPaciente', 'UsuarioSoloLectura';

3.- Eliminar el rol personalizado:

DROP ROLE SoloLecturaPaciente;

4.- Eliminar el usuario de la base de datos:

DROP USER UsuarioSoloLectura;

5.- Eliminar login del servidor:

DROP LOGIN UsuarioSoloLecturaLogin;

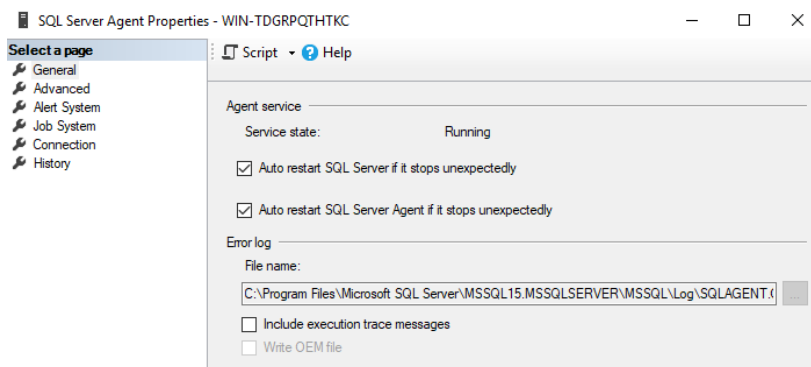
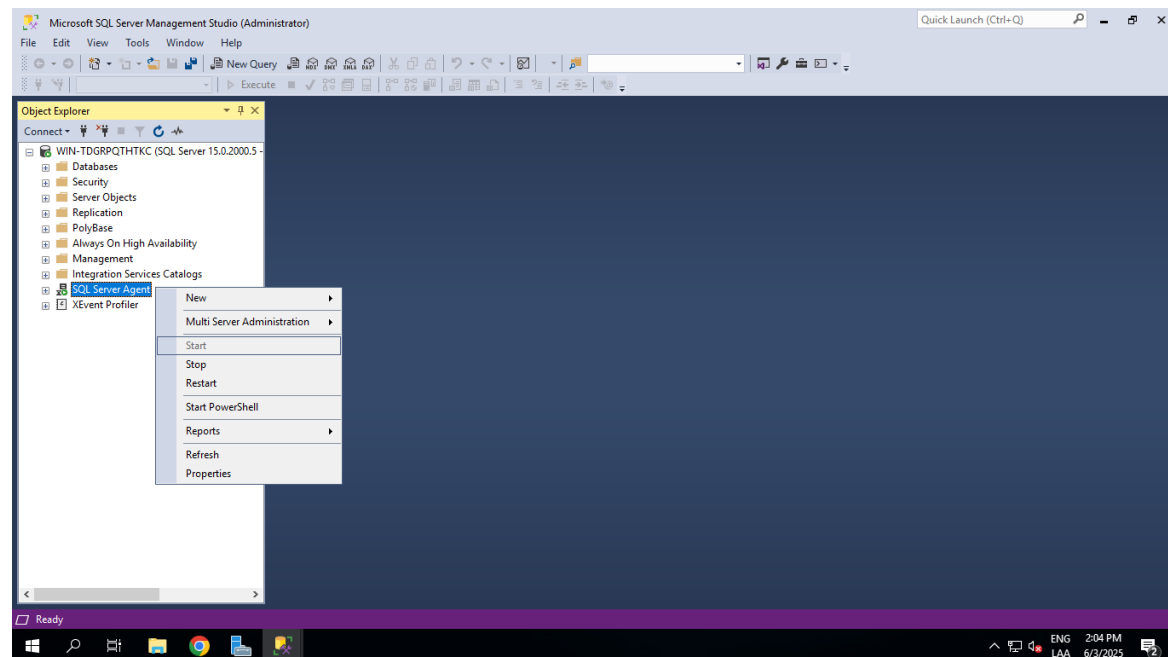
3. Crear y programar tareas automáticas en SQL Server

Paso 1: Activar SQL Server Agent

El SQL Server Agent debe estar activo para crear y ejecutar Jobs.

✓ Cómo activarlo:

En el explorador de objetos, hay que darle click derecho sobre SQL Server Agent y seleccionar "Start" si está detenido

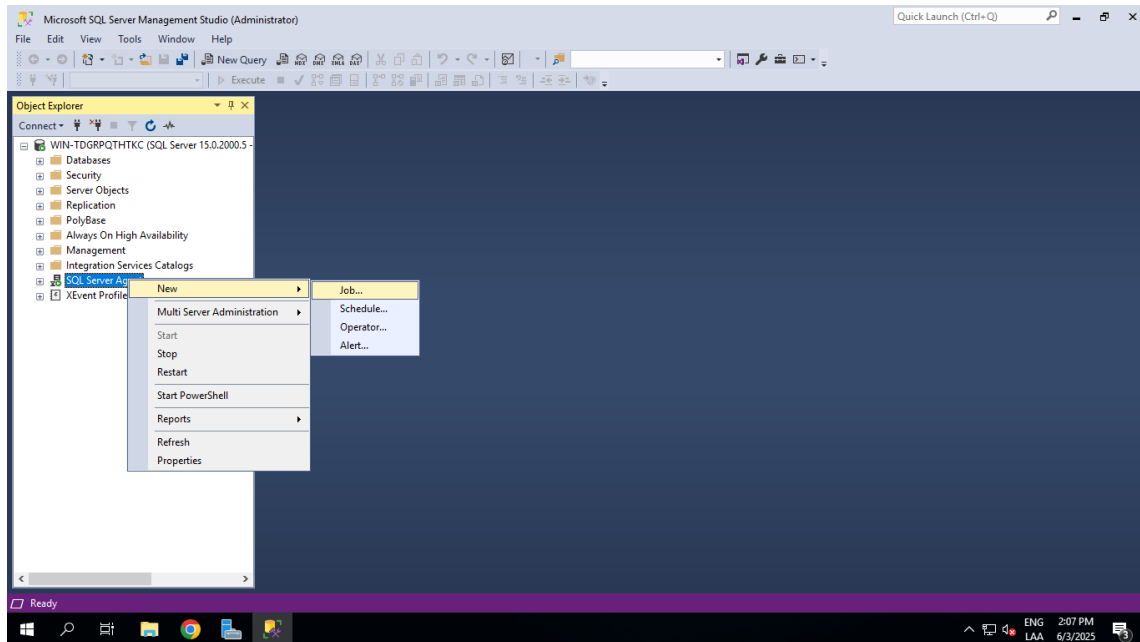


Como se puede apreciar el SQL Server Agent ya está corriendo.

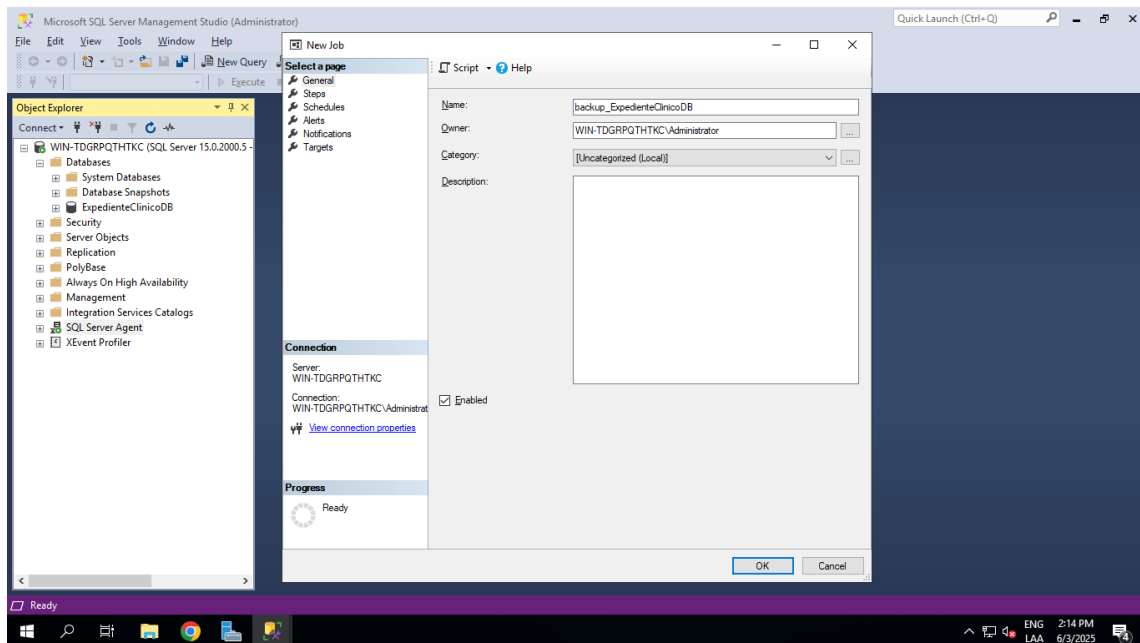
Paso 2: Crear el Job de respaldo automático

✓ Crear el Job manualmente

Damos click derecho a SQL Server Agent -> New -> Job



Asignamos un nombre, en este caso: backup_ExpedienteClinicoDB

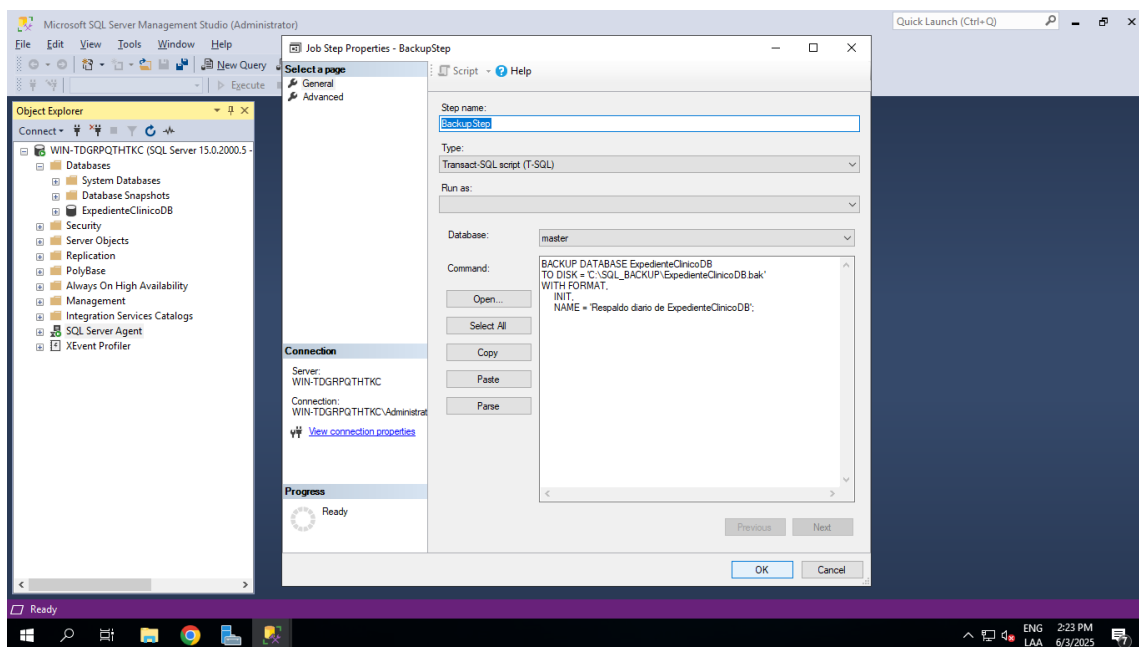


Agregamos un nuevo Step:

- Nombre del paso: BackupStep
- Tipo: Transact-SQL script (T-SQL)
- Base de datos: master

Y agregamos el siguiente comando:

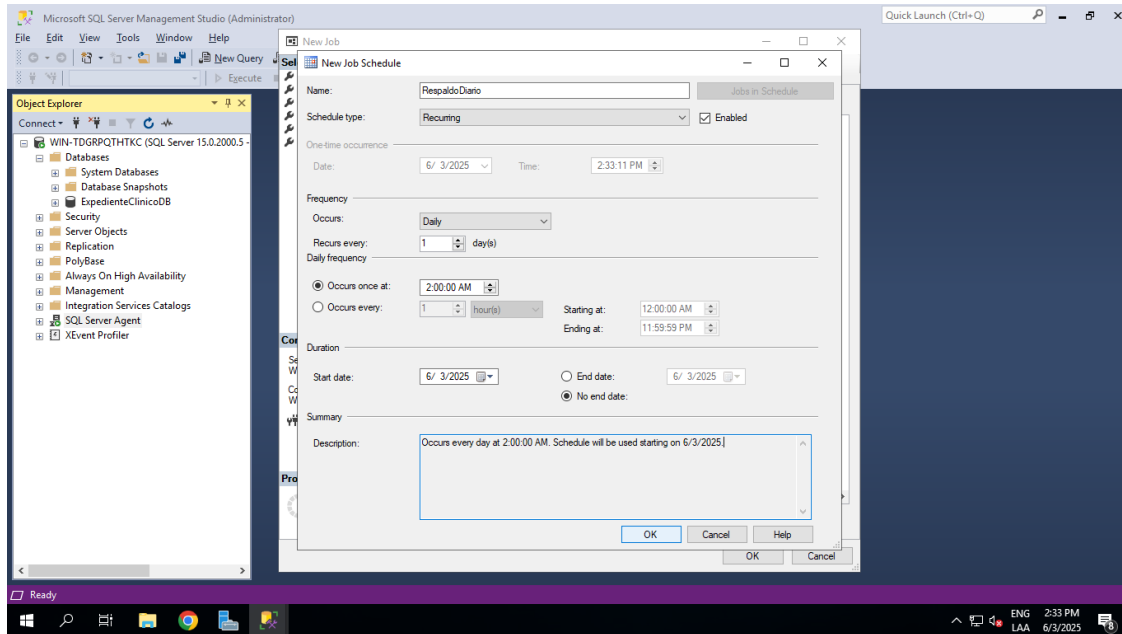
```
BACKUP DATABASE ExpedienteClinicoDB  
TO DISK = 'C:\SQL_BACKUP\ExpedienteClinicoDB.bak'  
WITH FORMAT,  
INIT,  
NAME= 'Respaldo diario de ExpedienteClinicoDB';
```



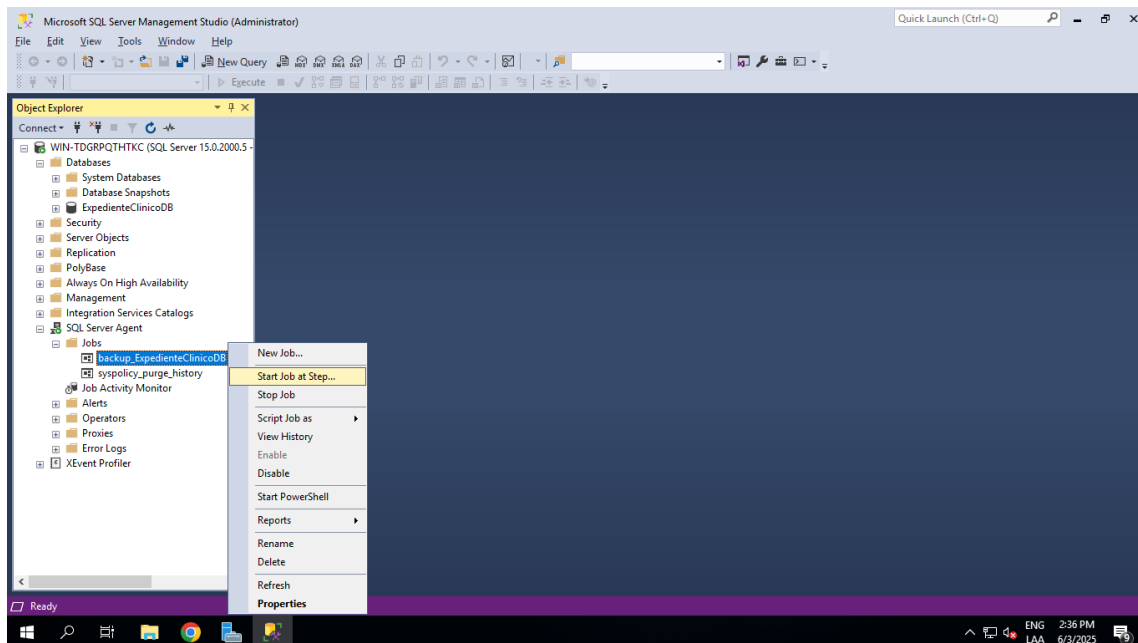
Paso 3: Programar el Job para que se ejecute diariamente.

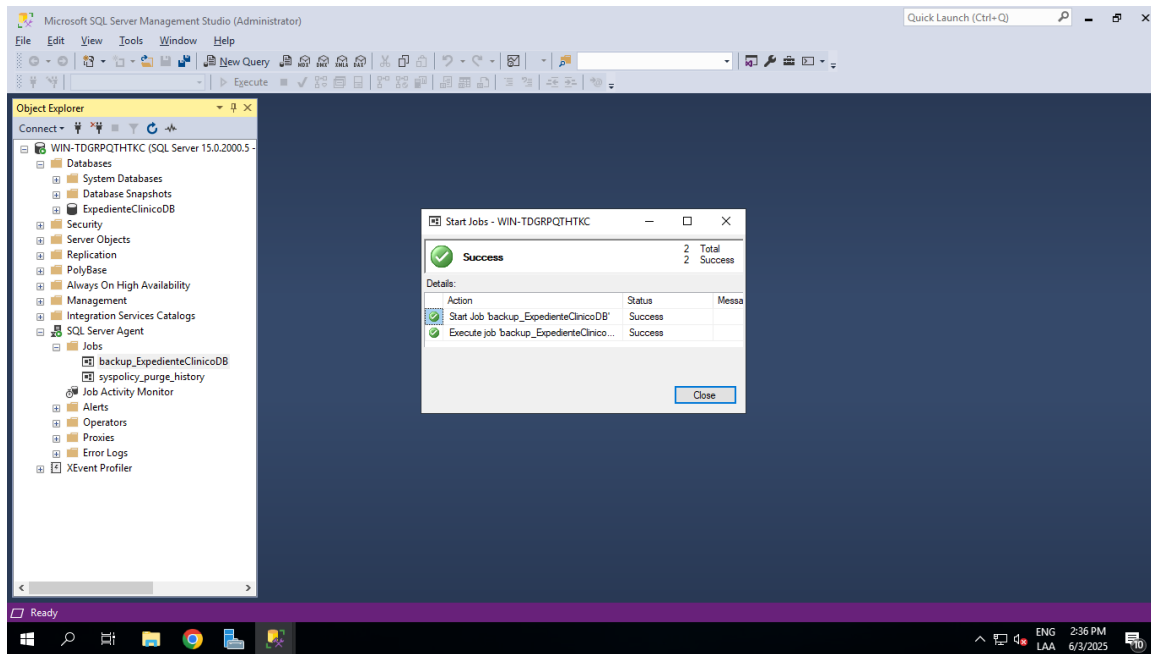
Nos dirigimos a la pestaña "Schedules" (Programaciones) y creamos una nueva programación:

- Nombre: RespaldoDiario
- Frecuencia: Daily
- Hora de ejecución: ejemplo 02:00:00 AM.



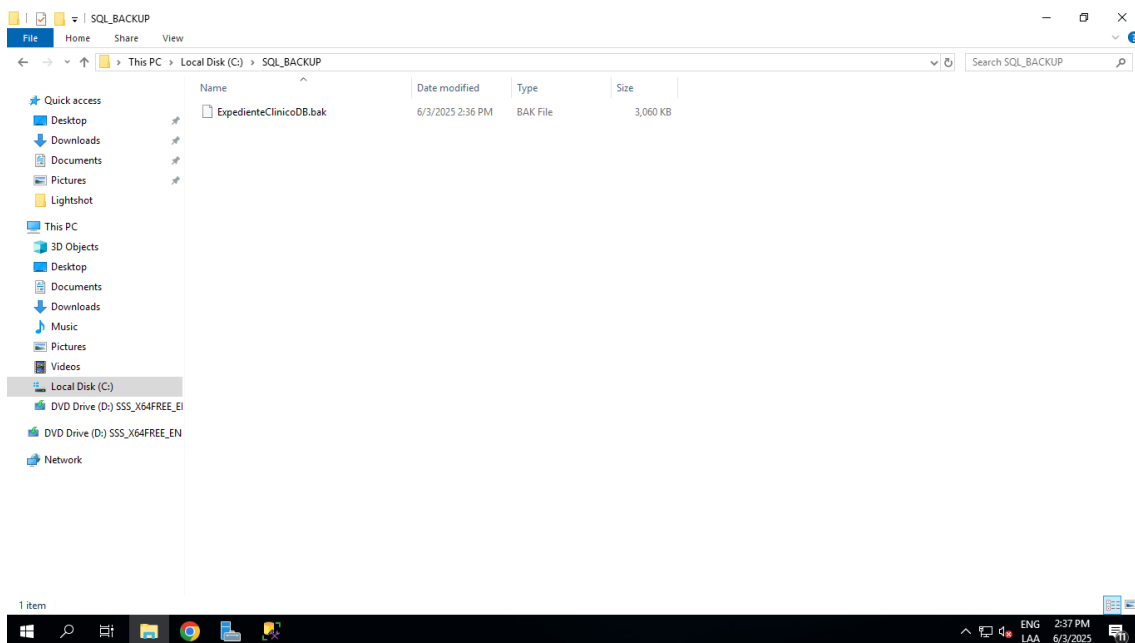
Paso 4: Lanzar y probar el Job manualmente:





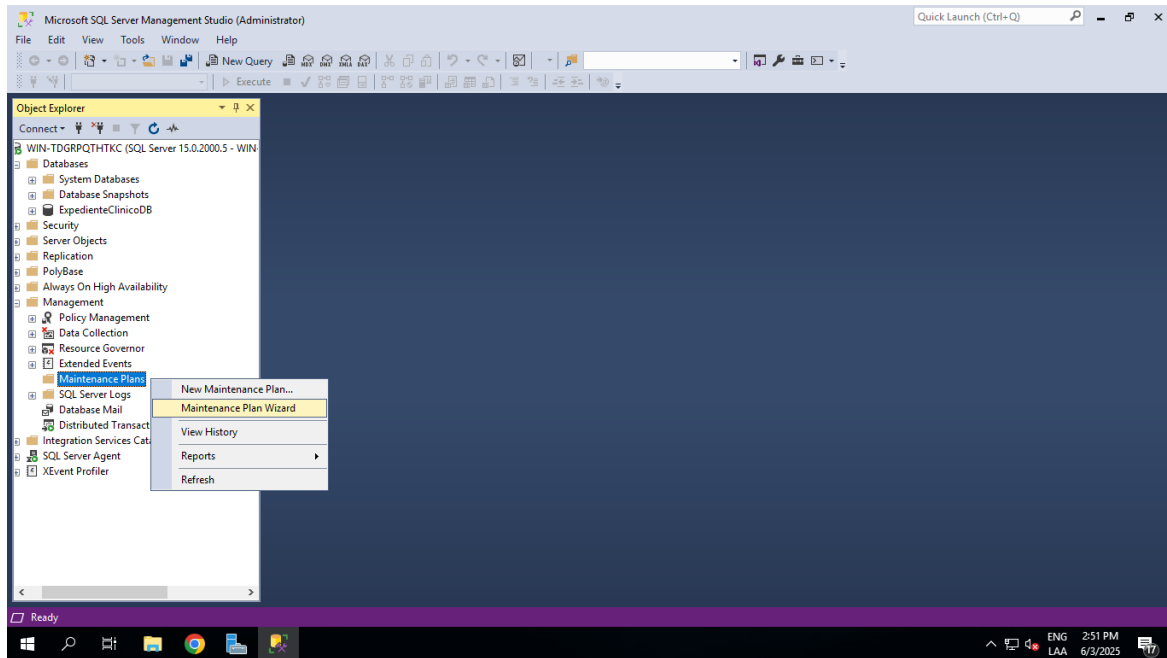
Paso 5: Verificar archivo de respaldo generado.

Vamos a la carpeta donde configuramos el respaldo y nos aseguramos que el archivo ExpedienteClinicoDB.bak se encuentra ahí.



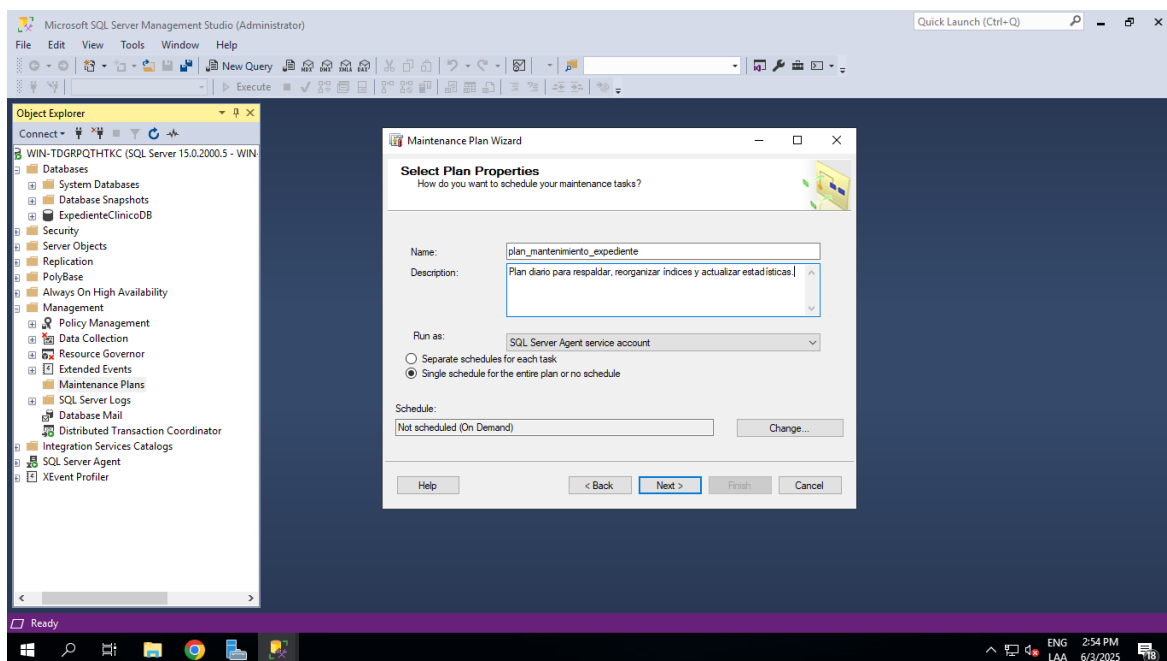
4. Crear un nuevo plan de mantenimiento

Paso 1: Abrir el asistente para plan de mantenimiento



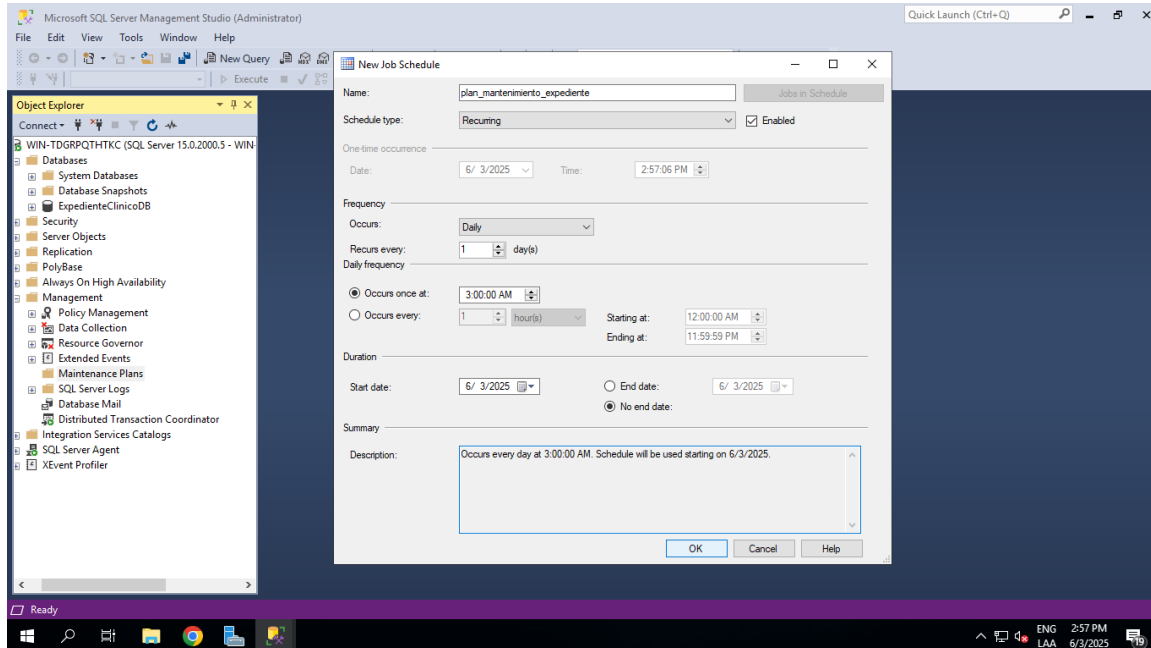
Paso 2: Configuración del plan.

- **Nombre del plan:** plan_mantenimiento_expediente
- **Tipo:** Seleccionamos Single schedule for the entire plan or no schedule.



Paso 3: Programar el plan.

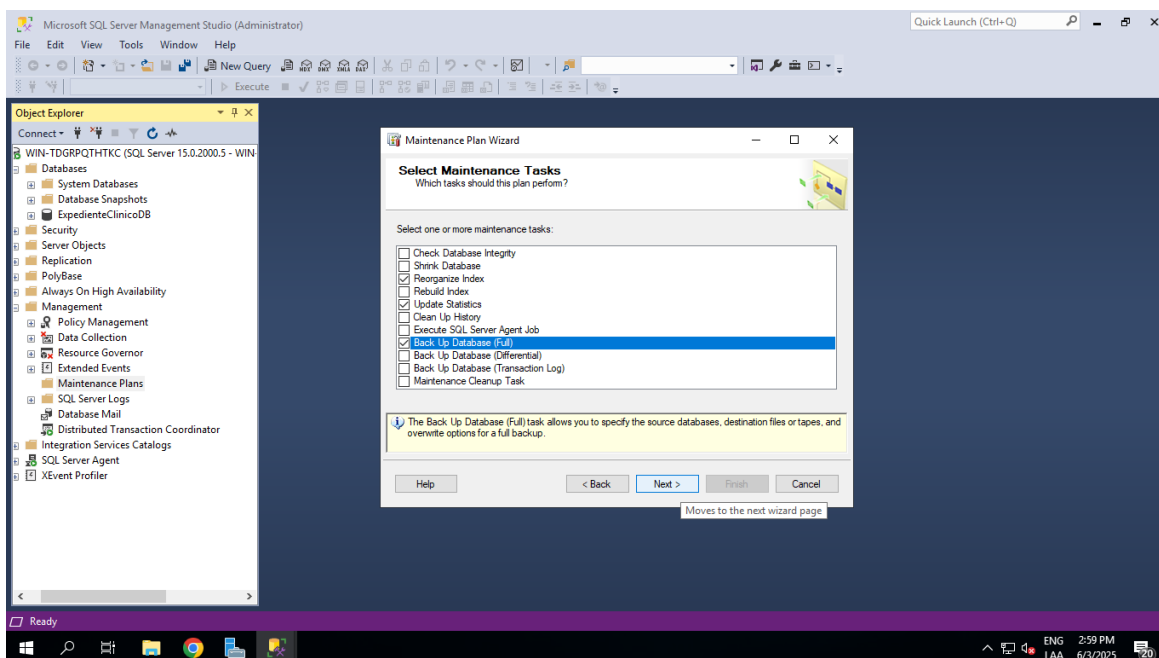
- **Nombre de la programación:** programacion_diaria
- **Frecuencia:** Diariamente
- **Hora:** Por ejemplo, 03:00 AM



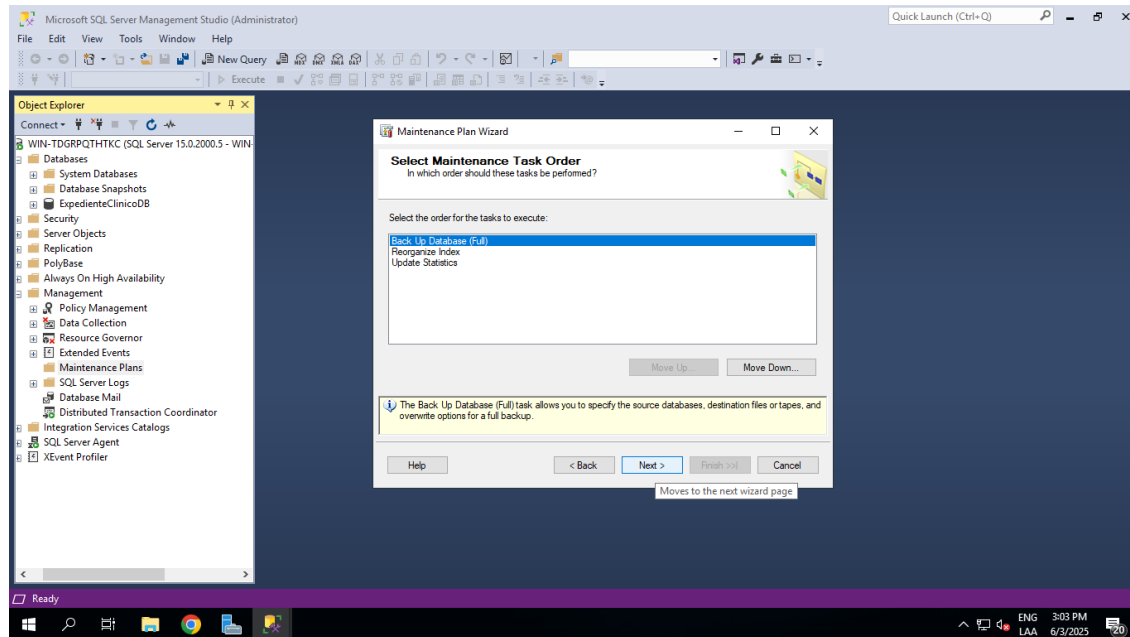
Paso 4: Selección de tareas.

Vamos a marcar las siguientes opciones:

- **Back Up Database (Full)**
- **Reorganize Index**
- **Update Statistics**

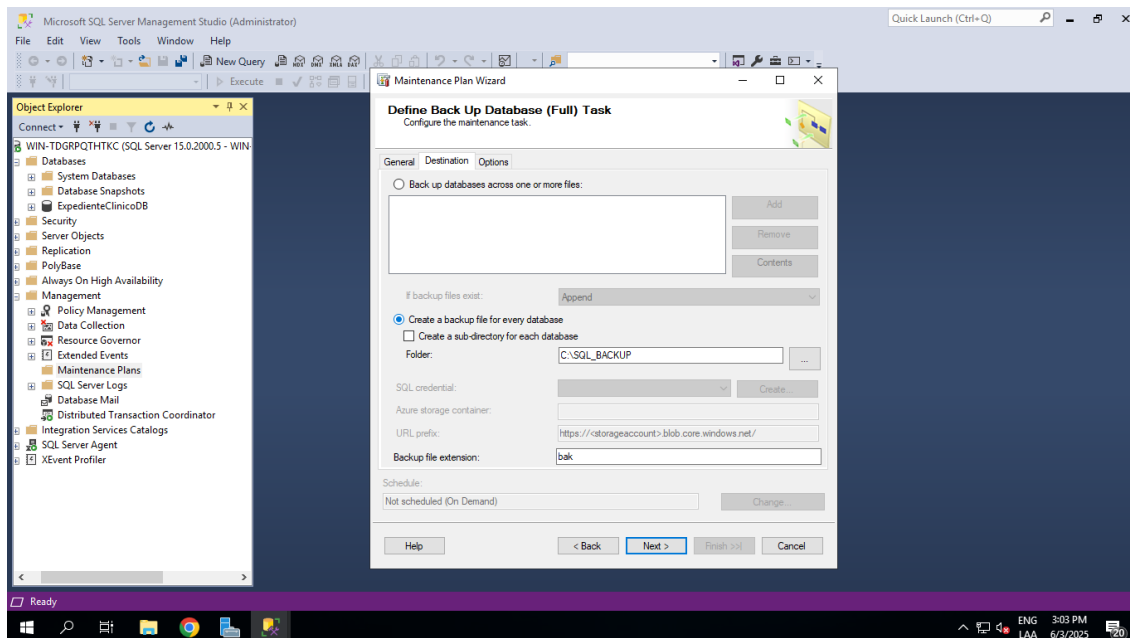


Paso 5: Ordenar tareas secuencialmente.

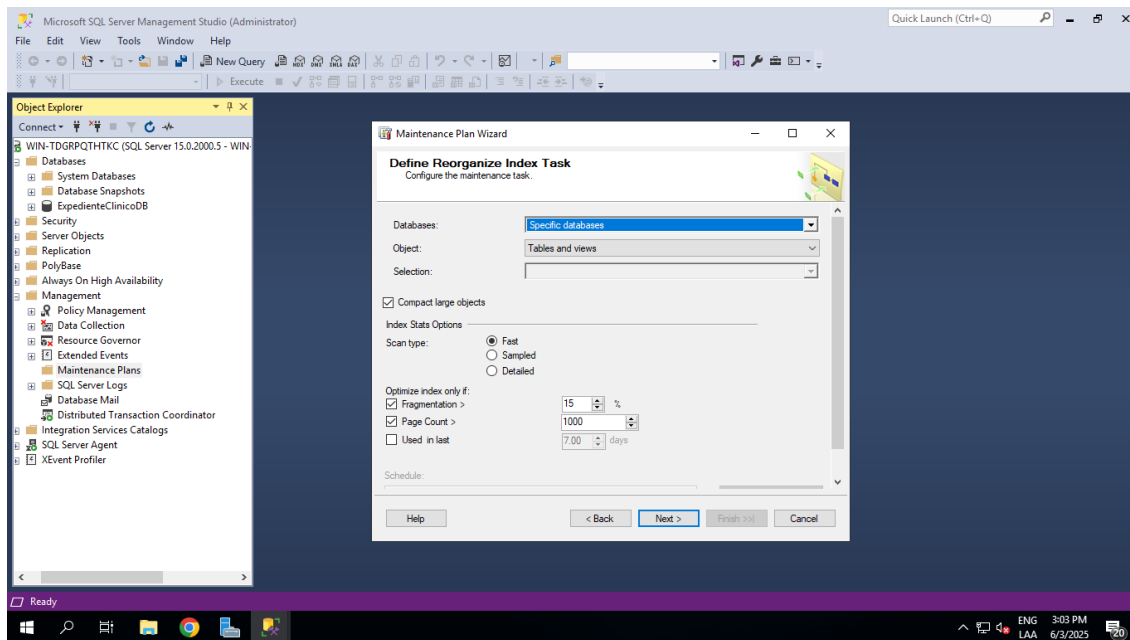


Paso 6: Configurar las tareas.

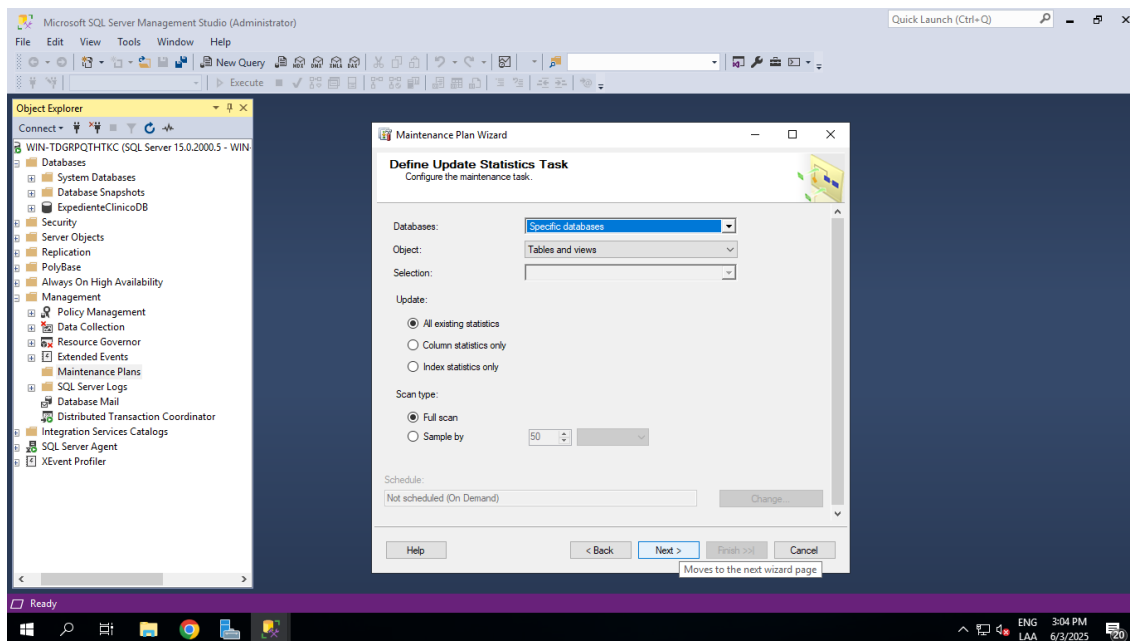
Backup Full:



Reorganizar Índices:

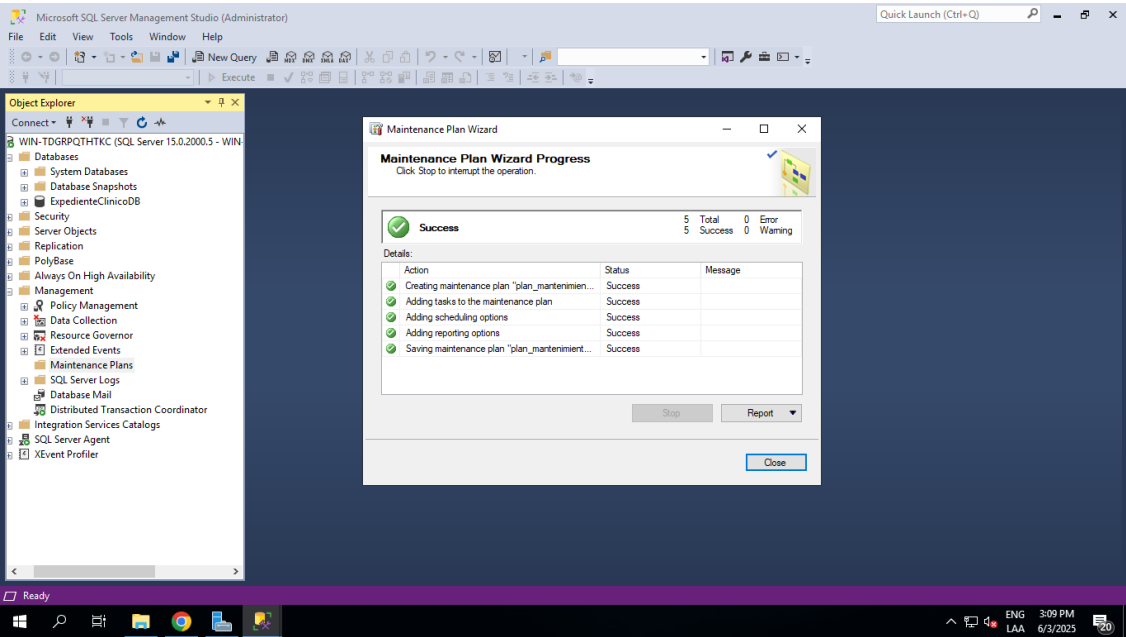
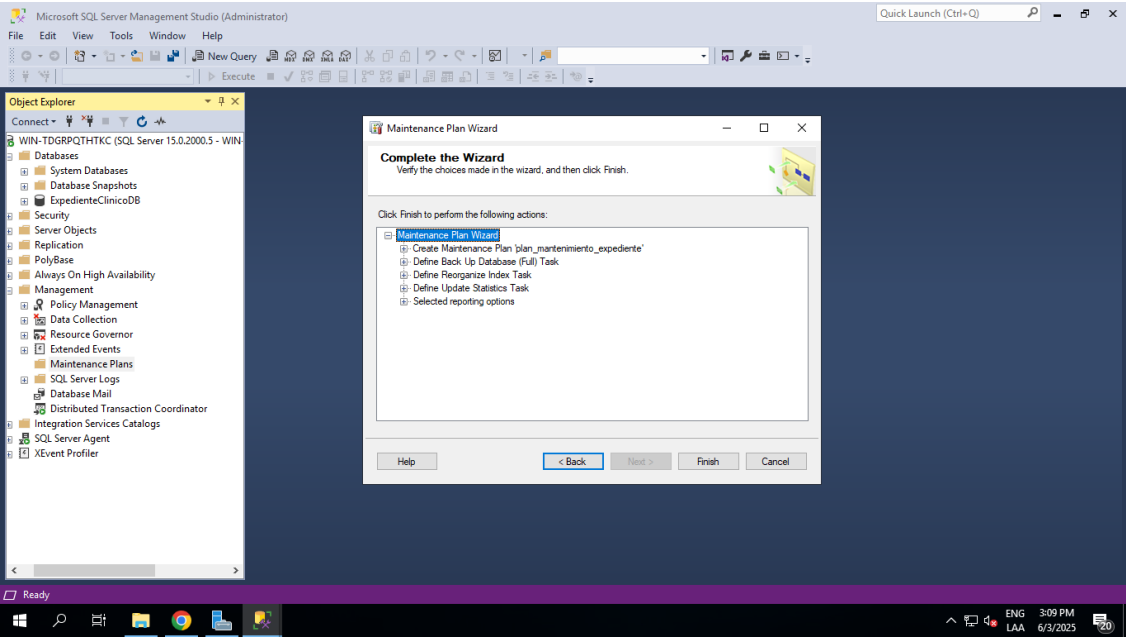


Actualizar estadísticas:

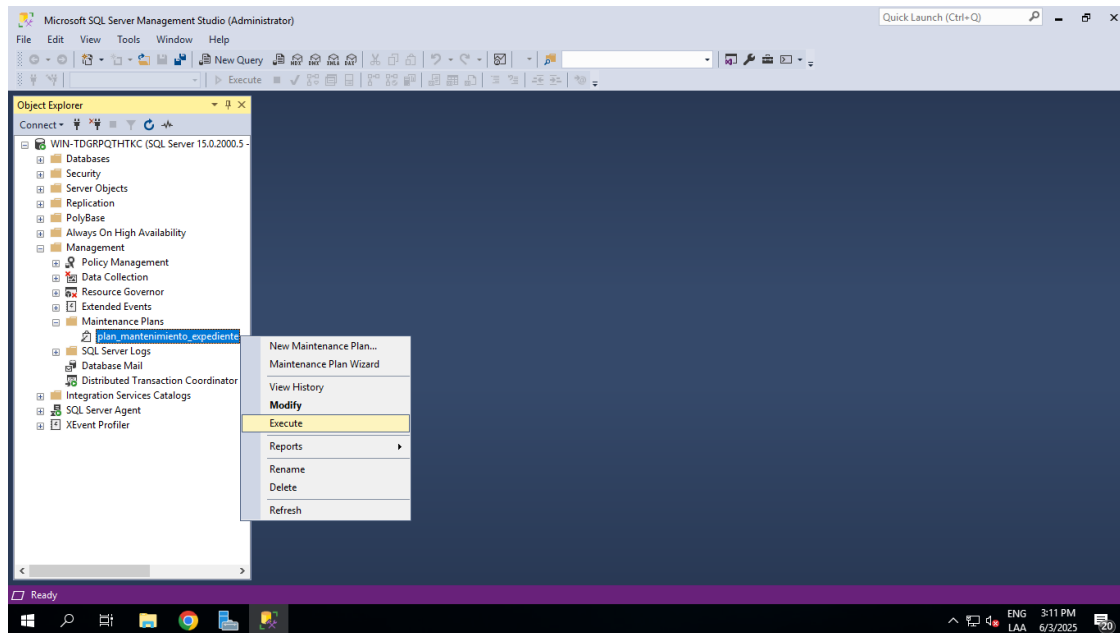


Paso 7: Guardar, ejecutar y verificar.

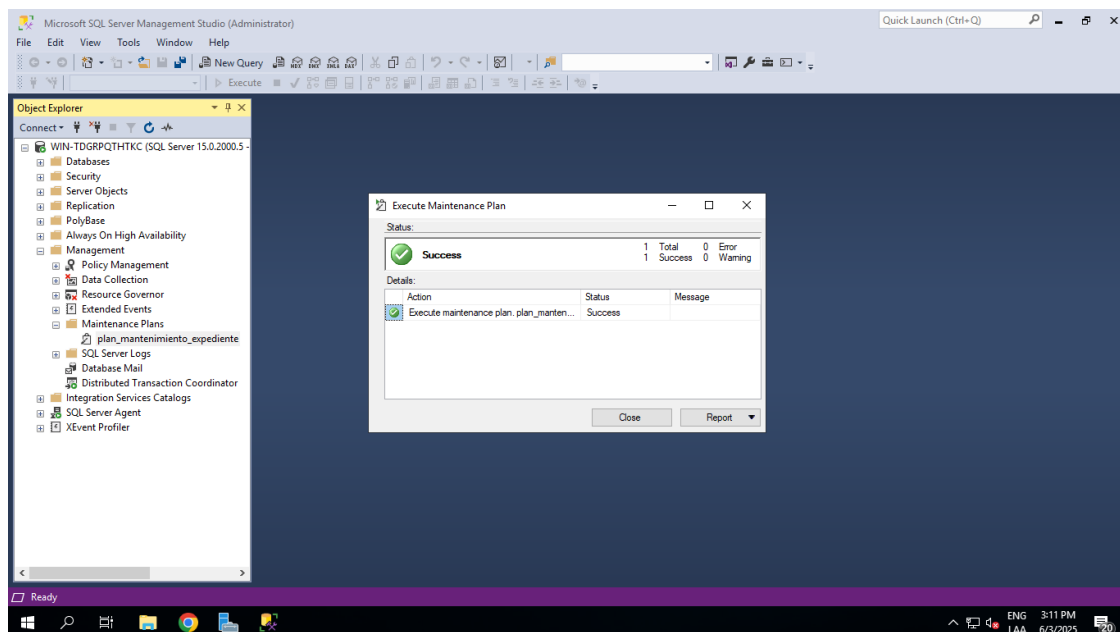
Primero guardamos el plan que acabamos de crear.



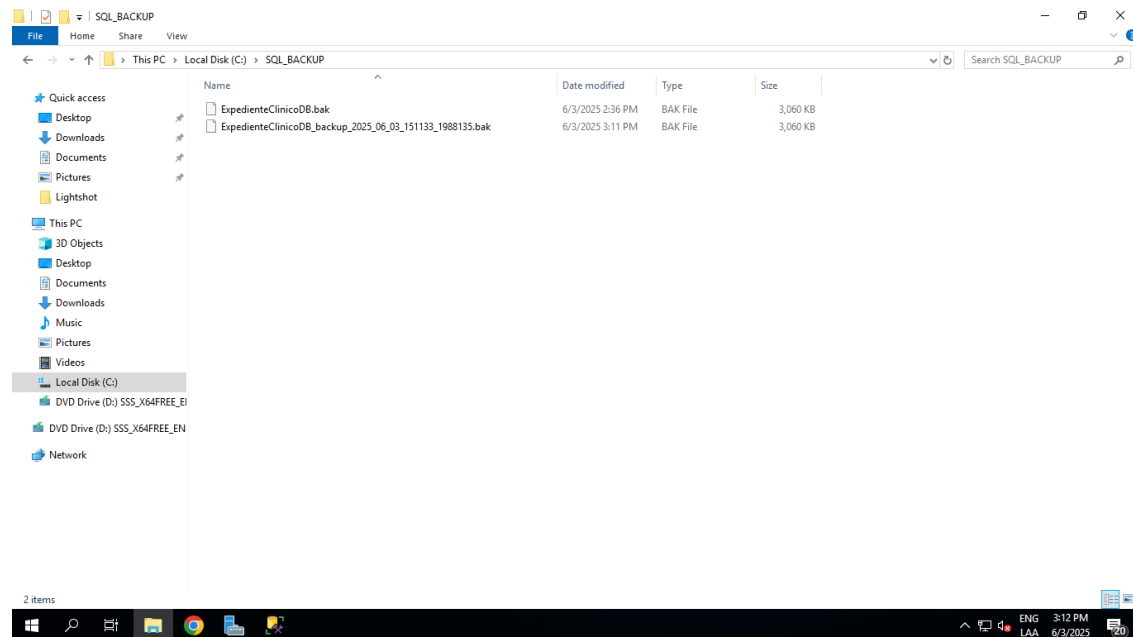
Ahora vamos a ejecutar el plan de mantenimiento:



El plan se ejecuta sin ningún tipo de error.



Y el archivo bak se encuentra en la carpeta de SQL_BACKUP



5. Modificar la base de datos

Agregar a todas las tablas un campo boolean de nombre deleted que no permita nulos con valor por omisión false

```
USE expediente_clinico_db;  
GO
```

```
ALTER TABLE dbo.establecimiento  
ADD deleted BIT NOT NULL CONSTRAINT df_establecimiento_deleted DEFAULT 0;  
GO
```

```
ALTER TABLE dbo.paciente  
ADD deleted BIT NOT NULL CONSTRAINT df_paciente_deleted DEFAULT 0;  
GO
```

```
ALTER TABLE dbo.expediente_clinico  
ADD deleted BIT NOT NULL CONSTRAINT df_expediente_clinico_deleted DEFAULT 0;  
GO
```

```
ALTER TABLE dbo.personal_salud  
ADD deleted BIT NOT NULL CONSTRAINT df_personal_salud_deleted DEFAULT 0;  
GO
```

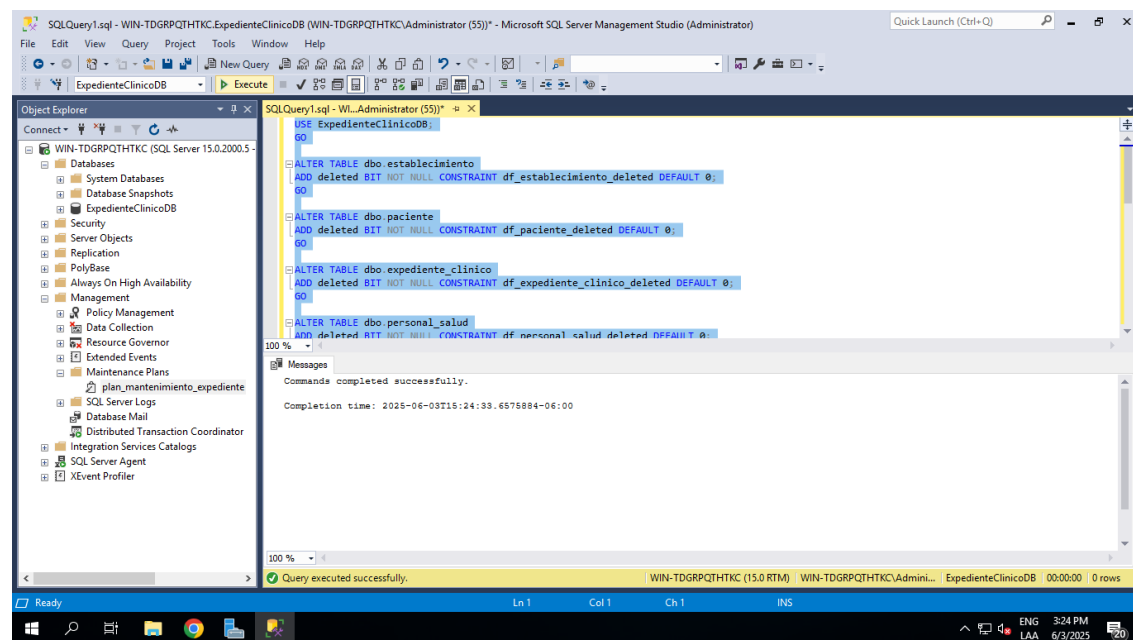
```
ALTER TABLE dbo.historia_clinica  
ADD deleted BIT NOT NULL CONSTRAINT df_historia_clinica_deleted DEFAULT 0;  
GO
```

```
ALTER TABLE dbo.nota_evolucion  
ADD deleted BIT NOT NULL CONSTRAINT df_nota_evolucion_deleted DEFAULT 0;  
GO
```

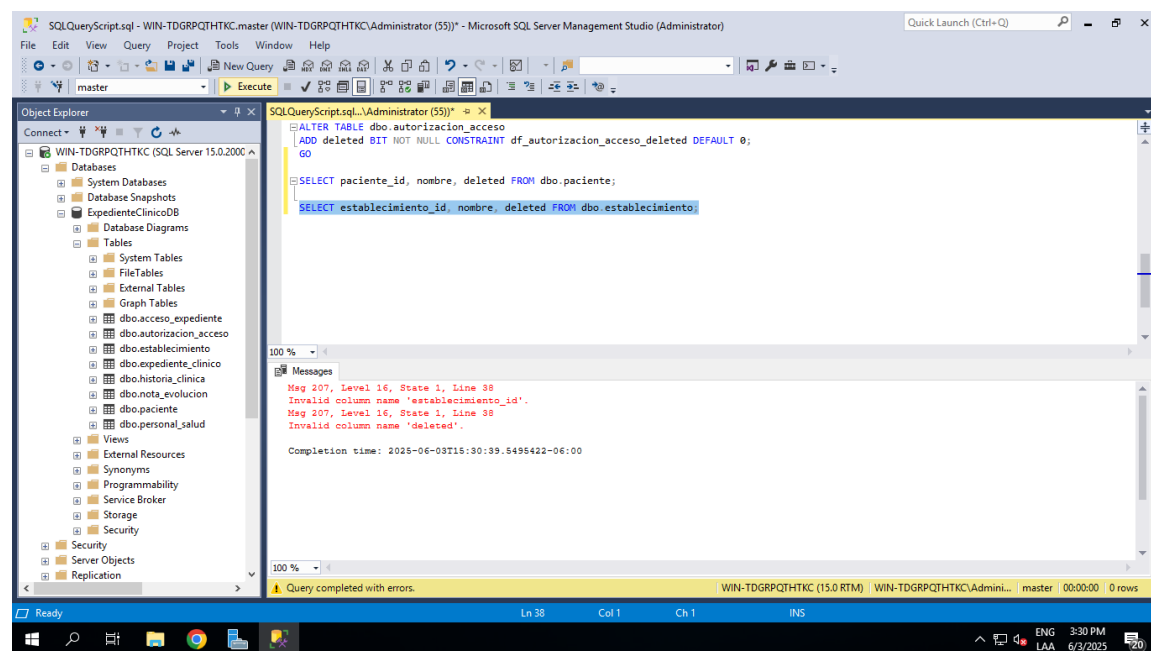
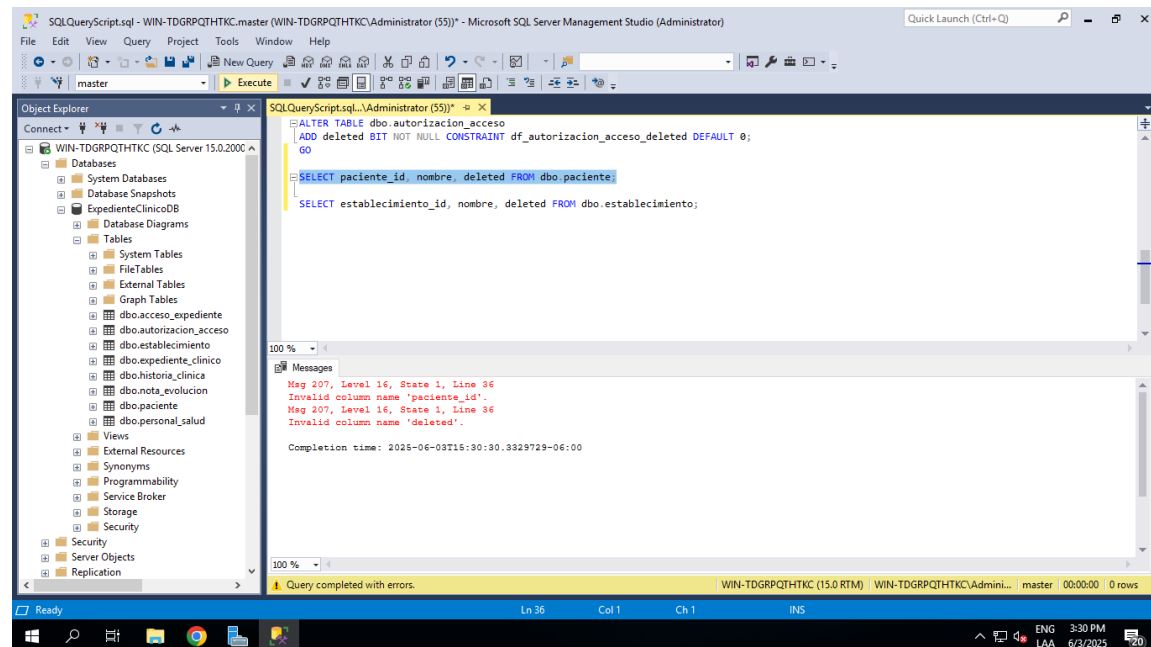
```
ALTER TABLE dbo.acceso_expediente  
ADD deleted BIT NOT NULL CONSTRAINT df_acceso_expediente_deleted DEFAULT 0;  
GO
```

```
ALTER TABLE dbo.autorizacion_acceso  
ADD deleted BIT NOT NULL CONSTRAINT df_autorizacion_acceso_deleted DEFAULT 0;  
GO
```

Ejecución exitosa del ScriptSQL:

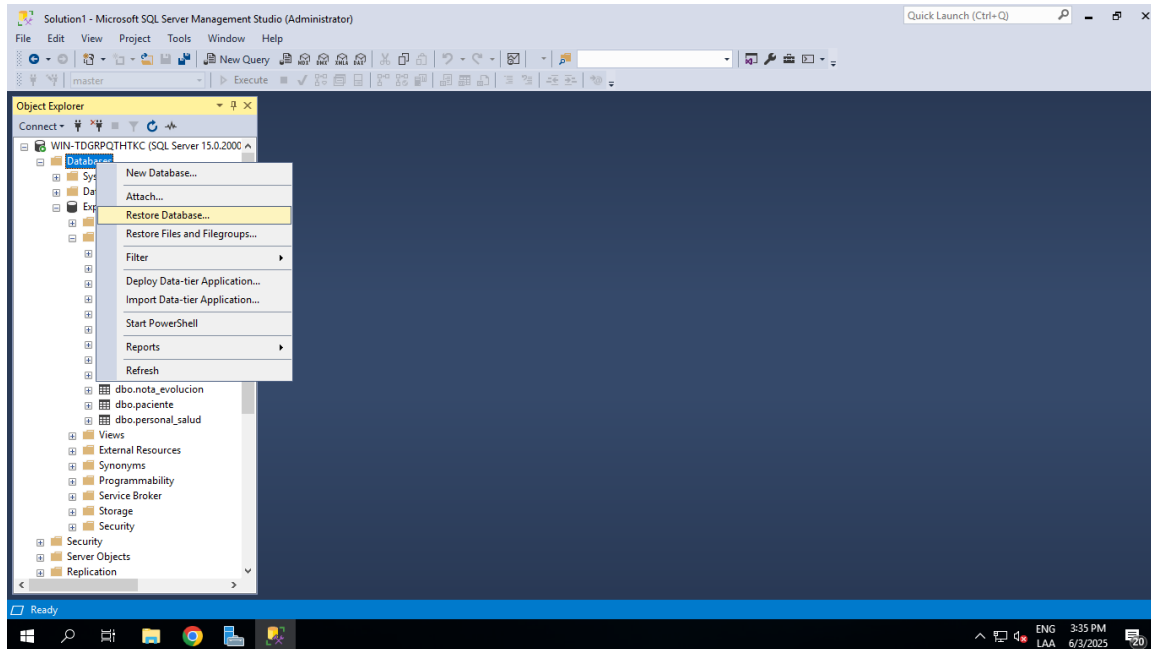


Resultado de los comandos en los datos:

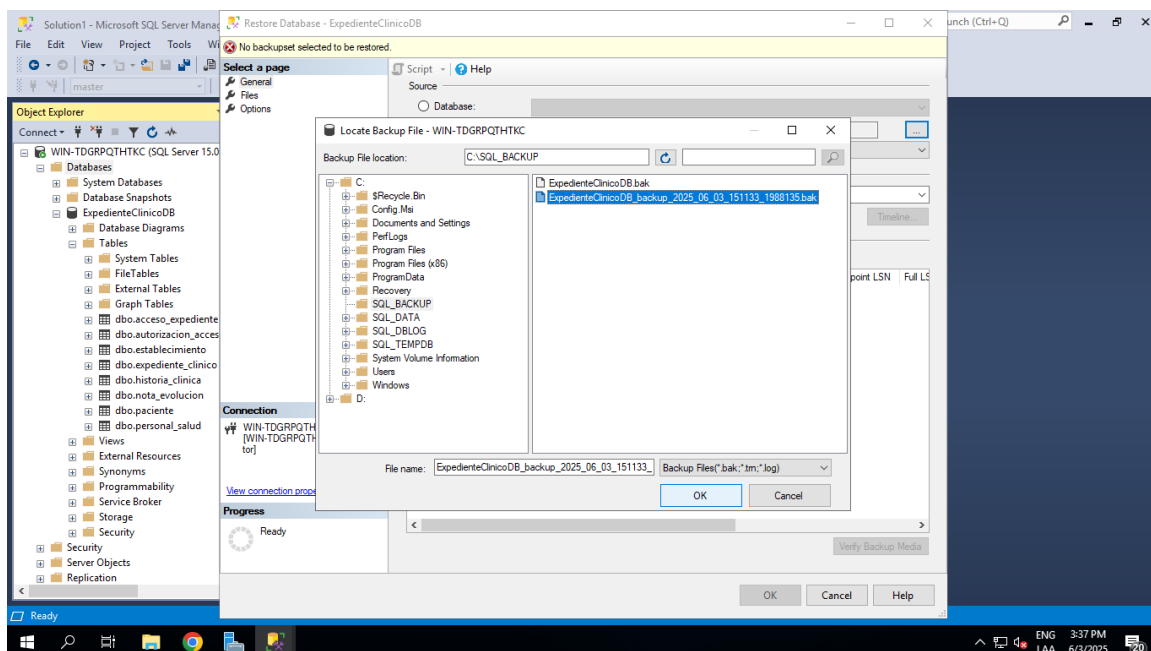


6. Restaurar la base de datos

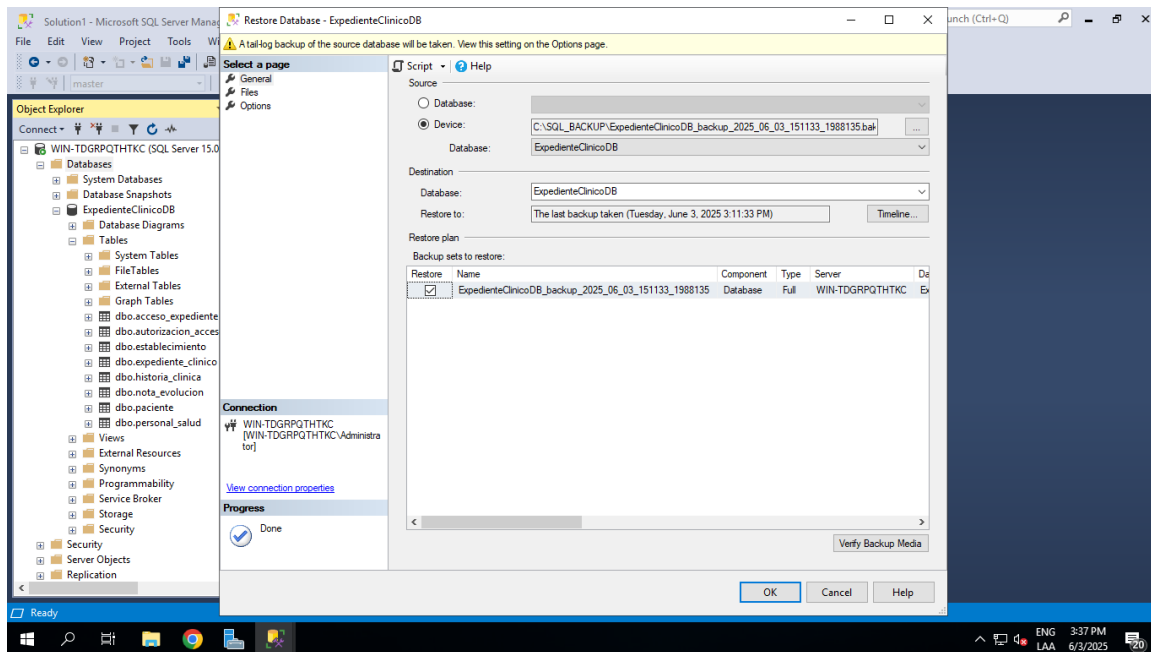
Hacemos click derecho en la opción Databases y le damos click a Restore Databases:



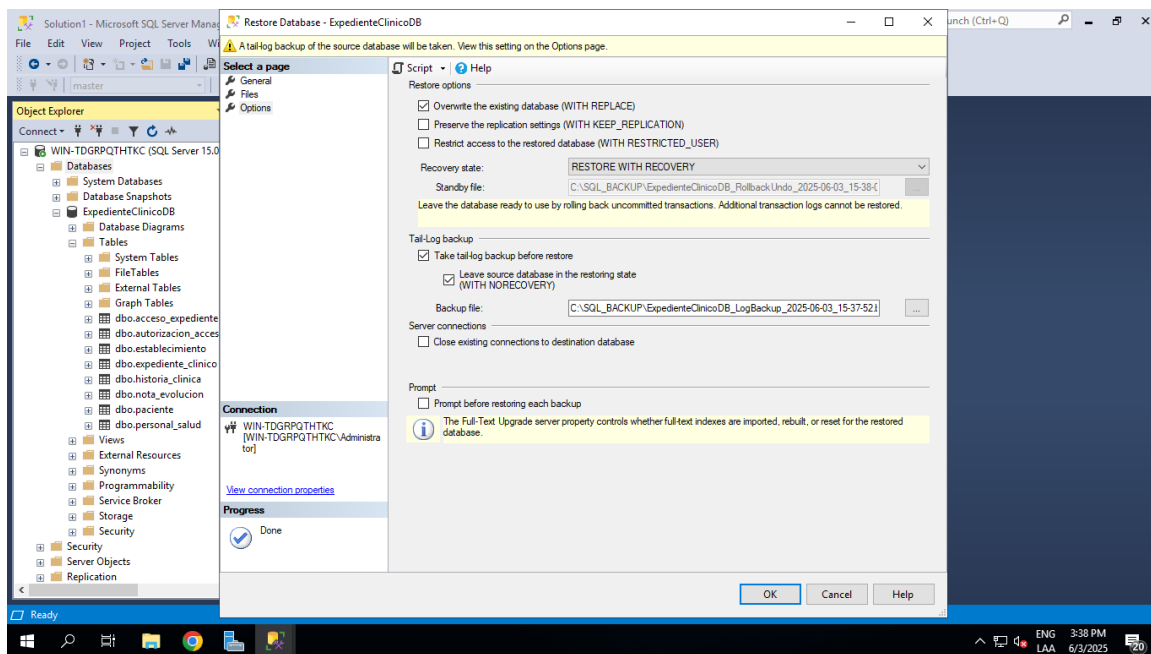
Seleccionamos “Device” y ponemos la ruta donde tenemos guardado nuestro archivo bak:



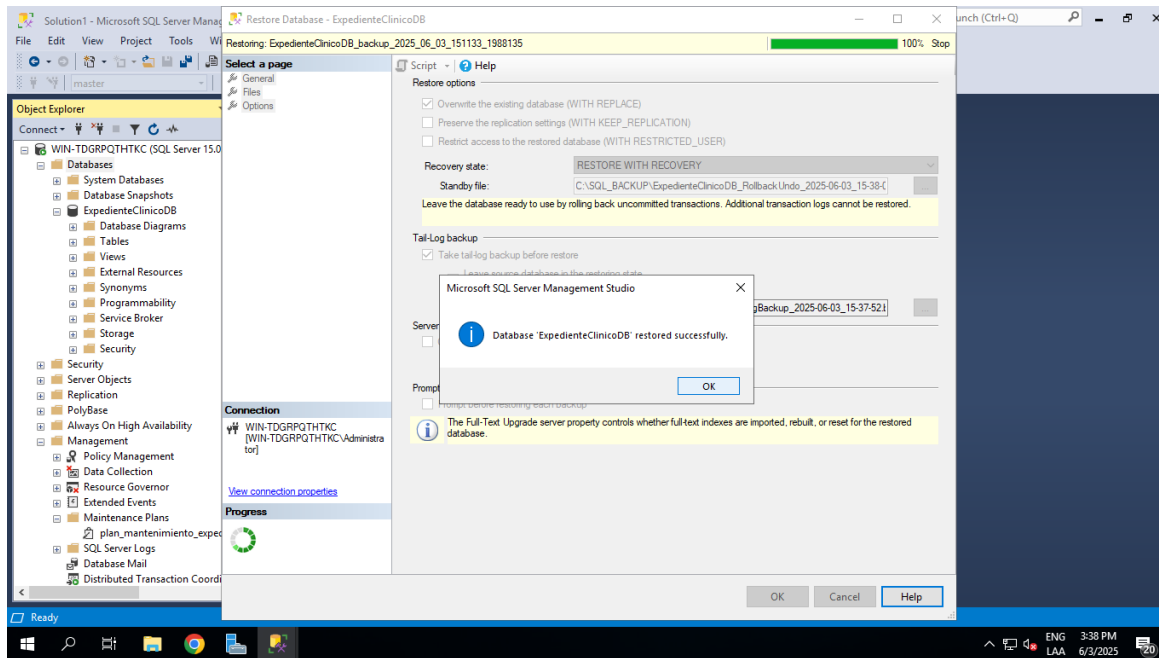
Y marcamos el destino, que en nuestro caso la database ExpedienteClinicoDB:



Y vamos a las opciones para marcar opción “Overwrite the existing database”:



Se ha restaurado la base de datos correctamente:



Conclusión:

Durante el desarrollo de esta práctica se realizaron tareas fundamentales de administración de bases de datos en SQL Server, incluyendo la creación de inicios de sesión, asignación de roles y permisos, programación de respaldos automáticos y planes de mantenimiento, así como la implementación de un esquema de borrado lógico mediante la modificación estructural de las tablas. Finalmente, se probó exitosamente la restauración de la base de datos a un punto anterior, validando así la integridad de los respaldos generados. Estas actividades fortalecen el control, seguridad y disponibilidad de la información en sistemas reales, habilidades clave para cualquier administrador de bases de datos.