Proceso de transformación de datos y carga en el Data Mart Final.

Simón Arbey Castaño Ríos
Juan Pablo Gonzalez Gil
Luisa Fernanda Gómez Quiroz

Docente: Antonio Jesús Valderrama Jaramillo

Institución Universitaria Digital De Antioquia
Facultad de Ingenierías
Ingeniería en Software y Datos
Medellin, Colombia

Bases de Datos

30 de septiembre de 2025

Tabla de contenido

Introducción 3

1 Revisión del modelo estrella (Star Schema). 4

1.1 Tablas principales 4

1.2 Observaciones sobre el diseño 4

1.3 Recomendaciones de modelado 5

2 Verificación y limpieza de la base de datos Staging. 5

3 Proceso ETL documentado: Etapas clave 7

3.1 Extracción 7

3.2 Transformación 7

3.3 Cargar (Load) – Construcción del Data Mart 9

4 Validaciones y controles aplicados (Quality Checks) 10

4.1 Controles realizados 10

5 Resultados y hallazgos principales 11

Introducción

En el marco del proyecto Jardinería, se ha llevado a cabo un proceso integral de extracción, transformación y carga (ETL) con el objetivo de construir un Data Mart eficiente y bien estructurado, basado en un modelo en estrella. Este informe presenta de manera clara y ordenada cada etapa del trabajo realizado, desde la revisión del modelo y la base de datos de Staging, hasta la implementación técnica y los controles de calidad aplicados.

A lo largo del documento encontrarás:

- Una evaluación detallada del modelo estrella y la base de datos intermedia (Staging).
- Los scripts SQL utilizados para extraer, transformar y cargar los datos hacia el Data Mart.
- Las validaciones realizadas para asegurar la integridad y precisión de la información.
- Una presentación final en formato de diapositivas, lista para comunicar los resultados del proyecto.

Este informe busca no solo documentar el proceso técnico, sino también facilitar su compresión y servir como guía para futuras implementaciones similares.

1 Revisión del modelo estrella (Star Schema).

Como parte del diseño del Data Mart para el proyecto Jardinería, se realizó una revisión detallada del modelo en estrella, identificando las principales tablas que lo conforman y evaluando sus relaciones y estructura.

1.1 Tablas principales

- **Dimensiones:**

  Estas tablas contienen los atributos descriptivos que permiten analizar los datos desde diferentes perspectivas:

  - **DimProducto:** Incluye información clave del producto como su código, nombre, proveedor, precio de venta y la categoría a la que pertenece.
  - **DimCategoria:** Define las categorías de productos con su respectiva descripción.
  - **DimCliente:** Contiene datos del cliente como nombre, ciudad, país y límite de crédito.
  - **DimTiempo:** Generada a partir de las fechas de pedidos y pagos, permite realizar análisis temporales por año, mes, día, trimestre, etc.

- **Tablas de hechos:**

  Estas almacenan los eventos transaccionales que se desean analizar:

  - **FactVentas:** Registra cada detalle de pedido, incluyendo producto, cliente, fecha, cantidad, precio unitario y total por línea.
  - **FactPagos:** Resume los pagos realizados por los clientes, vinculados a fechas específicas.

1.2 Observaciones sobre el diseño

- La granularidad de **FactVentas** está definida a nivel de detalle de pedido, lo que permite realizar análisis precisos por cliente, producto, pedido o periodo.
- Se utilizan **claves sustitutas** (por ejemplo, *_sk) para facilitar las relaciones entre tablas y permitir la historización de datos en caso de cambios lentos
- La tabla **DimCategoria** está adecuadamente normalizada, y **DimProducto** la referencia correctamente.

- La generación de **DimTiempo** a partir de fechas reales es una práctica recomendada para facilitar análisis temporales consistentes.

1.3 Recomendaciones de modelado

- Evaluar la implementación de **SCD Tipo 2** en **DimCliente** si se requiere conservar el historial de cambios en datos como dirección o límite de crédito
- Considerar agregar un campo explícito de nombre de categoría (aunque ya existe como desc_categoria) y verificar que categoria_id sea único.
- Validar las cardinalidades entre tablas y asegurarse de que las columnas utilizadas en los joins estén indexadas para mejorar el rendimiento. Los scripts ya incluyen ejemplos de índices que pueden servir como referencia.

2 Verificación y limpieza de la base de datos Staging.

Durante la revisión del script StagingJardineria.sql, se identificaron varias inconsistencias y errores menores que deben corregirse antes de ejecutar el script en un entorno SQL Server. Estas observaciones buscan garantizar que el proceso de carga y transformación de datos se realice sin contratiempos.

- **Nombre inconsistente de la base de datos:**
  El script crea la base de datos con **CREATE DATABASE StagingJardineria**, pero luego intenta usar **USE JardineriaStaging**, lo que genera un error. Se recomienda unificar el nombre como **JardineriaStaging,** en línea con lo utilizado en **DataMart.sql**.
- **Errores de sintaxis en definición de tablas:**
  En la instrucción **CREATE TABLE DIMOficina(...),** se detectó una coma adicional al final de la lista de columnas. Esta debe eliminarse para evitar errores de compilación.
- **Falta de terminadores y separadores:**
  Algunas instrucciones INSERT INTO, como la de FactDetallePedido, no terminan correctamente con punto y coma (;) ni están separadas del siguiente bloque, lo que puede generar conflictos al ejecutar el script.

- **Inconsistencias en nombres de objetos:**

  Aunque SQL Server no distingue entre mayúsculas y minúsculas, es buena práctica mantener la consistencia en el uso de nombres de tablas y columnas para facilitar la lectura y mantenimiento del código.

- **Confusión en la base de datos destino de los INSERTs:**

  Se encontró un bloque INSERT INTO JardineriaStaging.dbo.FactPago, mientras que la base de datos creada en el script es StagingJardineria. Es necesario corregir esta discrepancia para evitar errores de ejecución.

```
--Data Mart (Esquema dm) - Drop/Create
IF NOT EXISTS (SELECT 1 FROM sys.schemas WHERE name = 'dm')
  EXEC('CREATE SCHEMA dm AUTHORIZATION dbo');
GO

USE JardineriaStaging;
```

Además, es importante asegurarse de que todas las instrucciones INSERT INTO apunten correctamente a la base de datos JardineriaStaging, tal como fue creada en el script.

3 Proceso ETL documentado: Etapas clave

El desarrollo del Data Mart para el proyecto Jardinería se estructuró en tres etapas fundamentales del proceso ETL: extracción, transformación y carga. A continuación, se detalla cada una de ellas con ejemplos prácticos y observaciones relevantes.

3.1 Extracción

La primera fase consistió en obtener los datos desde la base original jardineria.dbo, que contiene las siguientes tablas: oficina, empleado, cliente, categoria_producto, producto, pedido, detalle_pedido y pago.

La extracción se realizó mediante sentencias INSERT INTO ... SELECT ... FROM, trasladando la información desde las tablas fuente hacia las tablas de staging. Por ejemplo, para poblar la tabla DimCliente:

```
INSERT INTO DimCliente
SELECT ID_Cliente, Nombre_Cliente, Ciudad, Pais, Limite_Credito
FROM jardineria.dbo.cliente;
```

Este enfoque permite centralizar los datos en un entorno intermedio donde pueden ser depurados y transformados antes de su carga final.

3.2 Transformación

Una vez extraídos los datos, se aplicaron diversas operaciones para asegurar su calidad, consistencia y utilidad analítica:

- **Limpieza de datos:** Se eliminaron registros duplicados utilizando ROW_NUMBER y DELETE, y se normalizaron textos con funciones como LTRIM, RTRIM y UPPER para estandarizar espacios y formatos.
- **Normalización de formatos:** Se ajustaron tipos de datos, longitudes y formatos numéricos para garantizar coherencia en el modelo.
- **Enriquecimiento de información:** Se creó la tabla DimTiempo, derivada de las fechas de pedidos y pagos, lo que permite realizar análisis temporales detallados.
- **Cálculos correctivos:** En caso de inconsistencias, se recalcularon totales de pedidos a partir del detalle (SUM(Cantidad * Precio_Unidad)).

Ejemplos de transformaciones aplicadas:

- **Eliminación de duplicados por PK natural:**

```
;WITH d AS (
    SELECT ID_Cliente, ROW_NUMBER() OVER (PARTITION BY ID_Cliente ORDER BY ID_Cliente) rn
    FROM dbo.DimCliente
    )
    DELETE FROM d WHERE rn > 1;
```

- **Estandarización de textos:**

```
UPDATE dbo.DimProducto
SET CodigoProducto = UPPER(LTRIM(RTRIM(CodigoProducto))),
    Nombre         = LTRIM(RTRIM(Nombre)),
    Proveedor      = LTRIM(RTRIM(Proveedor));
```

- **Generación de DimTiempo:**

```
CREATE TABLE dbo.DimTiempo(
  tiempo_sk INT IDENTITY(1,1) PRIMARY KEY,
  Fecha DATE UNIQUE,
  Anio  INT,
  Mes   INT,
  Dia   INT,
  Trimestre INT,
  NombreMes  VARCHAR(15),
  NombreDia  VARCHAR(15)
);

;WITH fechas AS (
  SELECT Fecha_Pedido AS Fecha FROM dbo.FactPedido
  UNION
  SELECT Fecha_Pago   AS Fecha FROM dbo.FactPago
)
INSERT INTO dbo.DimTiempo(Fecha, Anio, Mes, Dia, Trimestre, NombreMes, NombreDia)
SELECT f.Fecha,
       YEAR(f.Fecha),
       MONTH(f.Fecha),
       DAY(f.Fecha),
       DATEPART(QUARTER, f.Fecha),
       DATENAME(MONTH, f.Fecha),
       DATENAME(WEEKDAY, f.Fecha)
FROM fechas f
WHERE f.Fecha IS NOT NULL
GROUP BY f.Fecha;
```

3.3 Cargar (Load) – Construcción del Data Mart

En esta etapa se creó el esquema dm junto con las tablas de dimensiones y hechos, siguiendo el diseño del modelo en estrella. Para cargar los datos:

- Se utilizaron sentencias MERGE para las dimensiones, lo que permite insertar nuevos registros o actualizar los existentes según corresponda.
- Las tablas de hechos se poblaron mediante INSERT SELECT, utilizando expresiones comunes de tabla (CTE) para mapear las claves sustitutas (*_sk).

- **Ejemplo del MERGE para DimCliente:**

```
MERGE dm.DimCliente AS tgt
USING (
  SELECT ID_Cliente AS cliente_id, Nombre_Cliente, Ciudad, Pais, Limite_Credito
  FROM dbo.DimCliente
) AS src
ON (tgt.cliente_id = src.cliente_id AND tgt.vigente = 1)
WHEN MATCHED AND (
     ISNULL(tgt.nombre_cliente,'') <> ISNULL(src.Nombre_Cliente,'')
  OR ISNULL(tgt.ciudad,'')         <> ISNULL(src.Ciudad,'')
  OR ISNULL(tgt.pais,'')           <> ISNULL(src.Pais,'')
  OR ISNULL(tgt.limite_credito,0)  <> ISNULL(src.Limite_Credito,0)
) THEN UPDATE SET
     nombre_cliente = src.Nombre_Cliente,
     ciudad         = src.Ciudad,
     pais           = src.Pais,
     limite_credito = src.Limite_Credito
WHEN NOT MATCHED THEN
  INSERT (cliente_id, nombre_cliente, ciudad, pais, limite_credito)
  VALUES (src.cliente_id, src.Nombre_Cliente, src.Ciudad, src.Pais, src.Limite_Credito);
```

- **Carga de hechos con mapeo de claves sustitutas:**

```
;WITH map_cl AS (
  SELECT cliente_id, cliente_sk FROM dm.DimCliente WHERE vigente = 1
),
map_tm AS (
  SELECT Fecha, tiempo_sk FROM dm.DimTiempo
)
INSERT INTO dm.FactPagos(pago_id, cliente_sk, tiempo_sk, total)
SELECT pg.ID_Pago,
       mc.cliente_sk,
       mt.tiempo_sk,
       pg.Total
FROM dbo.FactPago pg
INNER JOIN map_cl mc ON mc.cliente_id = pg.ID_Cliente
INNER JOIN map_tm mt ON mt.Fecha      = pg.Fecha_Pago;
```

4 Validaciones y controles aplicados (Quality Checks)

Para asegurar la calidad y consistencia de los datos en el Data Mart del proyecto Jardinería, se implementaron diversas validaciones extraídas del script DataMart.sql. Estas verificaciones permiten detectar errores, inconsistencias y asegurar que la información cargada sea confiable para el análisis.

4.1 Controles realizados

- **Conteo de registros por tabla:** Se compararon los volúmenes de datos entre las tablas de staging y las del Data Mart (dm) para verificar que no haya pérdidas durante el proceso ETL.
- **Detección de valores nulos en claves sustitutas (SKs):** Se revisaron las tablas de hechos, especialmente FactVentas, para identificar registros que no estén correctamente relacionados con sus dimensiones (producto, cliente, tiempo).
- **Verificación de paridad de totales:** Se calcularon los totales de ventas en staging (Cantidad * Precio_Unidad) y se compararon con los totales registrados en el Data Mart (total_linea) para asegurar que los montos coincidan.
- **Revisión puntual por pedido (spot check):** Se analizaron pedidos específicos para detectar diferencias entre el total registrado en staging y la suma de sus líneas en el Data Mart.
- **Comparación de pagos por cliente:** Se contrastaron los totales de pagos por cliente entre staging y el Data Mart para validar la integridad de la información financiera.

- **Conteo de registro por tablas:**

```
--Conteos
SELECT 'DimCliente' AS Tabla, COUNT(*) AS Registros FROM dm.DimCliente
UNION ALL SELECT 'DimCategoria', COUNT(*) FROM dm.DimCategoria
UNION ALL SELECT 'DimProducto', COUNT(*) FROM dm.DimProducto
UNION ALL SELECT 'DimTiempo', COUNT(*) FROM dm.DimTiempo
UNION ALL SELECT 'FactVentas', COUNT(*) FROM dm.FactVentas
UNION ALL SELECT 'FactPagos', COUNT(*) FROM dm.FactPagos;

--Nulos en SKs
SELECT TOP 50 * FROM dm.FactVentas WHERE producto_sk IS NULL OR cliente_sk IS NULL OR tiempo_sk IS NULL;
SELECT TOP 50 * FROM dm.FactPagos  WHERE cliente_sk IS NULL OR tiempo_sk IS NULL;

--Paridad de totales (Detalle vs DM)
SELECT SUM(CONVERT(DECIMAL(18,2), dp.Cantidad) * dp.Precio_Unidad) AS total_staging
FROM dbo.FactDetallePedido dp;

SELECT SUM(total_linea) AS total_dm
FROM dm.FactVentas;
```

5 Resultados y hallazgos principales

Una vez corregidos los inconvenientes detectados en la base de datos de staging y ejecutadas las cargas definidas en el proceso ETL, se logró poblar correctamente el Data Mart. Las tablas de dimensiones fueron completadas con sus respectivas claves sustitutas, y las tablas de hechos contienen ahora los registros necesarios para el análisis.

Durante la etapa de limpieza, se identificaron duplicados en algunas tablas de staging. Estos fueron eliminados utilizando la técnica de numeración con ROW_NUMBER(), lo que permitió conservar únicamente los registros válidos.

También se detectó una discrepancia inicial entre el total calculado a partir del detalle de pedidos y el total reportado en la tabla de pedidos. Tras revisar y recalcular los valores —incluyendo la reindexación de la tabla y el ajuste del campo Total con una suma directa (SUM(...))—, las diferencias fueron corregidas satisfactoriamente.