Evidencia de Aprendizaje

S30 - Evidencia de aprendizaje 3. Proceso de transformación de datos y carga en el data mart final

Integrantes

BRAYAN SANTIAGO JARAMILLO AMÉZQUITA

ALEICER VESGA RUEDA

Docente

Antonio Jesus Valderrama Jaramillo

Bases de Datos II

PREICA2502B010064

Ingeniería de software y datos

Institución Universitaria Digital de Antioquia

Medellín, Antioquia

30 de septiembre de 2025

Introducción

Este informe documenta el diseño y la implementación de un proceso de Extracción, Transformación y Carga (ETL), desarrollado para migrar y consolidar datos desde una base de datos transaccional (jardineria) hacia un Data Mart analítico (jardineria_dw).

La idea principal es convertir datos operativos en un formato optimizado para el análisis de negocio. Para lograrlo, se utiliza una base de datos de staging (staging_jardineria) como paso intermedio. Esta área de preparación permite separar el proceso de la base de datos de producción, facilitando la limpieza y manipulación de los datos antes de su carga final.

El Data Mart resultante, estructurado bajo un modelo en estrella, está diseñado para facilitar la generación de reportes y responder a preguntas estratégicas del negocio de manera eficiente, como la identificación de productos clave, el rendimiento de ventas por región y la segmentación de clientes.

Objetivo general

Organizar y transformar la información de la base de datos de Jardinería para llevarla desde el origen hasta un modelo final que permita analizar las ventas y tomar decisiones, como identificar cuáles son los productos más vendidos y el comportamiento de los clientes.

Objetivos específicos:

- Revisar y limpiar los datos que vienen de la base original para asegurar que estén completos y correctos.
- Pasar la información al área de staging como paso intermedio, cuidando que conserve su coherencia.
- Construir un modelo de base de datos en forma de estrella que permita analizar las ventas de manera clara.
- Cargar los datos en el modelo final para facilitar consultas y reportes.
- Realizar consultas sencillas que ayuden a responder preguntas clave para el negocio, como el producto más vendido o el cliente con mayor compra.

1. Preparación

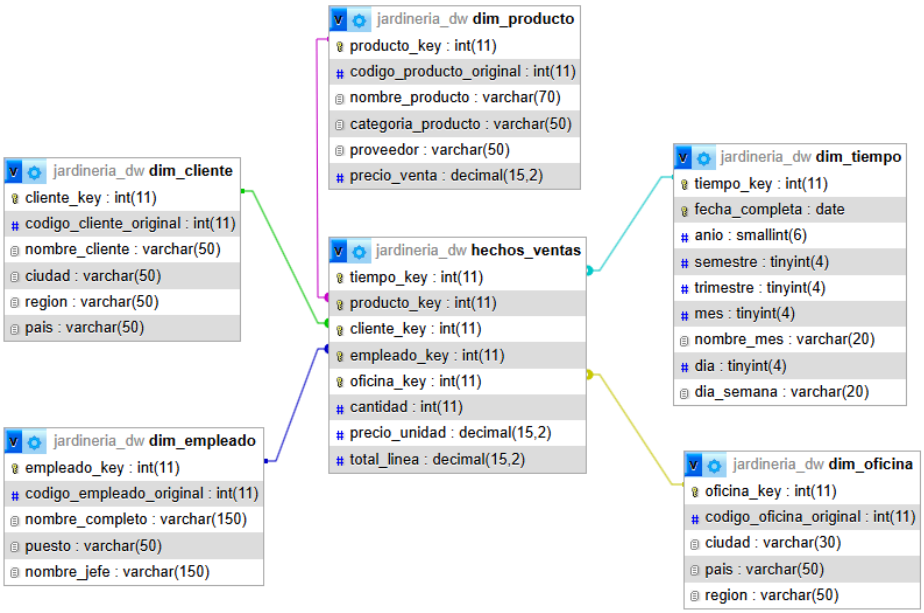
El primer paso del proceso es la construcción de la base de datos de destino, jardineria_dw. Su estructura se basa en un modelo en estrella, el cual se compone de una tabla de hechos central que almacena métricas cuantitativas y un conjunto de tablas de dimensiones que proveen contexto descriptivo.

Tabla de Hechos (hechos_ventas): Almacena las métricas de negocio, como la cantidad de productos vendidos y el total monetario de cada transacción.

Tablas de Dimensiones (dim_tiempo, dim_producto, dim_cliente, dim_oficina, dim_empleado): Contienen los atributos que describen los hechos. Por ejemplo, la dimensión dim_cliente almacena la información geográfica del cliente, mientras que dim_producto describe las características de los artículos vendidos.

Figura 1

Diseño del modelo estrella



Nota. **Imagen tomada de phpMyAdmin.**

2. Extracción de datos desde la base de datos origen hacia la base de datos de Staging

La fase de extracción consiste en transferir los datos desde el sistema transaccional (jardineria) a un área de preparación intermedia (staging_jardineria). El propósito de esta área de staging es aislar el proceso ETL del sistema de producción para evitar impactos en el rendimiento y disponer de un entorno controlado para la manipulación y limpieza de los datos.

Las siguientes consultas ejemplifican el proceso de extracción, donde los datos se copian desde las tablas de origen a sus contrapartes en la base de datos de staging.

Figura 2

Consulta para migrar datos de la tabla clientes

```
-- 1) Clientes
INSERT INTO staging_jardineria.stg_cliente
(source_id, nombre_cliente, nombre_contacto, apellido_contacto, telefono, fax, linea_direccion1,
linea_direccion2, ciudad, region, pais, codigo_postal, empleado_rep_ventas_id, limite_credito, raw_row)
SELECT
  c.ID_cliente,
  TRIM(c.nombre_cliente),
  TRIM(c.nombre_contacto),
  TRIM(c.apellido_contacto),
  NULLIF(TRIM(c.telefono),''),
  NULLIF(TRIM(c.fax),''),
  TRIM(c.linea_direccion1),
  NULLIF(TRIM(c.linea_direccion2),''),
  NULLIF(TRIM(c.ciudad),''),
  NULLIF(TRIM(c.region),''),
  NULLIF(TRIM(c.pais),''),
  NULLIF(TRIM(c.codigo_postal),''),
  c.ID_empleado_rep_ventas,
  c.limite_credito,
  JSON_OBJECT('ID_cliente', c.ID_cliente, 'nombre_cliente', c.nombre_cliente)
FROM jardineria.cliente c
ON DUPLICATE KEY UPDATE
  nombre_cliente = VALUES(nombre_cliente),
  telefono = VALUES(telefono),
  raw_row = VALUES(raw_row);
```

Figura 3

Consulta para migrar datos de la tabla productos

```
-- 2) Productos
INSERT INTO staging_jardineria.stg_producto
(source_id, CodigoProducto, nombre, categoria_id, dimensiones, proveedor,
 descripcion, cantidad_en_stock, precio_venta, precio_proveedor, raw_row)
SELECT
  p.ID_producto,
  p.CodigoProducto,
  TRIM(p.nombre),
  p.Categoria,
  NULLIF(TRIM(p.dimensiones),''),
  NULLIF(TRIM(p.proveedor),''),
  p.descripcion,
  p.cantidad_en_stock,
  p.precio_venta,
  p.precio_proveedor,
  JSON_OBJECT('ID_producto', p.ID_producto, 'nombre', p.nombre)
FROM jardineria.producto p
ON DUPLICATE KEY UPDATE
  nombre = VALUES(nombre),
  cantidad_en_stock = VALUES(cantidad_en_stock),
  precio_venta = VALUES(precio_venta),
  raw_row = VALUES(raw_row);
```

Figura 4

Consulta para migrar datos de la tabla pedidos

```
-- 3) Pedidos
INSERT INTO staging_jardineria.stg_pedido
(source_id, fecha_pedido, fecha_esperada,
 fecha_entrega, estado, comentarios, cliente_id, raw_row)
SELECT
  pd.ID_pedido,
  pd.fecha_pedido,
  pd.fecha_esperada,
  pd.fecha_entrega,
  TRIM(pd.estado),
  pd.comentarios,
  pd.ID_cliente,
  JSON_OBJECT('ID_pedido', pd.ID_pedido, 'estado', pd.estado)
FROM jardineria.pedido pd
ON DUPLICATE KEY UPDATE
  estado = VALUES(estado),
  fecha_entrega = VALUES(fecha_entrega),
  raw_row = VALUES(raw_row);
```

Figura 5

Consulta para migrar datos de la tabla detalle pedido

```
-- 4) Detalle pedido
INSERT INTO staging_jardineria.stg_detalle_pedido
(source_id, pedido_id, producto_id, cantidad,
 precio_unidad, numero_linea, raw_row)
SELECT
  dp.ID_detalle_pedido,
  dp.ID_pedido,
  dp.ID_producto,
  dp.cantidad,
  dp.precio_unidad,
  dp.numero_linea,
  JSON_OBJECT('ID_detalle_pedido', dp.ID_detalle_pedido, 'ID_pedido', dp.ID_pedido)
FROM jardineria.detalle_pedido dp
ON DUPLICATE KEY UPDATE
  cantidad = VALUES(cantidad),
  precio_unidad = VALUES(precio_unidad),
  raw_row = VALUES(raw_row);
```

Figura 6

Consulta para migrar datos de la tabla pagos

```
-- 5) Pagos
INSERT INTO staging_jardineria.stg_pago
(source_id, cliente_id, forma_pago, id_transaccion,
 fecha_pago, total, raw_row)
SELECT
  pg.ID_pago,
  pg.ID_cliente,
  TRIM(pg.forma_pago),
  TRIM(pg.id_transaccion),
  pg.fecha_pago,
  pg.total,
  JSON_OBJECT('ID_pago', pg.ID_pago, 'total', pg.total)
FROM jardineria.pago pg
ON DUPLICATE KEY UPDATE
  total = VALUES(total),
  raw_row = VALUES(raw_row);
```

Figura 7

Consulta para migrar datos de las tablas categoría, empleado, oficina

```
-- 6) Categoría, empleado, oficina (similar)
INSERT INTO staging_jardineria.stg_categoria_producto
(source_id, desc_categoria, descripcion_texto, descripcion_html, imagen, raw_row)
SELECT
  c.Id_Categoria, c.Desc_Categoria, c.descripcion_texto, c.descripcion_html, c.imagen,
  JSON_OBJECT('Id_Categoria', c.Id_Categoria)
FROM jardineria.Categoria_producto c
ON DUPLICATE KEY UPDATE desc_categoria = VALUES(desc_categoria), raw_row = VALUES(raw_row);

INSERT INTO staging_jardineria.stg_empleado
(source_id, nombre, apellido1, apellido2, extension, email, oficina_id, id_jefe, puesto, raw_row)
SELECT
  e.ID_empleado, e.nombre, e.apellido1, e.apellido2, e.extension, e.email, e.ID_oficina, e.ID_jefe, e.puesto,
  JSON_OBJECT('ID_empleado', e.ID_empleado)
FROM jardineria.empleado e
ON DUPLICATE KEY UPDATE email = VALUES(email), raw_row = VALUES(raw_row);

INSERT INTO staging_jardineria.stg_oficina
(source_id, descripcion, ciudad, pais, region, codigo_postal, telefono, linea_direccion1, linea_direccion2, raw_row)
SELECT
  o.ID_oficina, o.Descripcion, o.ciudad, o.pais, o.region, o.codigo_postal, o.telefono, o.linea_direccion1, o.linea_direccion2,
  JSON_OBJECT('ID_oficina', o.ID_oficina)
FROM jardineria.oficina o
ON DUPLICATE KEY UPDATE ciudad = VALUES(ciudad), raw_row = VALUES(raw_row);
```

3. y 4. Transformación y Carga: Staging a Data Mart

Carga de dim_tiempo: Esta dimensión se genera dinámicamente a partir de las fechas de los pedidos. La transformación descompone la fecha en componentes analíticos como año, mes, trimestre y día de la semana, lo cual simplifica las consultas basadas en periodos de tiempo.

Figura 8

Carga de dim_tiempo

```sql
-- Carga de dim_tiempo
INSERT INTO jardineria_dw.dim_tiempo (fecha_completa, anio, semestre, trimestre, mes, nombre_mes, dia, dia_semana)
SELECT DISTINCT
    fecha_pedido AS fecha_completa,
    YEAR(fecha_pedido) AS anio,
    IF(MONTH(fecha_pedido) <= 6, 1, 2) AS semestre,
    QUARTER(fecha_pedido) AS trimestre,
    MONTH(fecha_pedido) AS mes,
    CASE MONTH(fecha_pedido)
        WHEN 1 THEN 'Enero' WHEN 2 THEN 'Febrero' WHEN 3 THEN 'Marzo' WHEN 4 THEN 'Abril'
        WHEN 5 THEN 'Mayo' WHEN 6 THEN 'Junio' WHEN 7 THEN 'Julio' WHEN 8 THEN 'Agosto'
        WHEN 9 THEN 'Septiembre' WHEN 10 THEN 'Octubre' WHEN 11 THEN 'Noviembre' WHEN 12 THEN 'Diciembre'
    END AS nombre_mes,
    DAY(fecha_pedido) AS dia,
    CASE DAYOFWEEK(fecha_pedido)
        WHEN 1 THEN 'Domingo' WHEN 2 THEN 'Lunes' WHEN 3 THEN 'Martes' WHEN 4 THEN 'Miércoles'
        WHEN 5 THEN 'Jueves' WHEN 6 THEN 'Viernes' WHEN 7 THEN 'Sábado'
    END AS dia_semana
FROM staging_jardineria.stg_pedido
WHERE fecha_pedido IS NOT NULL;
```

Carga de dim_producto: La transformación principal aquí es la desnormalización de los datos. Se realiza una unión entre stg_producto y stg_categoria_producto para incluir el nombre de la categoría directamente en la dimensión, evitando uniones adicionales en las futuras consultas analíticas.

Figura 9

Carga de dim_protducto

```sql
-- Carga de dim_producto
INSERT INTO jardineria_dw.dim_producto (id_producto_original, codigo_producto_original, nombre_producto, categoria_producto, proveedor, precio_venta)
SELECT
    p.source_id,
    p.CodigoProducto,
    p.nombre,
    cp.desc_categoria,
    p.proveedor,
    p.precio_venta
FROM staging_jardineria.stg_producto p
LEFT JOIN staging_jardineria.stg_categoria_producto cp ON p.categoria_id = cp.source_id;
```

Carga de dim_cliente y dim_oficina: Para estas dimensiones, la transformación es mínima, consistiendo principalmente en la selección de los campos relevantes para el análisis y excluyendo datos puramente operativos (ej. fax, teléfono).

Figura 10

Carga de dim_cliente

```sql
-- Carga de dim_cliente
INSERT INTO jardineria_dw.dim_cliente (codigo_cliente_original, nombre_cliente, ciudad, region, pais)
SELECT
    source_id,
    nombre_cliente,
    ciudad,
    region,
    pais
FROM staging_jardineria.stg_cliente;
```

Figura 11

Carga de dim_oficina

```sql
-- Carga de dim_oficina
INSERT INTO jardineria_dw.dim_oficina (codigo_oficina_original, descripcion_oficina, ciudad, pais, region)
SELECT
    source_id,
    descripcion,
    ciudad,
    pais,
    region
FROM staging_jardineria.stg_oficina;
```

Carga de dim_empleado:

Esta carga implica dos transformaciones importantes: Concatenación de campos: Se combinan las columnas nombre, apellido1 y apellido en un único campo nombre_completo.

Resolución de jerarquía: Se realiza un auto-JOIN en la tabla stg_empleado para reemplazar el ID_jefe por el nombre completo del jefe, enriqueciendo así la dimensión.

Figura 12

Carga de dim_oficina

```sql
-- Carga de dim_empleado
INSERT INTO jardineria_dw.dim_empleado (codigo_empleado_original, nombre_completo, puesto, nombre_jefe)
SELECT
    e.source_id,
    CONCAT(e.nombre, ' ', e.apellido1, ' ', IFNULL(e.apellido2, '')) AS nombre_completo,
    e.puesto,
    IFNULL(CONCAT(j.nombre, ' ', j.apellido1), 'N/A') AS nombre_jefe
FROM staging_jardineria.stg_empleado e
LEFT JOIN staging_jardineria.stg_empleado j ON e.id_jefe = j.source_id;
```

Carga de la Tabla de Hechos: La carga final es la de la tabla hechos_ventas. Este proceso integra los datos transaccionales con las llaves de las dimensiones ya pobladas. La consulta realiza uniones complejas a través de las tablas de staging para obtener los identificadores necesarios y luego los asocia con las llaves primarias (_key) de las tablas de dimensión.

Se aplica una regla de negocio crítica: únicamente se cargan los registros de pedidos cuyo estado sea 'Entregado', para asegurar que el análisis se base en ventas confirmadas.

Figura 13

Carga de hechos_ventas

```sql
INSERT INTO jardineria_dw.hechos_ventas (
    tiempo_key, producto_key, cliente_key, empleado_key, oficina_key,
    cantidad, precio_unidad, total_linea
)
SELECT
    dt.tiempo_key,
    dp.producto_key,
    dc.cliente_key,
    de.empleado_key,
    do.oficina_key,
    det.cantidad,
    det.precio_unidad,
    (det.cantidad * det.precio_unidad) AS total_linea
FROM staging_jardineria.stg_detalle_pedido det
-- Uniones con tablas de Staging para obtener los IDs originales
JOIN staging_jardineria.stg_pedido ped ON det.pedido_id = ped.source_id
JOIN staging_jardineria.stg_cliente cli ON ped.cliente_id = cli.source_id
JOIN staging_jardineria.stg_empleado emp ON cli.empleado_rep_ventas_id = emp.source_id
-- Uniones con las Dimensiones para obtener las llaves foráneas (keys) del Data Mart
JOIN jardineria_dw.dim_tiempo dt ON ped.fecha_pedido = dt.fecha_completa
JOIN jardineria_dw.dim_producto dp ON det.producto_id = dp.id_producto_original -- **UNIÓN CORREGIDA**
JOIN jardineria_dw.dim_cliente dc ON ped.cliente_id = dc.codigo_cliente_original
JOIN jardineria_dw.dim_empleado de ON emp.source_id = de.codigo_empleado_original
JOIN jardineria_dw.dim_oficina do ON emp.oficina_id = do.codigo_oficina_original -- **UNIÓN CORREGIDA**
-- Aplicación de la regla de negocio
WHERE ped.estado = 'Entregado';
```

Conclusiones

1. Con este trabajo logramos organizar la información de la base de datos de Jardinería de una manera más clara y estructurada, lo que permite analizarla con mayor facilidad.
2. El uso del proceso por etapas (origen → staging → data mart) ayudó a asegurar que los datos pasaran por una limpieza y transformación antes de llegar al modelo final.
3. El modelo en estrella nos permitió concentrar la información en torno a las ventas y relacionarlas con clientes, productos, empleados, oficinas y tiempo, lo cual hace posible responder preguntas importantes.
4. Con este esquema es mucho más sencillo identificar productos más vendidos, conocer el comportamiento de los clientes y obtener información útil para la toma de decisiones.
5. En general, este proyecto mostró la importancia de preparar bien los datos antes de analizarlos, ya que la calidad y la organización de la información son clave para obtener conclusiones confiables.

Bibliografía

de Arregui, M. (2024, agosto 27). Cómo utilizar el modelo estrella en una base de datos. *OBS Business School*. https://www.obsbusiness.school/blog/como-utilizar-el-modelo-estrella-en-una-base-de-datos

JulCsc. (s/f). *Introducción a los datamarts*. Microsoft.com. Recuperado el 30 de septiembre de 2025, de https://learn.microsoft.com/es-es/power-bi/transform-model/datamarts/datamarts-get-started

Star schema. (2022, abril 21). Databricks. https://www.databricks.com/glossary/star-schema