

# Informe Técnico

---

## Normalización a Tercera Forma Normal (3FN) y Análisis de Performance en SQL Server

**Proyecto:** LegacyRetail

**Diplomado:** Gestión de Datos – 2026

**Estudiante:** Karen Suarez

---

### 1. Introducción

Durante varios años, LegacyRetail S.A. almacenó la información de sus ventas en un archivo plano sin ningún tipo de estructura relacional. Este archivo contenía todos los datos mezclados en una sola tabla, lo que dificultaba el análisis, generaba datos repetidos y provocaba errores al momento de realizar consultas.

Al intentar migrar esta información directamente a SQL Server, se evidenciaron múltiples problemas tanto a nivel de diseño como de rendimiento. Por esta razón, el objetivo de este proyecto fue reorganizar la información aplicando los principios de la **Tercera Forma Normal (3FN)** y analizar cómo un buen diseño impacta directamente en el performance de las consultas SQL.

El trabajo se divide en dos partes principales:

- **Misión A:** Normalización del modelo de datos.
  - **Misión B:** Análisis de rendimiento y comparación de consultas.
- 

### 2. Misión A — Normalización del Modelo de Datos

---

#### 2.1 Análisis inicial del problema

El primer problema identificado fue que toda la información se encontraba en una sola estructura, donde se repetían constantemente los datos de clientes, productos y sucursales. Por ejemplo, un mismo cliente aparecía varias veces con el mismo correo electrónico, y una misma sucursal se repetía en múltiples registros de ventas.

Esto generaba:

- Redundancia de información
- Riesgo de inconsistencias
- Dificultad para mantener los datos actualizados

Antes de normalizar, fue necesario entender qué información correspondía realmente a cada entidad del negocio.

---

#### 2.2 Limpieza del entorno de trabajo

Antes de comenzar con el diseño del modelo relacional, se realizó una limpieza completa del entorno de trabajo, eliminando todas las tablas existentes y asegurando que la base de datos **LegacyRetail** estuviera lista para una nueva implementación.

The screenshot shows a Jupyter Notebook workspace titled 'Untitled (Workspace)'. The left sidebar displays a file tree with various files and folders, including 'solution\_schema.sql', 'raw\_sales\_dump.csv', and 'template\_schema.sql'. The main area contains a code cell with the following SQL script:

```

localhost [ LegacyRetail
1 USE LegacyRetail;
2 GO
3
4 DROP TABLE IF EXISTS Ventas;
5 DROP TABLE IF EXISTS Clientes;
6 DROP TABLE IF EXISTS Productos;
7 DROP TABLE IF EXISTS Sucursales;
8 DROP TABLE IF EXISTS RawVentas;
9 GO
10

```

The 'Messages' panel shows the execution log:

- Started executing query at [Line 1](#)
- Commands completed successfully.
- Started executing query at [Line 3](#)
- Commands completed successfully.
- Total execution time: 00:00:00.008

The bottom status bar indicates the environment is 'localhost' and the database is 'LegacyRetail'.

Este paso fue clave para evitar errores causados por ejecuciones previas y para poder validar correctamente cada etapa del proceso.

---

### 2.3 Creación de la tabla de staging (RawVentas)

Se creó la tabla **RawVentas** como una tabla de staging, cuyo objetivo fue almacenar los datos originales del archivo CSV sin aplicar transformaciones inmediatas.

```

CREATE TABLE RawVentas (
    Producto VARCHAR(100),
    Categoria VARCHAR(50),
    Precio DECIMAL(10,2),
    Cantidad INT,
    Sucursal VARCHAR(100),
    Ciudad_Sucursal VARCHAR(100),
    Fecha_Venta DATE
);
GO
  
```

Build with Agent  
AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

Open in New Tab

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL QUERY RESULTS

Messages

3:45:25 PM Started executing query at Line 1  
Commands completed successfully.  
3:45:25 PM Started executing query at Line 3  
Commands completed successfully.  
3:45:25 PM Started executing query at Line 10  
Commands completed successfully.  
Total execution time: 00:00:00.056

solution\_schema.sql

Dockerfile

poetry.lock

pyproject.toml

README.md

gitignore

docker-compose.yml

19°C Mayorm, nublado

Buscar

15:45 20/01/2026

Durante esta etapa se identificaron varios problemas en los datos, como espacios en blanco, diferencias en el formato de texto y repetición de información, lo cual confirmó la necesidad de normalizar el modelo.

## 2.4 Diseño y creación de las tablas normalizadas

Luego del análisis de los datos crudos, se definieron las entidades principales del sistema y se crearon las siguientes tablas:

- Clientes
- Productos
- Sucursales
- Ventas

Cada tabla fue diseñada con una clave primaria y con atributos que dependen únicamente de dicha clave, cumpliendo con los principios de la Tercera Forma Normal.

The screenshot shows a Jupyter Notebook workspace with several open files:

- `docker-compose.yml`
- `solution_schema.sql`
- `raw_sales_dump.csv`
- `template_`
- `diplomado > reto_5_6_dgd_2026-main > 02_sql > 1_ddl_diseño > solution_schema.sql`

The `solution_schema.sql` file contains the following SQL code:

```

198 SELECT COUNT(*) AS RawVentas FROM RawVentas; -- 200
199 SELECT COUNT(*) AS Clientes FROM Clientes; -- 21
200 SELECT COUNT(*) AS Productos FROM Productos; -- 12
201 SELECT COUNT(*) AS Sucursales FROM Sucursales; -- 5
202
203
204

```

The `template_` file shows a table structure:

name	U
Cuentas	U
Datos	U
Productos	U
Sucursales	U
Ventas	U

The `solution_schema.sql` file also contains data for the `Ventas` table:

RawVentas	U
1	200

Below the table structure, there is a section titled "Build with Agent" with the following text:

AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

Agent: Auto

La tabla **Ventas** se definió como la tabla central del modelo, encargada de relacionar a los clientes, productos y sucursales.

## 2.5 Carga de datos y dificultades encontradas

Durante la carga de datos hacia las tablas normalizadas se presentaron algunos retos, especialmente relacionados con la identificación única de los clientes y la eliminación de duplicados.

Para resolver esto, se utilizó el correo electrónico como identificador único de cliente, permitiendo consolidar múltiples registros repetidos en una sola fila dentro de la tabla **Clientes**.

## 2.6 Validación de registros cargados

Una vez completada la carga, se realizó un conteo de registros para validar que la información se hubiera distribuido correctamente entre las tablas.

Resultados obtenidos:

- RawVentas: 200 registros
- Clientes: 21 registros
- Productos: 12 registros
- Sucursales: 5 registros
- Ventas: 200 registros

The screenshot shows a Jupyter Notebook workspace titled 'Untitled (Workspace)'. The code editor contains several SQL queries for validating the 'Ventas' table:

```

195 -- 8. VALIDACION FINAL
196
197
198 SELECT COUNT(*) AS RawVentas FROM RawVentas; -- 200
199 SELECT COUNT(*) AS Clientes FROM Clientes; -- 21
200 SELECT COUNT(*) AS Productos FROM Productos; -- 12
201 SELECT COUNT(*) AS Sucursales FROM Sucursales; -- 5
202 SELECT COUNT(*) AS Ventas FROM Ventas; -- 200
203 GO
204

```

The 'OUTPUT' tab shows the results of these queries, indicating successful execution and completion times. A message at the bottom states 'Total execution time: 00:00:00.744'.

Estos resultados confirmaron que el proceso de normalización fue exitoso y que la redundancia fue eliminada.

## 2.7 Revisión de la estructura de la tabla Ventas

Se revisó la estructura de la tabla **Ventas** para verificar que los tipos de datos, la clave primaria y las columnas de referencia estuvieran correctamente definidas.

The screenshot shows the 'solution\_schema.sql' file in the Jupyter Notebook workspace. It displays the structure of the 'Ventas' table, including its columns, data types, and constraints:

Name	Type	Owner	Created_datetime
Ventas	user table	dbo	2026-01-20 20:53:59.437

Column_name	Type	Computed	Length	Prec	Scale	Nullable
VentaID	int	no	4	10	0	no
Fecha	date	no	3	10	0	yes
Cantidad	int	no	4	10	0	yes
Precio_Unitario	decimal	no	9	12	2	yes
ClienteID	int	no	4	10	0	yes
ProductoID	int	no	4	10	0	yes
SucursalID	int	no	4	10	0	yes

Identity	Seed	Increment	Not For Replication
VentaID	1	1	0

RowGuidCol

No rowguidcol column defined.
-------------------------------

Data_located_on_filegroup

PRIMARY
---------

index_name	index_description	index_keys
PK_Ventas_5B41514CE51EEA1	clustered, unique, primary key located on PRIMARY	VentaID

constraint_type	constraint_name	delete_action	update_action	status_enabled
FOREIGN KEY	FK_Ventas_Clientel_467D75B8	No Action	No Action	Enabled

## 2.8 Validación de integridad referencial

Finalmente, se validó que las claves foráneas estuvieran correctamente configuradas, asegurando la relación entre las tablas y evitando la inserción de datos inconsistentes.

```

USE LegacyRetail;
GO
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO

```

Messages

4:52:49 PM Started executing query at Line 1  
Commands completed successfully.  
4:52:49 PM Started executing query at Line 3  
Commands completed successfully.  
Total execution time: 00:00:00.011

## 3. Misión B — Análisis de Performance

### 3.1 Preparación del entorno de pruebas

Para analizar el rendimiento de las consultas, se activaron las opciones **STATISTICS IO** y **STATISTICS TIME**, lo que permitió observar el consumo de recursos de cada consulta ejecutada.

Cliente	Producto
1 ANA MARIA	Disco SSD 1TB
2 ANDRES LOPEZ	Disco SSD 1TB
3 Karol G	Disco SSD 1TB
4 Claudia Bahamon	Disco SSD 1TB
5 Carlos Ruiz	Disco SSD 1TB
6 egan bernal	Disco SSD 1TB
7 falcao garcia	Disco SSD 1TB
8 feid	Disco SSD 1TB
9 james rodriguez	Disco SSD 1TB
10 Jorge Rodriguez	Disco SSD 1TB
11 J Balvin	Disco SSD 1TB
12 juan perez	Disco SSD 1TB
13 LAURA ACU+æA	Disco SSD 1TB
14 Luisa Fernanda	Disco SSD 1TB

### 3.2 Consulta con CROSS JOIN y problemas detectados

Se ejecutó una consulta utilizando **CROSS JOIN** entre las tablas **Clientes** y **Productos**. Esta consulta generó un producto cartesiano, combinando todos los registros de ambas tablas.

```

File Edit Selection View Go Run Terminal Help < > Untitled (Workspace)
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL QUERY RESULTS
Results (1) Messages
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 0 ms.

SQL Server Execution Times:
CPU time = XXXX ms
elapsed time = XXXX ms

```

AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

Build with Agent

Describe what to build next

Agent Auto

Ln 19, Col 1 Spaces: 4 UTF-8 CRLF { } SQL SQLCMD: Off MSSQL 00:00:00.088 localhost LegacyRetail 16:54 20/01/2026

### 3.3 Resultados y métricas del CROSS JOIN

El resultado de esta consulta generó una gran cantidad de filas sin relación real de negocio, lo que se reflejó directamente en un aumento del consumo de recursos.

```

File Edit Selection View Go Run Terminal Help < > Untitled (Workspace)
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL QUERY RESULTS
Results (2) Messages
107 falcao garcia Memoria RAM 16...
108 feid Memoria RAM 16...
109 james rodriguez Memoria RAM 16...
110 Jorge Rodriguez Memoria RAM 16...
111 J Balvin Memoria RAM 16...

```

Cliente	Producto	Fecha	Cantidad
1 Luisa Fernanda	Laptop Gamer X	2024-02-27	2
2 Luisa Fernanda	Tarjeta de Video	2024-05-31	4
3 Maria Gomez	Laptop Gamer X	2024-02-29	5
4 J Balvin	Laptop Gamer X	2024-06-15	5
5 SHAKIRA MEBARAK	Monitor 24	2024-01-02	2
6 SHAKIRA MEBARAK	Webcam HD	2024-02-25	3
7 Luisa Fernanda	Mouse Wireless	2024-04-01	3
8 J Balvin	Headset Pro	2024-05-17	1

AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

Build with Agent

Describe what to build next

Agent Auto

Ln 26, Col 1 Spaces: 4 UTF-8 CRLF { } SQL SQLCMD: Off MSSQL 200 rows affected 00:00:00.399 localhost LegacyRetail 16:59 20/01/2026

Las métricas obtenidas mostraron un alto número de lecturas lógicas, evidenciando que este tipo de consulta no es adecuada para este escenario.

```

solution_tuning.sql
diplomado > reto_5_6_dgd_2026-main > 02_sql > 2_performance_lab > solution_tuning.sql
31 GO
32
33 -- Conteo real
34 SELECT COUNT(*) AS Total_Inner
35 FROM Ventas;
36 GO
37

```

	Cliente	Producto	Fecha	Cantidad
193	feid	Webcam HD	2024-03-16	3
194	SHAKIRA MEBARAK	Tecaldo Mecanico	2024-05-18	3
195	Nairo Quintana	Disco SSD 1TB	2024-03-06	4
196	LAURA ACU+æA	Impresora Laser	2024-04-29	1
197	ANA MARIA	Monitor 24	2024-01-24	4
198	juan perez	Headset Pro	2024-04-26	3
199	Miguel Angel	Memoria RAM 16GB	2024-01-28	2
200	egan bernal	Laptop Gamer X	2024-06-05	2
	Total_Cross			
1	240			
	Total_Inner			
1	200			

### 3.4 Consulta optimizada con INNER JOIN

Posteriormente, se ejecutó una consulta utilizando **INNER JOIN**, relacionando correctamente las tablas a través de la tabla **Ventas**.

```

solution_tuning.sql
diplomado > reto_5_6_dgd_2026-main > 02_sql > 2_performance_lab > solution_tuning.sql
localhost | LegacyRetail
1 USE LegacyRetail;
2 GO
3
4 SET STATISTICS IO ON;
5 SET STATISTICS TIME ON;
6 GO
7
8 SELECT
9     c.Nombre AS Cliente,
10    p.Nombre AS Producto
11 FROM Clientes c
12     INNER JOIN Productos p
13         ON c.ClienteID = p.ClienteID
14

```

SQL Server Execution Times:  
CPU time = 0 ms, elapsed time = 2 ms.  
Started executing query at Line 14  
SQL Server parse and compile time:  
CPU time = 0 ms, elapsed time = 0 ms.  
(200 rows affected)  
Table 'workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead  
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead  
Table 'Ventas'. Scan count 1, logical reads 4, physical reads 0, page server reads 0, read-ahead  
Table 'Productos'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead  
Table 'Clientes'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead

SQL Server Execution Times:  
CPU time = 20 ms, elapsed time = 20 ms.  
Started executing query at Line 26  
SQL Server parse and compile time:  
CPU time = 0 ms, elapsed time = 0 ms.  
(1 row affected)  
Table 'Productos'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead  
Table 'Clientes'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead

Esta consulta devolvió únicamente los registros con relación real, reduciendo significativamente el consumo de recursos.

---

## 4. Comparación de Resultados

Aspecto	CROSS JOIN	INNER JOIN
Relación de negocio	No existe	Correcta
Filas generadas	Excesivas	Necesarias
Lecturas lógicas	Altas	Reducidas
Rendimiento	Bajo	Optimizado

---

## 5. Conclusiones y Aprendizajes

- La normalización permitió organizar la información de manera clara y estructurada.
- Separar las entidades redujo la redundancia y mejoró la calidad de los datos.
- Un mal uso de **CROSS JOIN** puede generar graves problemas de rendimiento.
- El uso adecuado de **INNER JOIN** mejora considerablemente la eficiencia de las consultas.
- El análisis de métricas es fundamental para justificar decisiones técnicas.

Este proyecto permitió comprender la importancia del diseño de bases de datos y cómo este impacta directamente en el rendimiento y la estabilidad de un sistema.

### Diagrama Entidad–Relación (DER)

El siguiente diagrama representa el modelo relacional normalizado a Tercera Forma Normal (3FN), donde las entidades Clientes, Productos y Sucursales se relacionan a través de la tabla Ventas, la cual actúa como tabla central del modelo.

