CREACIÓN DE UNA BASE DE DATOS STAGING

Integrantes

Didier Álvarez Bustamante

Jonathan Álvarez Bustamante

Institución Universitaria Digital de Antioquia

Facultad de Ingeniería y Ciencias Agropecuarias

Programa de Ingeniería de Software y Datos

Docente

Antonio Jesús Valderrama Jaramillo

PREICA2502B010064

Base de datos II

18 de septiembre, 2025

Tabla De Contenido

1. Introducción.

Este documento explica, con enfoque académico y práctico, la transformación de la base de datos OLTP Jardinería hacia una base Staging y, desde allí, la construcción de un Modelo Estrella (Data Warehouse de ventas). La motivación principal es separar responsabilidades: el OLTP prioriza transaccionalidad y normalización; Staging prioriza calidad de datos y trazabilidad de las cargas (ETL), minimizando impacto en el sistema fuente y simplificando las transformaciones.

2. Objetivos.

General.

Construir un flujo OLTP → Staging → Modelo Estrella que permita analítica de ventas con integridad y trazabilidad, documentando el diseño, las transformaciones y las validaciones realizadas.

Específicos.

- Normalizar la ingesta en una capa Staging que reciba datos de la base OLTP sin reglas de negocio rigidas pero con tipos compatibles y consistentes.
- Diseñar un Modelo Estrella claro (dimensiones y hechos) orientado a consultas de negocio (por cliente, producto, representante, oficina, estado del pedido y tiempo).
- Definir y ejecutar reglas de transformación y carga (ETL/ELT) desde OLTP hacia staging y staging al DW.

3. Planteamiento del problema.

La base transaccional (OLTP) Jardinería está optimizada para registrar operaciones (pedidos, pagos, etc.), pero no para analítica rápida. Consultas como “ventas por mes, por producto, por oficina y por estado” requieren muchos JOINs, penalizando tiempos y complejidad. Se requiere una arquitectura analítica (DW) que simplifique agregaciones, tendencias y comparativos, garantizando calidad de datos y compatibilidad de tipos.

4. Análisis del problema.

- **Heterogeneidad de tipos**: en OLTP, algunas claves usan **VARCHAR** (producto) y otras **INT** (cliente, oficina). En la Staging deben **concordar** para evitar errores de carga al DW.

- **Estados de pedido**: en OLTP, el estado es texto; en el DW conviene **dimensionalizarlo** para filtrar/segmentar.

- **Tiempo**: las fechas de pedido y entrega requieren una **Dimensión Tiempo** que permita agrupar por año/mes/día.

- **Trazabilidad**: la Staging sirve como **buffer**: conserva datos cercanos al origen y permite re-procesos sin tocar OLTP.

5. Propuesta.

i. Arquitectura general.

- OLTP Jardinería (fuente): tablas de negocio normalizadas.
- Staging Jardinería: copias "enderezadas" para compatibilidad de tipos y controles básicos (nulos, longitudes, trimming).
- DW (Modelo Estrella): dimensiones (Cliente, Producto, Empleado, Oficina, Estado, Tiempo) y Hechos de Ventas con métricas (cantidad, precio unitario, total).

ii. Correcciones realizadas.

Tipado consistente de claves: alinear `ID_producto` como **VARCHAR(15)** en todas las capas (Staging y Hechos) para que coincida con el origen.

Orden de creación: evitar `ALTER TABLE` sobre tablas **antes** de crearlas; definir columnas (p. ej. `region`) al momento de `CREATE TABLE`.

Nulabilidad y longitudes: marcar campos obligatorios (p. ej. fechas) y ajustar longitudes para evitar truncamientos.

Nomenclatura: uniformar prefijos `Stg_` para staging y `dim_/hechos_` en el DW.

Llaves y FKs en DW: asegurar que las **FK** de la tabla de hechos apunten a las dimensiones correspondientes.

Futuro (mejora sugerida): introducir **claves sustitutas** (surrogate keys) y columnas de manejo temporal (SCD) en dimensiones.

iii. Diseño de la staging (tablas y compatibilidades).

Principios: Misma granularidad que en OLTP. Tambien los tipos de datos compatibles con el DW final (sin conversiones riesgosas en la última milla). Además, sin FKs estrictas para agilizar la carga; integridad validada por consultas de control.

Por ejemplo:

```
CREATE TABLE Stg_producto (
ID_producto VARCHAR(15), -- alinear con OLTP
nombre VARCHAR(70),
proveedor VARCHAR(50),
ID_Categoria INT,
descripcion TEXT,
precio_venta NUMERIC(15,2)
);

CREATE TABLE Stg_detalle_pedido (
ID_detalle_pedido INT,
ID_pedido INT,
ID_producto VARCHAR(15), -- corregido: VARCHAR(15)
cantidad INT,
precio_unidad NUMERIC(15,2)
);
```

iv. Reglas de transformación (OLTP a staging).

- **Trim y normalización de texto** (`LTRIM/RTRIM`, mayúsculas coherentes en códigos si aplica).
- **Casting de fechas** a `DATE/DATETIME` según DW.
- **Casting de claves** (p. ej. `ID_producto` a `VARCHAR(15)`).
- **Enriquecimientos** simples (p. ej. derivar `region` si no viene explícita y hay regla de negocio).

Ejemplo:

```
INSERT INTO Stg_producto (ID_producto, nombre, proveedor, ID_Categoria,
descripcion, precio_venta)
SELECT p.ID_producto, p.nombre, p.proveedor, p.Categoria, p.descripcion,
p.precio_venta
FROM jardineria.dbo.producto p;

INSERT INTO Stg_detalle_pedido (ID_detalle_pedido, ID_pedido, ID_producto,
cantidad, precio_unidad)
SELECT ROW_NUMBER() OVER (ORDER BY dp.ID_pedido, dp.ID_producto) AS
ID_detalle_pedido,
dp.ID_pedido,
dp.ID_producto, -- ya como VARCHAR(15)
dp.cantidad,
dp.precio_unidad
FROM jardineria.dbo.detalle_pedido dp;
```

v. Diseño del modelo estrella (DW).

Dimensiones: `dim_cliente`, `dim_producto`, `dim_empleado`, `dim_oficina`, `dim_estado_pedido`, `dim_tiempo`.

Hechos: `hechos_ventas` (grano: **línea de pedido**).
Métricas: `cantidad`, `precio_unitario`, `total_venta`.

Ejemplo de dimensiones y hechos:

```
CREATE TABLE dim_tiempo (
id_fecha INT PRIMARY KEY, -- YYYYMMDD
fecha DATE,
anio INT,
mes INT,
dia INT
);

CREATE TABLE hechos_ventas (
id_hecho INT IDENTITY(1,1) PRIMARY KEY,
id_fecha INT,
id_cliente INT,
id_producto VARCHAR(15),
id_empleado INT,
id_oficina INT,
id_estado INT,
cantidad INT,
precio_unitario NUMERIC(15,2),
total_venta NUMERIC(15,2),
FOREIGN KEY (id_fecha) REFERENCES dim_tiempo(id_fecha),
FOREIGN KEY (id_cliente) REFERENCES dim_cliente(id_cliente),
FOREIGN KEY (id_producto) REFERENCES dim_producto(id_producto),
FOREIGN KEY (id_empleado) REFERENCES dim_empleado(id_empleado),
FOREIGN KEY (id_oficina) REFERENCES dim_oficina(id_oficina),
FOREIGN KEY (id_estado) REFERENCES dim_estado_pedido(id_estado)
);
```

vi. Carga staging a DW.

A continuación se describe un ejemplo de carga de datos de la base de datos staging a DW, para más detalles revisar los documentos .SQL adjuntos en esta entrega.

Dimensión Tiempo (Derivada de fechas en pedidos):

```
INSERT INTO dim_tiempo (id_fecha, fecha, anio, mes, dia)
SELECT DISTINCT CONVERT(INT, FORMAT(p.fecha_pedido, 'yyyyMMdd')) AS id_fecha,
p.fecha_pedido,
YEAR(p.fecha_pedido), MONTH(p.fecha_pedido), DAY(p.fecha_pedido)
FROM Stg_pedido p;
```

- **Descripción del análisis realizado a los datos Jardinería.**

Se cargaron 3 archivos con **poblados**: (1) Datos de OLTP Jardinería, (2) Datos de Staging Jardinería y (3) Datos del DW (Modelo Estrella). A partir de ellos:

- **OLTP → Staging**: inserciones a `Stg_oficina`, `Stg_empleado`, `Stg_cliente`, `Stg_pedido`, `Stg_detalle_pedido`, `Stg_producto`, `Stg_pago` y `Stg_Categoria_producto` manteniendo **tipos compatibles** y la granularidad de origen.
- **Staging → DW**: inserciones a `dim_*` y `hechos_ventas` con grano **línea de pedido**, `id_fecha = YYYYMMDD` y `total_venta = cantidad * precio_unidad`.
- **Notas de calidad de datos**: se documentan posibles **duplicados** (p.ej., clientes) y **nulos** (p.ej., `fecha_entrega`), proponiendo controles.

Consultas de control sobre los poblados reales:

```
-- 1) Duplicados de cliente por (nombre_cliente, ciudad, pais)
SELECT nombre_cliente, ciudad, pais, COUNT(*) AS veces
FROM Stg_cliente
GROUP BY nombre_cliente, ciudad, pais
HAVING COUNT(*) > 1;

-- 2) Productos en detalle sin match en dimensión producto
SELECT d.ID_producto
FROM Stg_detalle_pedido d
LEFT JOIN dim_producto p ON d.ID_producto = p.id_producto
WHERE p.id_producto IS NULL
GROUP BY d.ID_producto;

-- 3) Estados de pedido con variantes/espacios
SELECT estado, COUNT(*)
FROM Stg_pedido
GROUP BY estado
ORDER BY COUNT(*) DESC;

-- 4) Reconciliación líneas de detalle vs. hechos
SELECT (SELECT COUNT(*) FROM hechos_ventas) AS cnt_hechos,
(SELECT COUNT(*) FROM Stg_detalle_pedido) AS cnt_detalle;
```

- Diagramas.

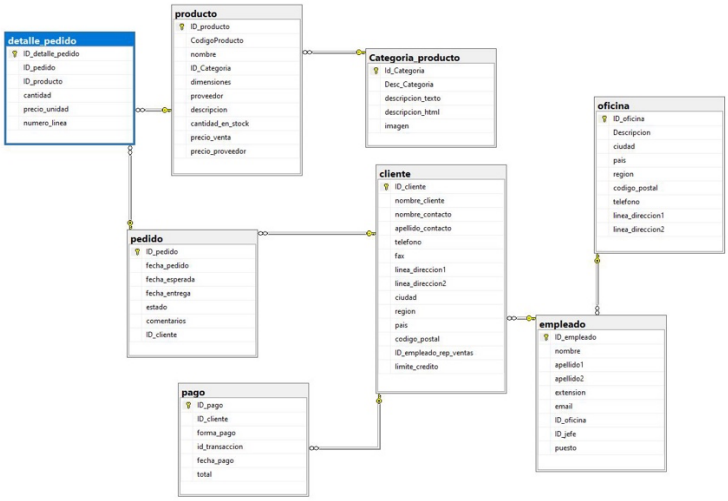

Imagen 1. Esquema relacional de la base de datos jardineria

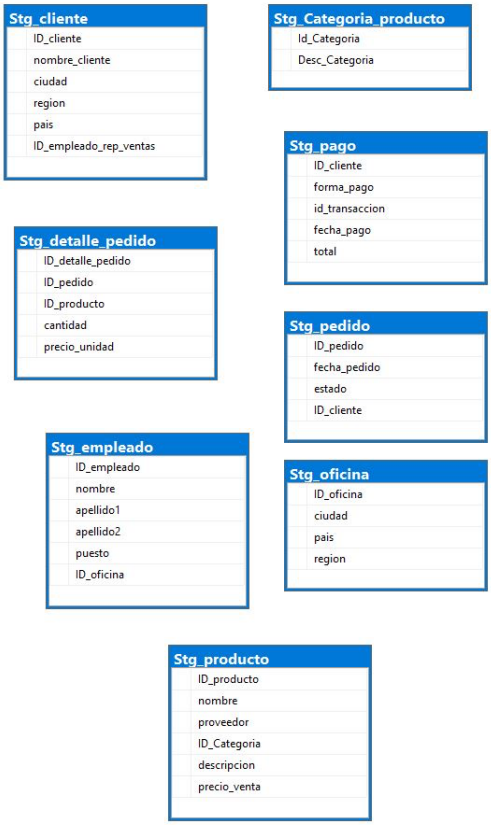

Imagen 2. Modelo staging de la base de datos jardineria

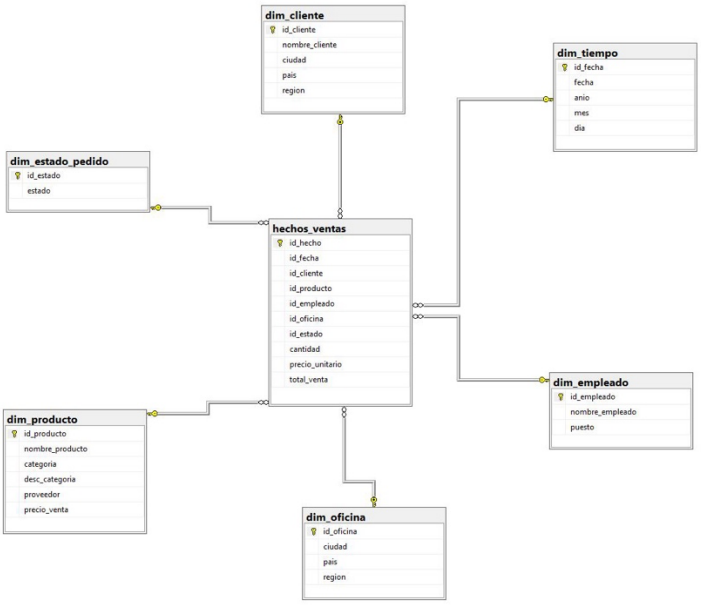

Imagen 3. Modelo estrella jardineria

6. Referencias.

- Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). John Wiley & Sons.

- Inmon, W. H. (2005). *Building the Data Warehouse* (4th ed.). John Wiley & Sons.

- Kimball, R., Thornthwaite, W., Mundy, J., & Becker, B. (2011). *The Microsoft Data Warehouse Toolkit: With SQL Server 2008 R2 and the Microsoft Business Intelligence Toolset* (2nd ed.). John Wiley & Sons.

- Microsoft. (s. f.). *Azure Data Studio documentation*. Microsoft Learn. Recuperado el 18 de septiembre de 2025, de https://learn.microsoft.com/azure-data-studio/

- Microsoft. (s. f.). *CREATE DATABASE (Transact-SQL)*. Microsoft Learn. Recuperado el 18 de septiembre de 2025, de https://learn.microsoft.com/sql/t-sql/statements/create-database-transact-sql

- Microsoft. (s. f.). *INSERT (Transact-SQL)*. Microsoft Learn. Recuperado el 18 de septiembre de 2025, de https://learn.microsoft.com/sql/t-sql/statements/insert-transact-sql

- Microsoft. (s. f.). *BACKUP DATABASE (Transact-SQL)*. Microsoft Learn. Recuperado el 18 de septiembre de 2025, de https://learn.microsoft.com/sql/t-sql/statements/backup-transact-sql

- Microsoft. (s. f.). *SQL Server in Docker containers on Linux*. Microsoft Learn. Recuperado el 18 de septiembre de 2025, de https://learn.microsoft.com/sql/linux/sql-server-linux-docker-container-deployment