



CURSO DE SQL INTERMEDIO

**INTRODUCCIÓN A
SQL SERVER**

Alan Badillo Salas

Ciudad de México. Septiembre 2025

Temario

Módulo I - Fundamentos y modelado de datos con SQL

1. Primera y segunda forma normal
2. Tercera y cuarta forma normal
3. Otras formas normales
4. Manejo de llaves primarias y foráneas

Módulo II - Consultas avanzadas

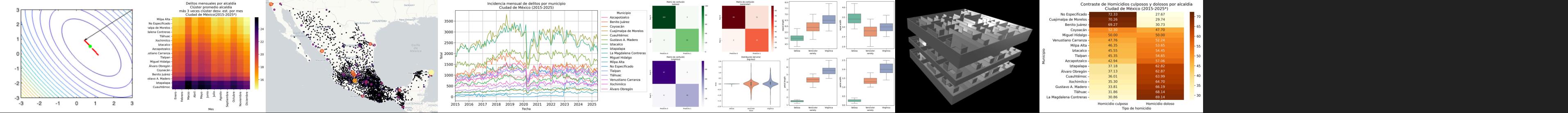
1. Uniones de tipo izquierdo y derecho
2. Uniones centrales y extremas
3. Otras uniones
4. Subconsultas y CTE

Módulo III - Agrupamiento y ventaneo

1. Agrupamiento y funciones de agregación
2. Filtros y subtotales
3. Ventaneado y particiones
4. Ponderaciones, desplazamientos y agregaciones

Módulo IV - Administración y optimización

1. Eficiencia en índices y subconsultas
2. Vistas y transacciones
3. Procedimientos y funciones personalizadas
4. Plan de ejecución



Trayectoria

- **10 años como instructor** de cursos de programación avanzada y ciencia de datos
 - Diplomado de Java (2 módulos) - CIC/IPN
 - Diplomado de IOS SwiftUI (6 módulos) - Grupo Salinas
 - Diplomado de NodeJS y Vue (4 módulos) - CFE
 - **Diplomado de Ciencia de Datos (8 módulos)** - Banjercito
 - Cursos de JavaEE y React - Banxico
 - Cursos de SQL, Python y Excel - Scotiabank
 - **Cursos de Machine Learning** - CENACE
 - **Cursos de Python, IoT, ChatGPT y Machine Learning** - CIC/IPN
 - Cursos de Seguridad Linux y Docker - Telcel
- **Líder de proyecto** y desarrollador de software
 - Plataforma para venta farmacéutica - G. EDOMEX
 - Plataforma de adelantos de nóminas - G. Flores
 - Plataforma de financiamiento celular - C. México
- **Investigación académica**
 - **Redes neuronales estadísticamente informadas** - MCMAI
 - Incidencia delictiva en la CDMX - MCMAI
 - Predicción del salario en adultos con XGBoost y Random Forest - MCMAI
 - **Redes neuronales convolucionales para el reconocimiento de ropa** - MCMAI
 - Modelos jerárquicos bayesianos para el desempeño académico - MCMAI
 - Regresión logística en accidentes de tráfico - MCMAI
 - Evolución diferencial en defunciones - MCMAI
 - Análisis de regionalización en actividades económicas - G. Economía
 - Encuesta de intención de voto para la elección de alcalde - B. Hidalgo
 - Diagramas de transición de interfaces para la generación de código - S. UAM



Alan Badillo Salas

Maestría en Matemáticas Aplicadas UAM

Maestría en Inteligencia Artificial IPN

Lic. en Matemáticas Aplicadas UAM

Proyectos destacados

1. Plataforma de aprendizaje profundo para predicción clínica en COVID-19

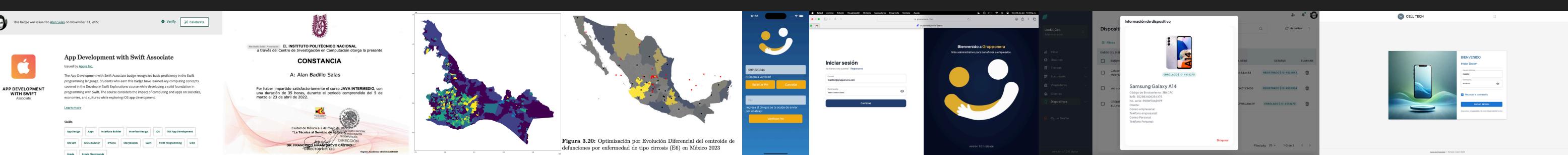
- Predicción de desenlaces clínicos usando redes **neuronales profundas, modelos de ensamble y regresión logística multivariada con efectos mixtos**.
- Implementación cloud con **Matlab, Python, Azure** y consumo en tiempo real vía **React Native**.

2. Programación genética para navegación autónoma de robots virtuales en 3D

- Desarrollo de **algoritmos genéticos para optimizar rutas** en entornos virtuales.
- **Diseño de un lenguaje de programación** basado en C/C++ y Python.

3. Modelo de control dinámico para biorreactores STR/Fed-Batch

- Diseño de sistemas de control dinámico en modelos de cinética enzimática.
- Simulación en LabView con microcontroladores Texas Instruments.



REPOSITORIO

dragonnoma~~d~~/sql-intermedio-scotiabank-2025



Curso de SQL Intermedio para Scotiabank
(Septiembre 2025)

1 Contributor 0 Issues 0 Stars 0 Forks

[GitHub](#)

dragonnoma~~d~~/sql-intermedio-scotiabank-2025: Curso de SQL Intermedio para Scotiabank (Septiembre 2025)

Curso de SQL Intermedio para Scotiabank (Septiembre 2025) - dragonnoma~~d~~/sql-intermedio-scotiabank-2025

[GitHub](#)

<https://github.com/dragonnoma/sql-intermedio-scotiabank-2025>

JERARQUÍA

◆ Administración de objetos y bases de datos en SQL Server

En SQL Server, todo está organizado en **una jerarquía**:

1. Servidor (SQL Server Instance)

- Es la instalación de SQL Server que corre como servicio en el sistema operativo.
- Dentro de una instancia puedes tener varias **bases de datos**.

2. Bases de datos (Databases)

- Cada base contiene su propio **esquema de objetos**.
- Tiene tablas, vistas, procedimientos, funciones, triggers, etc.
- Además, guarda **usuarios propios** (ligados a logins del servidor).

3. Esquemas (Schemas)

- Son “contenedores” dentro de una base.
- Permiten organizar objetos y manejar permisos más fácilmente.
- Ejemplo: `ventas.Clientes` vs `compras.Proveedores` .

4. Objetos (Objects)

- Tablas (`CREATE TABLE`)
- Vistas (`CREATE VIEW`)
- Procedimientos (`CREATE PROCEDURE`)
- Funciones (`CREATE FUNCTION`)
- Triggers (`CREATE TRIGGER`)
- Índices (`CREATE INDEX`)

👉 En resumen: **Servidor → Base de datos → Esquema → Objetos**

◆ Seguridad en SQL Server

SQL Server maneja la seguridad en **dos niveles** principales:

1. Seguridad de servidor

- **Logins:** son cuentas a nivel de instancia (pueden ser de SQL Server o de Windows).
- Roles de servidor: como `sysadmin`, `securityadmin`, `serveradmin`, etc.
- Estos roles controlan cosas globales: creación de bases, configuración, seguridad general.

2. Seguridad de base de datos

- **Usuarios:** cada login puede estar mapeado a un usuario dentro de una base.
- Roles de base de datos: como `db_owner`, `db_datareader`, `db_datawriter`.
- Permisos sobre objetos: `GRANT`, `DENY`, `REVOKE`.

👉 Esto hace que SQL Server sea muy granular:

Puedes dar permisos a un **usuario** solo para `SELECT` en una tabla, o incluso en ciertas columnas.

COMPARACIÓN

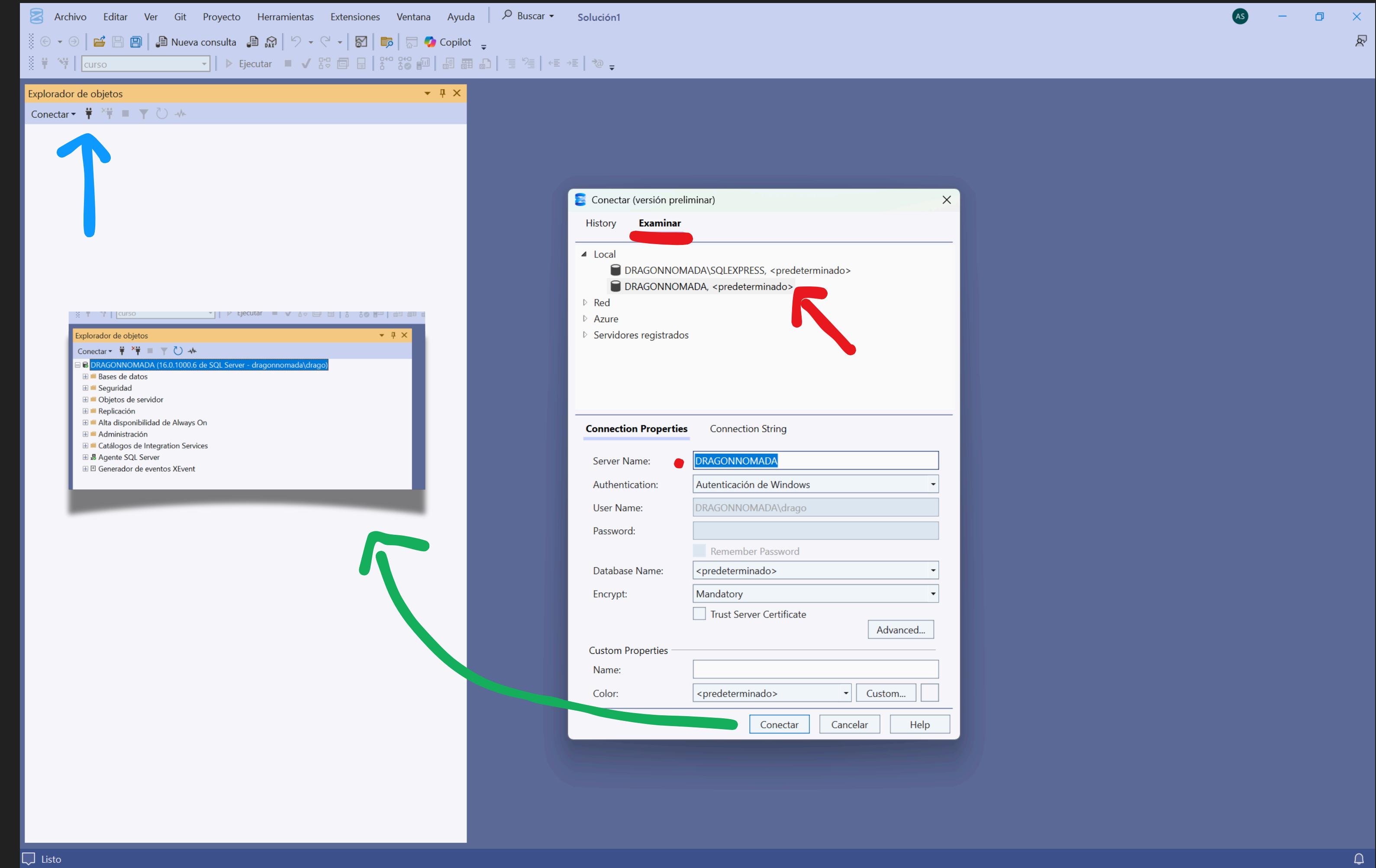
- ◆ Comparación con otros SQL
 - MySQL
 - La seguridad es más simple: se maneja con usuarios globales (no hay la misma separación login/usuario).
 - Los privilegios se asignan sobre bases, tablas o columnas, pero sin roles tan definidos.
 - Ejemplo: `GRANT SELECT ON ventas.clientes TO 'usuario'@'localhost';`
 - PostgreSQL
 - Se parece más a SQL Server porque tiene **roles** que pueden actuar como usuarios o grupos.
 - También permite granularidad: permisos a nivel de esquema, tabla, columna y función.
 - Sin embargo, no tiene la separación estricta login/usuario de SQL Server: un rol puede ser todo a la vez.
 - SQL Server
 - Más ligado al ecosistema Windows: integración con **Active Directory** para autenticación.
 - La separación login (servidor) ↔ usuario (base) es muy clara.
 - Roles predefinidos bien estructurados para administración y seguridad.

❖ Para fijarlo con un ejemplo:

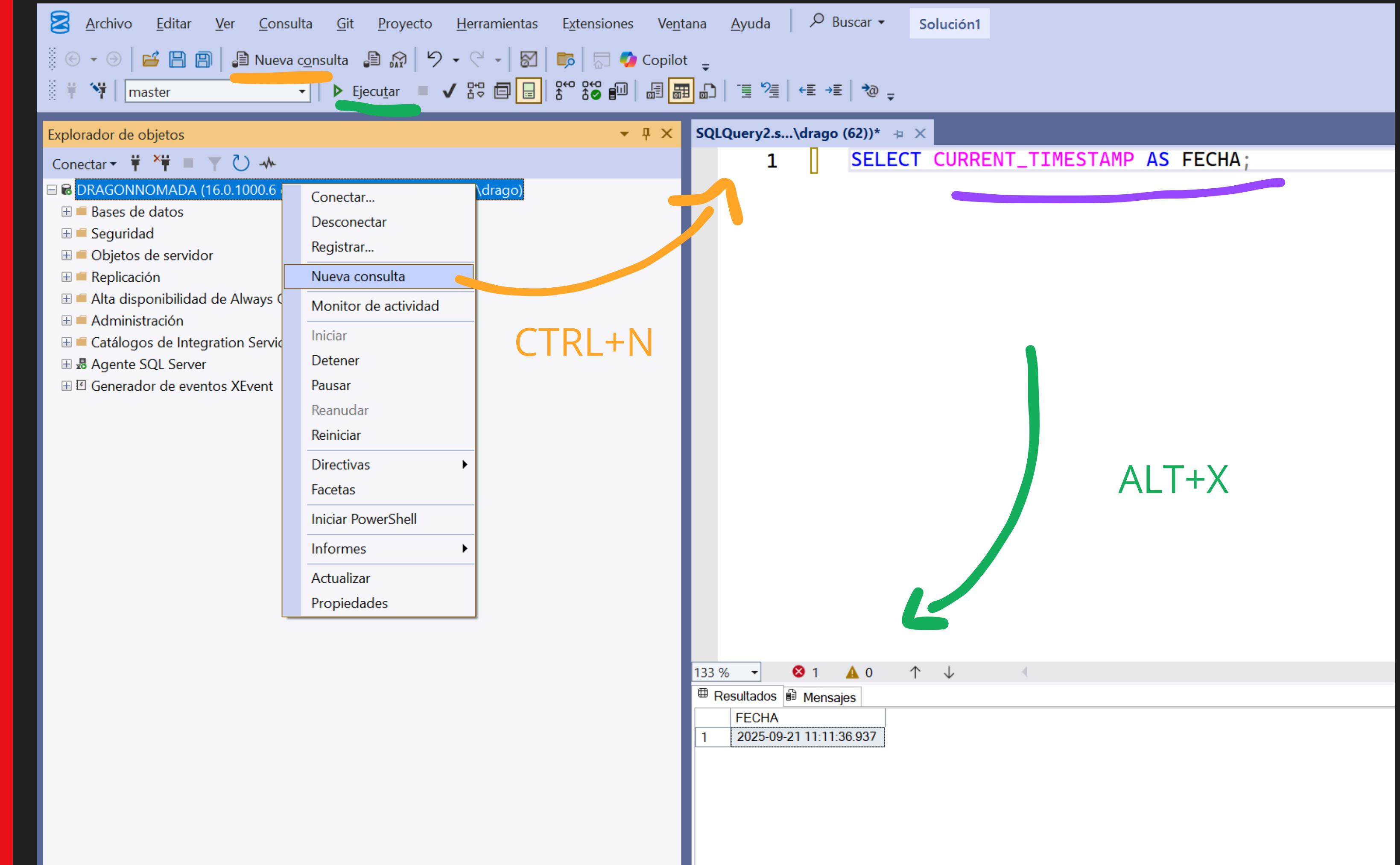
- En **SQL Server** podrías tener:
 - Login: `juan_login` (nivel servidor)
 - Usuario: `juan` (nivel base de datos)
 - Rol: `db_datareader` (solo puede leer tablas)

En **MySQL**, Juan sería solo un usuario con permisos de `SELECT` sobre ciertas tablas.

CONEXIÓN



CONSULTAS



CONSULTAS

◆ 1. Crear una base de datos

sql

 Copiar código

```
CREATE DATABASE MiBaseDeDatos;
```

◆ 2. Crear una tabla

sql

 Copiar código

```
USE MiBaseDeDatos; -- seleccionamos la base
```

```
CREATE TABLE Clientes (
    ClienteID INT PRIMARY KEY IDENTITY(1,1), -- autoincremental
    Nombre NVARCHAR(100) NOT NULL,
    Email NVARCHAR(150) UNIQUE,
    FechaRegistro DATETIME DEFAULT GETDATE()
);
```

◆ 3. Insertar un registro

sql

 Copiar código

```
INSERT INTO Clientes (Nombre, Email)
VALUES ('Juan Pérez', 'juan@example.com');
```

◆ 4. Consultar un registro

sql

 Copiar código

```
SELECT ClienteID, Nombre, Email, FechaRegistro
FROM Clientes
WHERE Nombre = 'Juan Pérez';
```

CONSULTAS

◆ 5. Actualizar un registro

sql

```
UPDATE Clientes  
SET Email = 'juanperez@example.com'  
WHERE ClienteID = 1;
```

Copiar código

◆ 6. Eliminar un registro

sql

```
DELETE FROM Clientes  
WHERE ClienteID = 1;
```

Copiar código

◆ 7. Eliminar una tabla

sql

```
DROP TABLE Clientes;
```

Copiar código

◆ 8. Eliminar una base de datos

sql

```
DROP DATABASE MiBaseDeDatos;
```

Copiar código

❖ Nota importante:

- `DROP TABLE` y `DROP DATABASE` **borran todo el contenido**, no hay "undo".
- En SQL Server es común usar `USE` para moverte entre bases de datos.

TIPOS DE DATOS

Tipos de datos principales en SQL Server

Categoría	Tipo de dato	Descripción / Uso común
Números enteros	INT	Entero de -2,147,483,648 a 2,147,483,647
	BIGINT	Entero grande (hasta ±9 cuatrillones)
	SMALLINT	Entero pequeño (±32 mil)
	TINYINT	Entero muy pequeño (0–255)
Números decimales	DECIMAL(p,s) / NUMERIC(p,s)	Número fijo, con precisión (p) y escala (s). Ej: dinero.
	FLOAT	Número de punto flotante (aprox.)
	REAL	Versión más pequeña de FLOAT
Cadenas de texto	CHAR(n)	Texto de longitud fija (rellena espacios)
	VARCHAR(n)	Texto de longitud variable (hasta 8,000)
	VARCHAR(MAX)	Texto muy grande (hasta 2 GB)
	NCHAR(n)	Como CHAR, pero soporta Unicode
	NVARCHAR(n)	Como VARCHAR, pero Unicode (acentos, emojis)
	NVARCHAR(MAX)	Texto Unicode muy grande

TIPOS DE DATOS

Tipos de datos principales en SQL Server

Categoría	Tipo de dato	Descripción / Uso común
Fecha y hora	DATE	Solo fecha (YYYY-MM-DD)
	TIME	Solo hora
	DATETIME	Fecha y hora (hasta 3 ms de precisión)
	DATETIME2	Fecha y hora más precisa (100 ns)
	SMALLDATETIME	Menos preciso, hasta el año 2079
	DATETIMEOFFSET	Fecha y hora con zona horaria
Booleano	BIT	0, 1 o NULL (muy usado para flags)
Binarios / Otros	BINARY(n)	Datos binarios fijos
	VARBINARY(n)	Datos binarios variables
	VARBINARY(MAX)	Binarios grandes (ej: imágenes)
	UNIQUEIDENTIFIER	GUID (identificador global único)

TIPOS DE DATOS

➤ Reglas rápidas para elegir:

- Si es **texto en español con acentos** → mejor `NVARCHAR`.
- Si es **clave numérica** → `INT` (a menos que necesites valores enormes → `BIGINT`).
- Para **dinero** → `DECIMAL(18,2)` suele ser estándar.
- Para **sí/no** → `BIT`.
- Para **tiempo real** (log de eventos) → `DATETIME2`.

EJEMPLOS

The screenshot shows the GitHub repository page for `microsoft/sql-server-samples`. The main navigation bar includes links for Code, Issues (263), Pull requests (5), Actions, Projects, Models, Security, and Insights. The repository is labeled as Public and has 770 watchers, 9k forks, and 10.7k stars. The sidebar on the left provides cloning options: Local, Codespaces, HTTPS (selected), SSH, and GitHub CLI. A red arrow highlights the 'Code' button in the sidebar, which corresponds to the 'Code' button in the top navigation bar. The repository's activity feed lists 4,010 commits, including merges from `microsoft/dependabot/npm_and_yarn/samples`, updates to dashboards, and schema changes. The 'About' section describes the repository as the official Microsoft GitHub Repository containing code samples for SQL Server, Azure SQL, Azure Synapse, and Azure SQL Edge. It also includes links for Readme, View license, Code of conduct, Security policy, Activity, Custom properties, 10.7k stars, 770 watching, 9k forks, and Report repository. The 'Releases' section shows 19 releases, with the latest being 'SQL Server Machine Learning Se...' from May 1, 2024.

<https://github.com/microsoft/sql-server-samples>

EJEMPLOS

◆ Pubs

Es más antigua (de los 90's) y representa una **editorial de libros**.

Se usaba en cursos de SQL Server 6.5 y 7.0, pero todavía aparece en [sql-server-samples](#).

Tablas principales:

- **Publishers** → editoriales.
- **Authors** → autores de libros.
- **Titles** → títulos publicados (libros).
- **TitleAuthor** → relación muchos-a-muchos entre libros y autores.
- **Sales** → ventas de libros en distintas tiendas.
- **Stores** → tiendas que venden los libros.
- **Employee** → empleados de la editorial.
- **Rroysched** → esquema de regalías para los autores.

❖ Ejemplo de relación:

Un **libro (title)** puede tener varios **autores**, y cada **venta** ocurre en una **tienda** con cierto **precio** y **cantidad**.

👉 Esta base se presta para practicar **consultas financieras y relaciones complejas**, como:

- ¿Cuál es el autor con mayores regalías?
- ¿Qué tienda vendió más libros en un trimestre?
- ¿Qué editorial publica más títulos?

EJEMPLOS

◆ Northwind

Es la más famosa y viene del mundo de una empresa ficticia de importación y exportación de alimentos.

Se diseñó para enseñar **consultas sobre relaciones comerciales** (clientes, pedidos, productos, proveedores).

Tablas principales:

- **Customers** → clientes de la empresa.
- **Suppliers** → proveedores de productos.
- **Employees** → empleados que gestionan pedidos.
- **Products** → productos que se venden.
- **Categories** → categorías de productos (lácteos, bebidas, etc.).
- **Orders** → pedidos de clientes.
- **Order Details** → detalle de cada pedido (qué producto, cantidad, precio).
- **Shippers** → transportistas que envían los pedidos.

↗ Ejemplo de relación:

Un **cliente** hace un **pedido**, que contiene varios **productos**, gestionados por un **empleado** y enviados por un **transportista**.

👉 Esta base es muy útil para practicar **joins** y **consultas de negocio** como:

- ¿Qué cliente compró más en 2024?
- ¿Cuál es el producto más vendido?
- ¿Qué empleado gestionó más pedidos?

EJEMPLOS

◆ Northwind

1. Ventas por cliente

sql

 Copiar código

```
SELECT c.CompanyName, SUM(od.Quantity * od.UnitPrice) AS TotalVentas
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
JOIN [Order Details] od ON o.OrderID = od.OrderID
GROUP BY c.CompanyName
ORDER BY TotalVentas DESC;
```

2. Productos más vendidos por categoría

sql

 Copiar código

```
SELECT cat.CategoryName, p.ProductName, SUM(od.Quantity) AS TotalUnidades
FROM Categories cat
JOIN Products p ON cat.CategoryID = p.CategoryID
JOIN [Order Details] od ON p.ProductID = od.ProductID
GROUP BY cat.CategoryName, p.ProductName
ORDER BY cat.CategoryName, TotalUnidades DESC;
```

3. Empleados con más pedidos gestionados

sql

 Copiar código

```
SELECT e.FirstName + ' ' + e.LastName AS Empleado, COUNT(*) AS Pedidos
FROM Employees e
JOIN Orders o ON e.EmployeeID = o.EmployeeID
GROUP BY e.FirstName, e.LastName
ORDER BY Pedidos DESC;
```