BASES DE DATOS II

PREICA2501B010095

S20 - EVIDENCIA DE APRENDIZAJE 2

CREACIÓN DE UNA BASE DE DATOS DE STAGING

REALIZADO POR:

Grupo:

BASEDEDATOSIIPRE 5

JEREMY IVAN PEDRAZA HERNANDEZ

PRESENTADO A:

INSTRUCTOR

VICTOR HUGO MERCADO RAMOS

INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

2025

Contenido

[Planteamiento del Problema 3](#_Toc193057782)

[Introducción 4](#_Toc193057783)

[Objetivos 4](#_Toc193057784)

[Análisis de la base de datos (Problema) 5](#_Toc193057785)

[Creación de conexiones: Origen y destino 6](#_Toc193057786)

[Creación de base de datos *Stagging* 7](#_Toc193057787)

[Comandos SQL 7](#_Toc193057788)

[Tabla: Categoria\_producto 8](#_Toc193057789)

[Tabla: cliente 9](#_Toc193057790)

[Tabla: detalle\_pedido 10](#_Toc193057791)

[Tabla: Empleado 11](#_Toc193057792)

[Tabla: Oficina 12](#_Toc193057793)

[Tabla: Pago 13](#_Toc193057794)

[Tabla: Pedido 14](#_Toc193057795)

[Table: Producto 15](#_Toc193057796)

[Código SQL para limpieza de Tablas *Stagging* 16](#_Toc193057797)

[Optimización de tablas 17](#_Toc193057798)

[Resultados 18](#_Toc193057799)

[Conclusiones 24](#_Toc193057800)

[Bibliografía 25](#_Toc193057801)

# Planteamiento del Problema

1. Análisis de los datos almacenados en la base de datos Jardinería:
   * Revisar los datos almacenados en el Jardinería para identificar cuáles son relevantes y cuáles se deben trasladar a la base de datos *Staging.*
2. Construcción de la base de Datos *Staging:*
   * Diseñar la estructura tablas que estarán en la base de datos *Staging*.
   * Construir las consultas que permitan traer los registros de Jardinería a la Base de Datos *Staging*.
   * Ejecutar las consultas y validar que los datos queden almacenados correctamente en la Base de datos *Staging*.
   * Construir el BK de ambas bases de datos.
3. Documentación y presentación:

Redactar un documento escrito que cumpla con normas APA y contenga:

* + Introducción
  + Objetivos
  + Planteamiento del problema
  + Análisis del problema
  + Propuesta de la solución con:
    - Correcciones a la entrega 1.
    - Descripción del análisis realizado a los datos *Jardinería* y cómo estos se trasladaron a la base de datos *Staging.*
    - Anexos: debe agregar los BK de las dos bases de datos y el documento del *script* de las consultas para crear la base de datos *Staging.*
    - Bibliografía.

# Introducción

En este documento, se describe el proceso de extracción, transformación y carga (ETL) de los datos desde la base de datos **Jardinería** hacia la base de datos **Staging**. La base de datos Staging servirá como un entorno intermedio donde se almacenarán los datos antes de ser procesados y enviados a sistemas de análisis o almacenamiento definitivo.

Este proceso garantiza la integridad y calidad de los datos antes de su uso en reportes o análisis empresariales. Se incluyen las estructuras de las tablas en **Staging**, las consultas necesarias para trasladar los datos y los mecanismos de validación para asegurar la correcta migración

# Objetivos

1. **Diseñar la estructura de las tablas en la base de datos Staging**, asegurando que reflejen la información necesaria para la integración de datos.
2. **Extraer los registros desde la base de datos Jardinería y cargarlos en Staging**, manteniendo la integridad y consistencia de la información.
3. **Validar la correcta inserción de los datos** en Staging mediante consultas de verificación.
4. **Garantizar la trazabilidad de los datos** mediante campos adicionales como fecha de carga y estado.
5. **Realizar un respaldo (backup) de ambas bases de datos** para asegurar la disponibilidad de la información en caso de errores o futuras auditorías.

# Análisis de la base de datos (Problema)

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Ilustración Modelo Transaccional Jardinería

1. Se ha cargado la base de datos en la instancia SQL y se insertan los datos, así mismo el código cuenta con relaciones en llaves foranes preestablecidas. El producto de esto se muestra en la presente imagen (Ilustración 1)

## Creación de conexiones: Origen y destino

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración Conexión origen y stagging

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración Conexión jardinería y stagging

Se crean las dos conexiones necesarias para poder realizar el traslado de la información de las tablas.

## Creación de base de datos *Stagging*

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración Creación de Base de datos Stagging

Se crea la base de datos **DBO\_Stagging\_Jardineria** para recepcionar los datos de la base de datos de jardinería, allí se creará cada consulta que permitirá traer los registros de cada tabla.

## Comandos SQL

1. A partir de cada tabla entraremos al SMSS y procederemos a revisar los datos almacenados en el Jardinería para identificar cuáles son relevantes y cuáles se deben trasladar a la base de datos *Stagging.*

### Tabla: Categoria\_producto

En el caso de la tabla ***categoría\_producto*** nos traemos todo menos ***descripción\_html*** e ***imagen*** ya que cuentan con campos nulos.

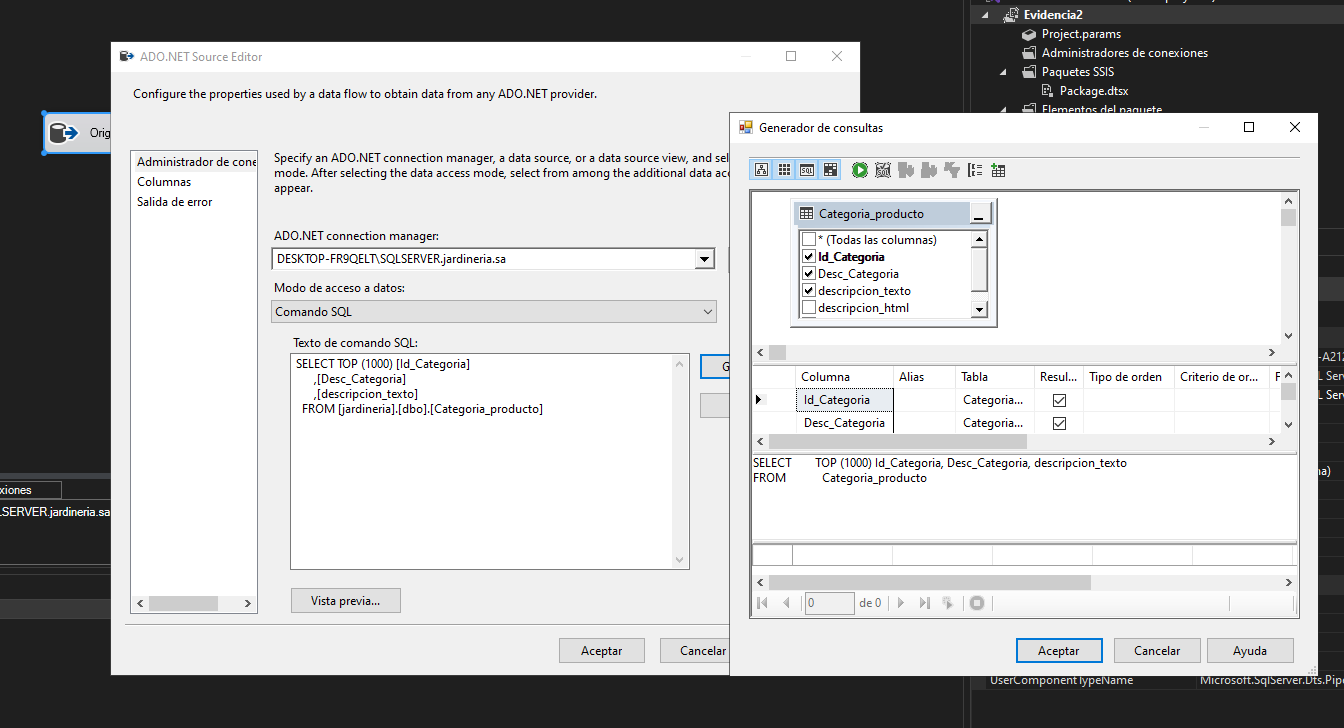


Ilustración Categoria\_Producto Comando SQL

Código SQL de tabla stagging

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

|  |  |
| --- | --- |
| 1  2  3  4  5 | **CREATE** **TABLE** "staging\_categoria\_producto" (  "Id\_Categoria" INT,  "Desc\_Categoria" VARCHAR(50),  "descripcion\_texto" TEXT  ) |

Ilustración SQL creación tabla

## Tabla: cliente

En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas

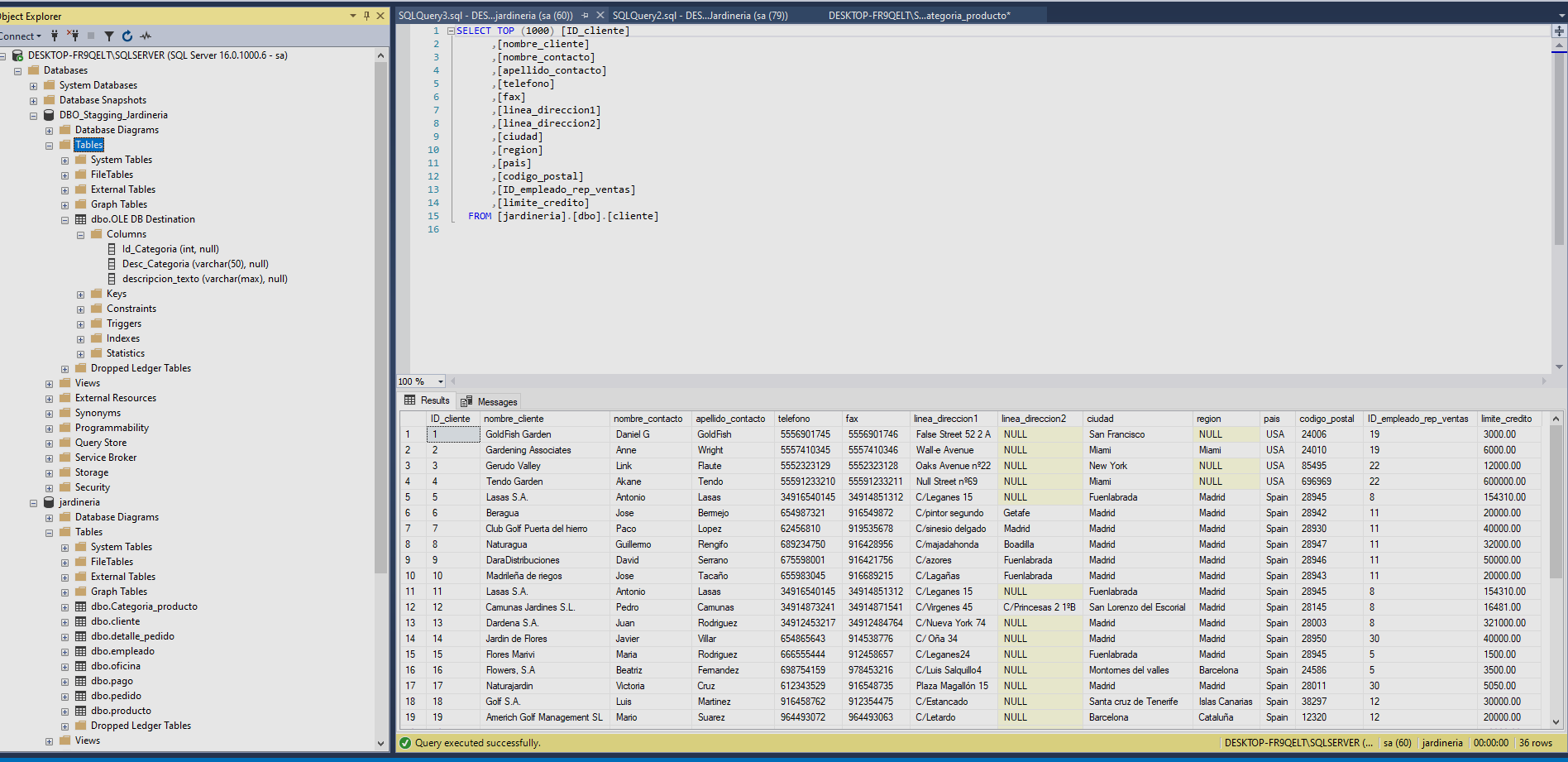


Ilustración Datos tabla cliente

Código SQL de tabla stagging

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **CREATE** **TABLE** "DB Stagging Cliente" (  "ID\_cliente" INT,  "nombre\_cliente" VARCHAR(50),  "nombre\_contacto" VARCHAR(30),  "apellido\_contacto" VARCHAR(30),  "telefono" VARCHAR(15),  "fax" VARCHAR(15),  "linea\_direccion1" VARCHAR(50),  "linea\_direccion2" VARCHAR(50),  "ciudad" VARCHAR(50),  "region" VARCHAR(50),  "pais" VARCHAR(50),  "codigo\_postal" VARCHAR(10),  "ID\_empleado\_rep\_ventas" INT,  "limite\_credito" NUMERIC (15,2)  ) | |

Ilustración Codigo SQL Stagging

## Tabla: detalle\_pedido

En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas

Interfaz de usuario gráfica, Tabla

El contenido generado por IA puede ser incorrecto.

Ilustración Datos tabla detalle\_pedido

Código SQL de tabla stagging

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7 | **CREATE** **TABLE** [DB Stagging Detalle Pedido] (  [ID\_pedido] int,  [ID\_producto] varchar(15),  [cantidad] int,  [precio\_unidad] numeric(15,2),  [numero\_linea] smallint  ) | |

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración Codigo SQL Stagging

## Tabla: Empleado

En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Ilustración Tabla Empleado

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11 | **CREATE** **TABLE** [DB Stagging Empleado] (  [ID\_empleado] int,  [nombre] varchar(50),  [apellido1] varchar(50),  [apellido2] varchar(50),  [extension] varchar(10),  [email] varchar(100),  [ID\_oficina] int,  [ID\_jefe] int,  [puesto] varchar(50)  ) | |

Ilustración Codigo SQL Stagging

## Tabla: Oficina

En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración Tabla Oficina

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11 | **CREATE** **TABLE** [DB Staggin Oficina] (  [ID\_oficina] int,  [Descripcion] varchar(10),  [ciudad] varchar(30),  [pais] varchar(50),  [region] varchar(50),  [codigo\_postal] varchar(10),  [telefono] varchar(20),  [linea\_direccion1] varchar(50),  [linea\_direccion2] varchar(50)  ) | |

Ilustración Codigo SQL Stagging

## Tabla: Pago

En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración Tabla Pago

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7 | **CREATE** **TABLE** [DB Stagging Pago] (  [ID\_cliente] int,  [forma\_pago] varchar(40),  [id\_transaccion] varchar(50),  [fecha\_pago] nvarchar(10),  [total] numeric(15,2)  ) | |

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración Codigo SQL Staggin

## Tabla: Pedido

En el caso de la tabla pedido omitimos únicamente la columna comentario ya que no trae información importante para la tabla stagging.

Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración Tabla Pedido

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8 | **CREATE** **TABLE** [DB Stagging Pedido] (  [ID\_pedido] int,  [fecha\_pedido] nvarchar(10),  [fecha\_esperada] nvarchar(10),  [fecha\_entrega] nvarchar(10),  [estado] varchar(15),  [ID\_cliente] int  ) | |

Ilustración Codigo SQL Stagging

## Table: Producto

Se omite la columna descripción y se importan las demás columnas

Interfaz de usuario gráfica, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

Ilustración Tabla SQL Pedido

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10 | **CREATE** **TABLE** [DB Stagging Producto] (  [ID\_producto] varchar(15),  [nombre] varchar(70),  [Categoria] int,  [dimensiones] varchar(25),  [proveedor] varchar(50),  [cantidad\_en\_stock] smallint,  [precio\_venta] numeric(15,2),  [precio\_proveedor] numeric(15,2)  ) | |

Ilustración Codigo SQL Stagging

## Código SQL para limpieza de Tablas *Stagging*

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7 | **TRUNCATE** **TABLE** [dbo].[DB Stagging Cliente];  **TRUNCATE** **TABLE** [dbo].[DB Stagging Detalle Pedido];  **TRUNCATE** **TABLE** [dbo].[DB Stagging Empleado];  **TRUNCATE** **TABLE** [dbo].[DB Staggin Oficina];  **TRUNCATE** **TABLE** [dbo].[DB Stagging Pago];  **TRUNCATE** **TABLE** [dbo].[DB Stagging Pedido];  **TRUNCATE** **TABLE** [dbo].[DB Stagging Producto]; | |

## Optimización de tablas

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Obtenemos el nombre de todas las tablas para mejorarlas como un proceso ETL usando SISS INCLUIMOS campos como **"Fecha de carga"** y **"Estado de carga"** para rastrear cuándo y cómo se cargaron los datos en la base de datos a cada tabla.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración Tarea complementaria manual

Codigo SQL para complementar las tablas

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7 | **ALTER** **TABLE** [dbo].[DB Stagging Cliente] **ADD** fecha\_carga DATETIME **DEFAULT** GETDATE(), estado\_carga VARCHAR(20);  **ALTER** **TABLE** [dbo].[DB Stagging Detalle Pedido] **ADD** fecha\_carga DATETIME **DEFAULT** GETDATE(), estado\_carga VARCHAR(20);  **ALTER** **TABLE** [dbo].[DB Stagging Empleado] **ADD** fecha\_carga DATETIME **DEFAULT** GETDATE(), estado\_carga VARCHAR(20);  **ALTER** **TABLE** [dbo].[DB Staggin Oficina] **ADD** fecha\_carga DATETIME **DEFAULT** GETDATE(), estado\_carga VARCHAR(20);  **ALTER** **TABLE** [dbo].[DB Stagging Pago] **ADD** fecha\_carga DATETIME **DEFAULT** GETDATE(), estado\_carga VARCHAR(20);  **ALTER** **TABLE** [dbo].[DB Stagging Pedido] **ADD** fecha\_carga DATETIME **DEFAULT** GETDATE(), estado\_carga VARCHAR(20);  **ALTER** **TABLE** [dbo].[DB Stagging Producto] **ADD** fecha\_carga DATETIME **DEFAULT** GETDATE(), estado\_carga VARCHAR(20); | |

## Resultados

Estado del flujo de tareas

Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Finalmente validamos los datos  
Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Complementamos los datos

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

De esta manera la próxima vez que se haga un cargue se ejecutará los triggers adecaudaos para registrar los datos en el momento indicado  
Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Lo único que queda pendiente es añadir un trigger para ejecutar los estados de cargue y registrar el timestamp  
Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Codigo SQL Triggers

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97 | *-- TRIGGER para DB Stagging Cliente*  **CREATE** **TRIGGER** trg\_Stagging\_Cliente  **ON** [dbo].[DB Stagging Cliente]  **AFTER** **INSERT**  **AS**  **BEGIN**  **SET** NOCOUNT **ON**;  **UPDATE** [dbo].[DB Stagging Cliente]  **SET** fecha\_carga = GETDATE(),  estado\_carga = 'Cargado'  **WHERE** ID\_cliente **IN** (**SELECT** ID\_cliente **FROM** inserted);  **END**;  **GO**  *-- TRIGGER para DB Stagging Detalle Pedido*  **CREATE** **TRIGGER** trg\_Stagging\_DetallePedido  **ON** [dbo].[DB Stagging Detalle Pedido]  **AFTER** **INSERT**  **AS**  **BEGIN**  **SET** NOCOUNT **ON**;  **UPDATE** [dbo].[DB Stagging Detalle Pedido]  **SET** fecha\_carga = GETDATE(),  estado\_carga = 'Cargado'  **WHERE** ID\_pedido **IN** (**SELECT** ID\_pedido **FROM** inserted);  **END**;  **GO**  *-- TRIGGER para DB Stagging Empleado*  **CREATE** **TRIGGER** trg\_Stagging\_Empleado  **ON** [dbo].[DB Stagging Empleado]  **AFTER** **INSERT**  **AS**  **BEGIN**  **SET** NOCOUNT **ON**;  **UPDATE** [dbo].[DB Stagging Empleado]  **SET** fecha\_carga = GETDATE(),  estado\_carga = 'Cargado'  **WHERE** ID\_empleado **IN** (**SELECT** ID\_empleado **FROM** inserted);  **END**;  **GO**  *-- TRIGGER para DB Stagging Oficina*  **CREATE** **TRIGGER** trg\_Stagging\_Oficina  **ON** [dbo].[DB Staggin Oficina]  **AFTER** **INSERT**  **AS**  **BEGIN**  **SET** NOCOUNT **ON**;  **UPDATE** [dbo].[DB Staggin Oficina]  **SET** fecha\_carga = GETDATE(),  estado\_carga = 'Cargado'  **WHERE** ID\_oficina **IN** (**SELECT** ID\_oficina **FROM** inserted);  **END**;  **GO**  *-- TRIGGER para DB Stagging Pago*  **CREATE** **TRIGGER** trg\_Stagging\_Pago  **ON** [dbo].[DB Stagging Pago]  **AFTER** **INSERT**  **AS**  **BEGIN**  **SET** NOCOUNT **ON**;  **UPDATE** [dbo].[DB Stagging Pago]  **SET** fecha\_carga = GETDATE(),  estado\_carga = 'Cargado'  **WHERE** ID\_cliente **IN** (**SELECT** ID\_cliente **FROM** inserted);  **END**;  **GO**  *-- TRIGGER para DB Stagging Pedido*  **CREATE** **TRIGGER** trg\_Stagging\_Pedido  **ON** [dbo].[DB Stagging Pedido]  **AFTER** **INSERT**  **AS**  **BEGIN**  **SET** NOCOUNT **ON**;  **UPDATE** [dbo].[DB Stagging Pedido]  **SET** fecha\_carga = GETDATE(),  estado\_carga = 'Cargado'  **WHERE** ID\_pedido **IN** (**SELECT** ID\_pedido **FROM** inserted);  **END**;  **GO**  *-- TRIGGER para DB Stagging Producto*  **CREATE** **TRIGGER** trg\_Stagging\_Producto  **ON** [dbo].[DB Stagging Producto]  **AFTER** **INSERT**  **AS**  **BEGIN**  **SET** NOCOUNT **ON**;  **UPDATE** [dbo].[DB Stagging Producto]  **SET** fecha\_carga = GETDATE(),  estado\_carga = 'Cargado'  **WHERE** ID\_producto **IN** (**SELECT** ID\_producto **FROM** inserted);  **END**;  **GO** | |

Resultado general  
Imagen de la pantalla de un celular de un mensaje en letras negras

El contenido generado por IA puede ser incorrecto.

## Tarea de Generar Backups

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

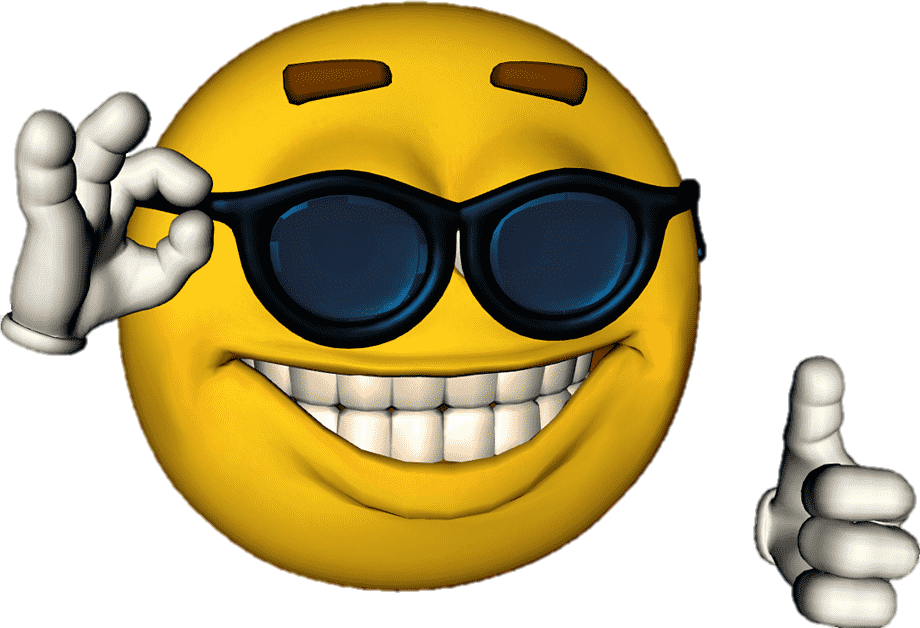
El contenido generado por IA puede ser incorrecto.

Se añade dos tareas correspondientes para generar los backups de las dos bases de datos, como medida preventiva

Codigo SQL

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | *-- Backup de la base de datos principal*  BACKUP **DATABASE** jardineria  **TO** DISK = 'C:\Backups\jardineria\_backup.bak'  **WITH** FORMAT, INIT,  NAME = 'Backup Completo de jardineria',  SKIP, NOREWIND, NOUNLOAD, STATS = 10;  **GO**  *-- Backup de la base de datos de staging*  BACKUP **DATABASE** DBO\_Stagging\_Jardineria  **TO** DISK = 'C:\Backups\jardineria\_staging\_backup.bak'  **WITH** FORMAT, INIT,  NAME = 'Backup Completo de jardineria\_staging',  SKIP, NOREWIND, NOUNLOAD, STATS = 10;  **GO** | |

**Resultados satisfactorios**

Diagrama

El contenido generado por IA puede ser incorrecto.

# Conclusiones

* La implementación de una base de datos Staging permite manejar la información de manera más organizada y controlada antes de ser procesada.
* La migración de datos debe realizarse mediante consultas optimizadas para evitar sobrecarga en los sistemas y asegurar la correcta transferencia.
* Es fundamental validar la cantidad de registros en ambas bases de datos para garantizar que no se pierda información durante el proceso de carga.
* La inclusión de campos como **fecha de carga** y **estado de carga** permite un mejor monitoreo y control del proceso ETL.
* La realización de respaldos antes y después del proceso de migración es una práctica clave para mitigar riesgos de pérdida de información.

# Bibliografía

Microsoft. (s.f.). *CREATE TRIGGER (Transact-SQL)*. Microsoft Learn. Recuperado el 16 de marzo de 2025, de <https://learn.microsoft.com/es-es/sql/t-sql/statements/create-trigger-transact-sql>

Aprendiendo SQL Server. (2013). *Programar triggers correctamente: ¿Qué hacer y qué no hacer?*. Blog Aprendiendo SQL Server. Recuperado el 16 de marzo de 2025, de <https://aprendiendosqlserver.blogspot.com/2013/09/programar-triggers-correctamente-que.html>

Blog SSIS Castellano. (2015). *Implementación y carga de dimensiones y tablas de hechos en SSIS*. Recuperado el 16 de marzo de 2025, de https://blogssiscastellano.blogspot.com/2015/07/implementacion-y-carga-de-dimensiones-y\_65.html

Certia. (s.f.). *Seis razones por las que usar staging en proyectos ETL*. Recuperado el 16 de marzo de 2025, de https://www.certia.net/seis-razones-por-las-que-usar-staging