Evidencia de Aprendizaje

S30 - Evidencia de aprendizaje 2. Creación de una base de datos de Staging

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

16 de septiembre de 2025

Introducción

El manejo adecuado de la información es un aspecto clave para el funcionamiento de cualquier organización. En este proyecto se trabaja con la base de datos Jardinería, la cual contiene datos relacionados con clientes, empleados, productos, pedidos y pagos. El objetivo principal es analizar esta información y seleccionar los datos más relevantes para trasladarlos a una base de datos Staging. Este proceso permite organizar y preparar la información de manera más clara y estructurada, lo que facilita su validación, respaldo y uso posterior en diferentes aplicaciones. De esta forma, se garantiza una gestión eficiente de los datos y se sientan las bases para futuras implementaciones o análisis más avanzados.

\

Objetivos

Objetivo General: Construir una base de datos Staging funcional para consolidar la información de ventas de la empresa Jardinería, facilitando su posterior análisis.

O**bjetivos Específicos:**

- Analizar la estructura y los datos de la base de datos Jardinería.
- Diseñar un modelo de datos des normalizado para la base de datos Staging.
- Desarrollar y ejecutar los scripts SQL para la creación y población de la base de datos jardineria_staging.
- Validar la integridad y consistencia de los datos migrados.

Planteamiento del Problema

La base de datos Jardinería tiene un diseño normalizado, ideal para registrar transacciones diarias (como registrar un nuevo pedido), pero es ineficiente para generar reportes analíticos complejos.

Realizar consultas que involucren ventas, clientes, productos y empleados requiere unir múltiples tablas (JOINs), lo que puede ser lento y complejo.

La empresa necesita una vista consolidada de sus datos de ventas para tomar decisiones estratégicas, pero la estructura actual lo dificulta.

1. Análisis de los datos almacenados en la base de datos Jardinería

Para el análisis de la base de datos “jardineria” se tienen en cuenta las tablas con las entidades más principales para el negocio: Tablas relevantes (del script original): cliente, pedido, detalle_pedido, producto, Categoria_producto, pago, empleado, oficina. Estas son las que contienen negocio/ventas/inventario que son útiles para análisis y pruebas.

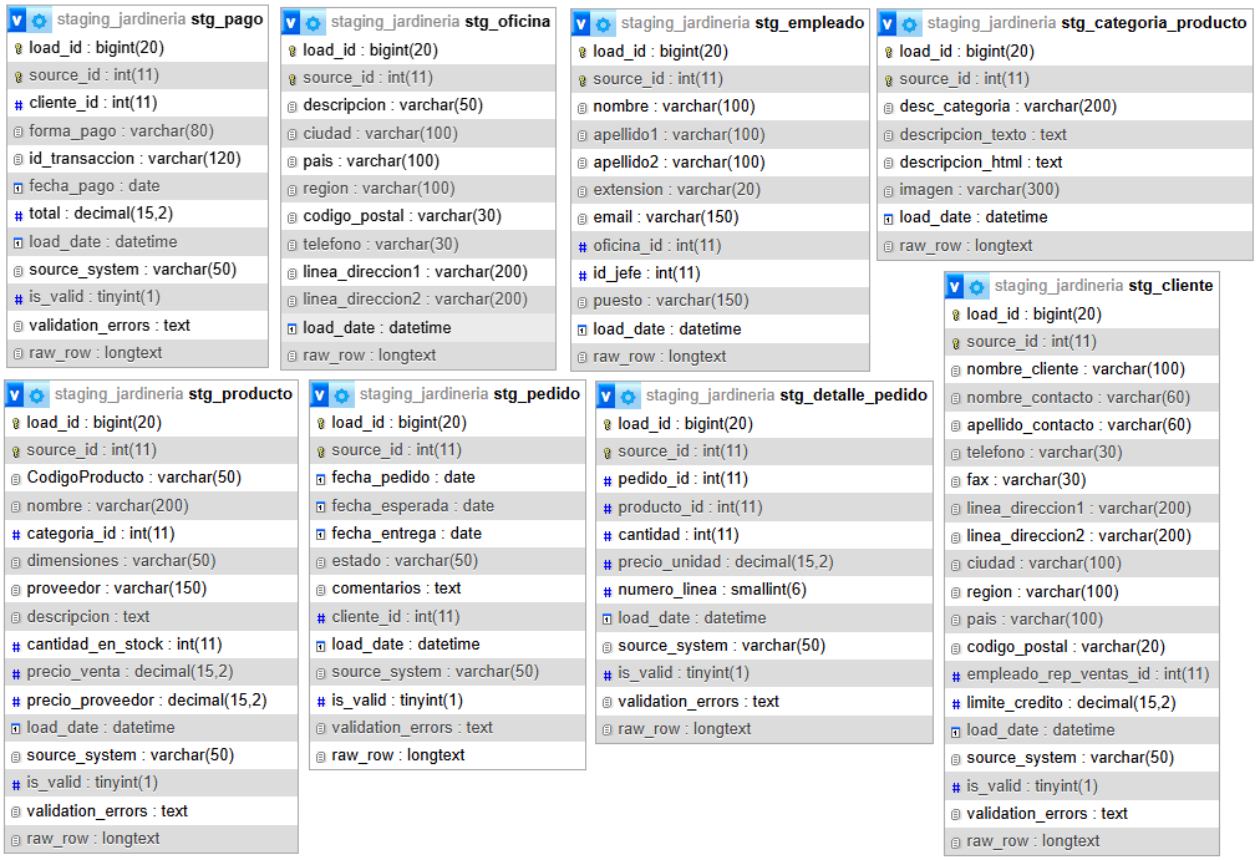
- cliente: ID_cliente, nombre_cliente, telefono, ciudad, ID_empleado_rep_ventas, limite_credito — necesario para segmentación y unión con pedido.
- pedido: ID_pedido, fecha_pedido, estado, ID_cliente — para análisis de pedidos y tiempos de entrega.
- detalle_pedido: ID_pedido, ID_producto, cantidad, precio_unidad — para análisis línea a línea (ingresos, unidades).
- producto + Categoria_producto: datos de catálogo (precios, stock, categoría).
- pago: ID_pago, ID_cliente, fecha_pago, total — conciliación pagos/pedidos.
- empleado, oficina: para enlazar representantes / áreas de venta.

2. Construcción de la base de Datos Staging

Diseñar la estructura tablas que estarán en la base de datos Staging

Figura 1

Diseño de tablas en la base de datos Staging



3. Consultas para Migrar los Datos (Proceso ETL)

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);
```

4. Ejecutar las consultas y validar que los datos queden almacenados correctamente en la Base de datos *Staging*

Figura 8

Resultados obtenidos a partir de la ejecución de las consultas

✔ 36 filas insertadas.
La Id de la fila insertada es: 36 (La consulta tardó 0,0065 segundos.)

```
-- 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);
```

[Editar en línea] [Editar] [Crear código PHP]

✔ 276 filas insertadas.
La Id de la fila insertada es: 276 (La consulta tardó 0,0205 segundos.)

```
-- 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);
```

[Editar en línea] [Editar] [Crear código PHP]

✔ 115 filas insertadas.
La Id de la fila insertada es: 115 (La consulta tardó 0,0086 segundos.)

```
-- 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);
```

[Editar en línea] [Editar] [Crear código PHP]

✔ 318 filas insertadas.
La Id de la fila insertada es: 318 (La consulta tardó 0,0089 segundos.)

```
-- 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);
```

[Editar en línea] [Editar] [Crear código PHP]

✔ 26 filas insertadas.
La Id de la fila insertada es: 26 (La consulta tardó 0,0033 segundos.)

```
-- 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);
```

[Editar en línea] [Editar] [Crear código PHP]

✔ 5 filas insertadas.
La Id de la fila insertada es: 5 (La consulta tardó 0,0038 segundos.)

```
-- 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);
```

[Editar en línea] [Editar] [Crear código PHP]

✔ 31 filas insertadas.
La Id de la fila insertada es: 31 (La consulta tardó 0,0054 segundos.)

```
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);
```

[Editar en línea] [Editar] [Crear código PHP]

✔ 9 filas insertadas.
La Id de la fila insertada es: 9 (La consulta tardó 0,0036 segundos.)

```
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);
```

[Editar en línea] [Editar] [Crear código PHP]

Conclusiones

1. El análisis de la base de datos Jardinería permitió identificar la relevancia de la información y establecer qué datos debían trasladarse a la base de datos Staging, logrando así un proceso más ordenado y coherente.
2. La construcción de la base Staging demostró la importancia de diseñar estructuras de tablas adecuadas que faciliten la integración y validación de la información, asegurando consistencia y claridad en los registros almacenados.
3. La ejecución de consultas para migrar los datos y su posterior validación garantizó que la información se transfiriera correctamente, minimizando errores y manteniendo la integridad de los datos.
4. La creación de respaldos (BK) de ambas bases de datos refuerza las buenas prácticas en la gestión de la información, aportando seguridad y asegurando la disponibilidad de los datos en caso de fallas o imprevistos.
5. Finalmente, el desarrollo de este trabajo permitió afianzar conocimientos prácticos en el manejo de bases de datos, desde el análisis inicial hasta la documentación y presentación de resultados, destacando la importancia de una gestión estructurada y confiable de la información.

Bibliografía

- Calzada, J. M. (2020, noviembre 5). Seis razones por las que usar staging -. Consultoria Certia. Formación | Consultoria | Desarrollo; Consultoria Certia. https://www.certia.net/seis-razones-por-las-que-usar-staging/
- Hermenau, R. (2023, abril 7). The fastest way to migrate MySQL databases to another server with mysqldump. Wp-staging.com; WP STAGING. https://wp-staging.com/migrating-a-mysql-database-to-another-server-using-mysqldump/
- Linhartová, B. (2024, junio 9). The role of staging tables in database administration. Baremon | Data Experts. https://www.baremon.eu/staging-tables-in-database-administration/