Evidencia de aprendizaje 4.Proyecto integrador-Repositorio de todas las actividades

Elaborado por:

Damián Pérez Salvatierra

Catalina Vinasco Arredondo

Yilma Valeria Espitia Sanabria

Docente:

Víctor Hugo Mercado

Curso:

Base de Datos II

Grupo: 76

Ingeniería de Software y Datos

Facultad de Ingeniería y de Ciencias Agropecuarias

Institución Universitaria Digital de Antioquia

2024

**Introducción**

Con base en los fundamentos expuestos anteriormente sobre la importancia de la gestión de datos y la implementación de modelos estrella Data Mart en entornos empresariales, el siguiente paso lógico es llevar a cabo la construcción práctica de dichos modelos utilizando la base de datos de jardinería proporcionada. Este proceso no solo implica la creación del modelo estrella Data Mart, sino también la implementación de una base de datos staging para facilitar la integración y gestión de los datos de manera eficiente.

La base de datos staging desempeña un papel crucial al permitir la incorporación de metadatos detallados, como fechas de incorporación y posibles modificaciones, lo que contribuye significativamente a la trazabilidad y calidad de los datos en el proceso de almacenamiento y análisis. Además, al unir este enfoque con la utilización de Visual Community como plataforma de visualización y análisis de datos, se potencia aún más la capacidad de extracción de información útil para la toma de decisiones empresariales.

Al combinar estas dos perspectivas, la construcción del modelo estrella Data Mart en una base de datos staging y su posterior implementación en Visual Community, se crea un flujo de trabajo integral que optimiza tanto la gestión como el análisis de datos críticos para la empresa. Este enfoque no solo mejora la eficiencia operativa, sino que también potencia la capacidad de identificar tendencias, patrones y oportunidades de mejora que puedan impulsar el rendimiento empresarial.

**Objetivo**

1. Construir un modelo estrella para un *data mart* que permita analizar y responder a tres categorías específicas: identificar el producto más vendido, la categoría con más productos y el año con más ventas, utilizando la base de datos Jardinería como fuente de datos.
2. Construir consultas SQL para la creación de una base de datos staging basada en el modelo relacional de la base de datos “Jardinería” aprovechando la capacidad de Visual Community para el análisis de datos. Teniendo en cuenta los requerimientos de la evidencia dos y las correcciones previas de la actividad anterior.
3. Desarrollar un proceso de transformación desde la base de datos origen, pasar *Staging* y luego hasta el *data mart* final, utilizando la base de datos de *staging* previamente creada. El objetivo es asegurar que los datos estén preparados para el análisis, incluyendo la identificación del producto más vendido y otros análisis relevantes para la toma de decisiones empresariales.

**Objetivos Específicos:**

* Identificar dimensiones relevantes, que contribuyan al análisis de ventas.
* Desarrollar el proceso ETL minuciosamente, haciendo una adecuada limpieza de datos de la base de datos jardinería.
* Revisar el diseño actual de la base de datos, eliminando duplicidad y mejorando la eficiencia del almacenamiento de datos.
* Desarrollar tablas de hechos y dimensiones donde se visualice el modelo estrella, teniendo como respaldo herramientas de administración de bases de datos.
* Desarrollar un proceso de transformación desde la base de datos origen, pasar *Staging* y luego hasta el *data mart* final, utilizando la base de datos de *staging* previamente creada.

**Planteamiento del problema**

Si bien en la base de datos de Jardinería se encuentra la relación entre varias tablas relacionadas al cliente, empleado, producto y geografía, estas tablas no proporcionan la información adecuada para lograr responder las preguntas del negocio. En primer lugar, la falta de conexión entre estas tablas y la dispersión de los datos de la base de datos de Jardinería hace que sea una tarea compleja el lograr identificar con la granularidad necesaria para saber cuál es el producto más vendido y cuál es la categoría con más productos. En segundo lugar, la falta de un modelo estrella dificulta la correcta interpretación y análisis de la información disponible no solo de las preguntas del negocio, sino también la identificación de posibles problemas, relaciones, patrones e incluso oportunidades que llegarán a presentar entre los datos.

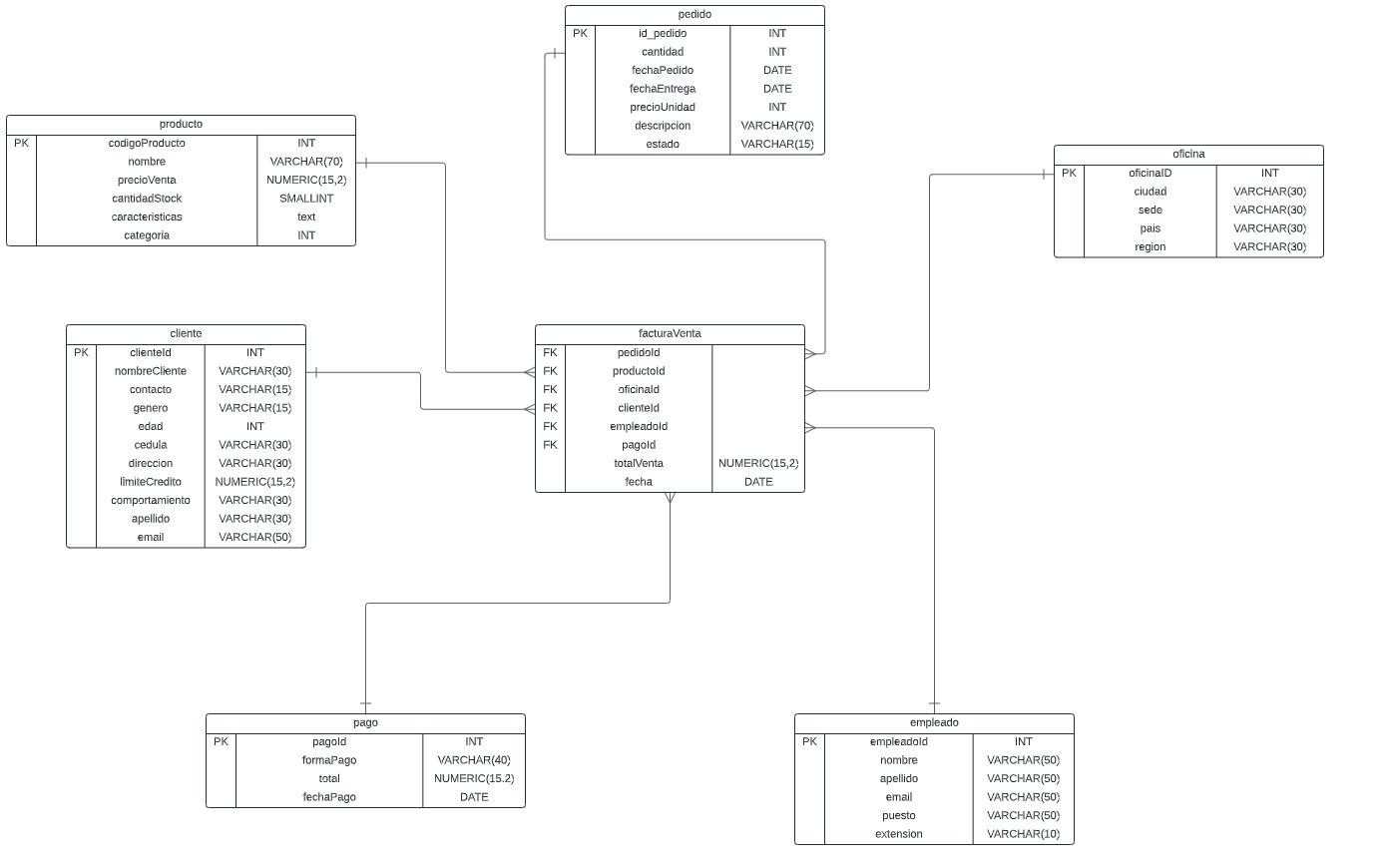
Por otro lado, la falta de un modelo Staging hace que se carezca de un entorno dedicado para la consolidación, limpieza y transformación de los datos antes de su integración en el modelo estrella. La falta de este paso puede llevar a inconsistencias en los datos, duplicando registros y dificultando el proceso de ETL. Por lo tanto, la ausencia de un modelo Staging representa un obstáculo significativo para la orientación de información precisa y significativa a partir de la base de datos jardinería.

**Análisis del Problema**

De acuerdo al análisis minuciosamente realizado dentro de nuestro grupo, pudimos observar que, gracias al modelo de estrella, hay una forma de identificar las tres categorías solicitadas en la evidencia solicitada. Ya que el problema radica en la falta de conexión entre las tablas y la dispersión de datos en la base de datos de Jardinería, al implementar un modelo estrella con FacturaVentas y dimensiones resolvería estas deficiencias.Además, pudimos observar que, gracias al modelo de estrella, hay una forma de identificar las tres categorías solicitadas en la evidencia solicitada. Ya que el problema radica en la falta de conexión entre las tablas y la dispersión de datos en la base de datos de Jardinería, al implementar un modelo estrella con facturaVenta y dimensiones resolvería estas deficiencias. Para abordar la dispersión de datos en la base de datos de Jardinería y facilitar la implementación del modelo estrella, proponemos la creación de una base de datos Staging la cual servirá como un entorno intermedio donde se puedan consolidar, limpiar y transformar los datos de origen. Mediante el uso de base de datos de Staging podremos realizar procesos de ETL de manera eficiente y así garantizando una mejor estructura de la información.

**Modelo Estrella**

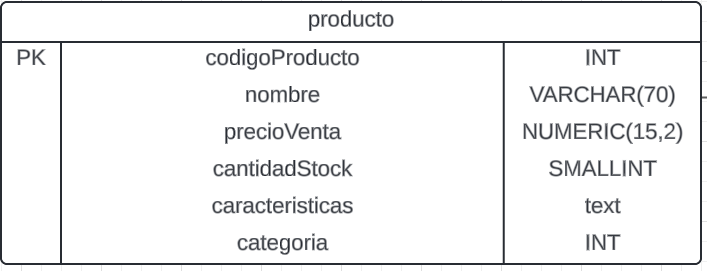
Se parte de una tabla de hechos FacturaVentas, relacionada a seis dimensiones, pedido, producto, cliente, oficina, empleado, pago; las cuales están relacionadas cada dimensión de 1 a muchos con la tabla de hechos. Dichas relaciones y modelo nos permitirán entre otras, resolver las preguntas propuestas en el objetivo.



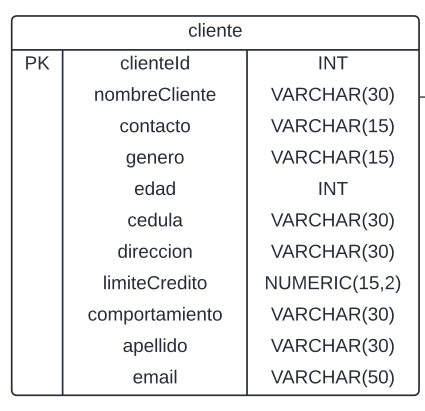
**Lista de dimensiones propuestas**

Logramos identificar seis dimensiones que son las siguientes:

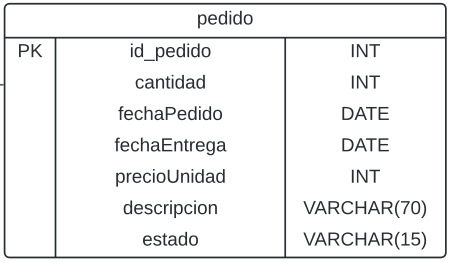
***Producto:***



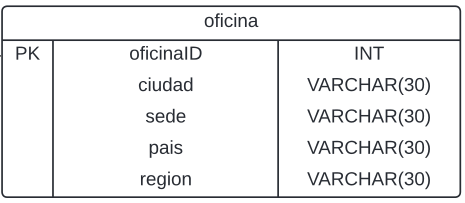
***Cliente***



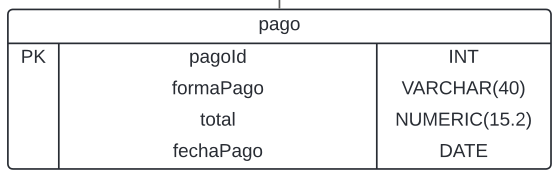
***Pedido***



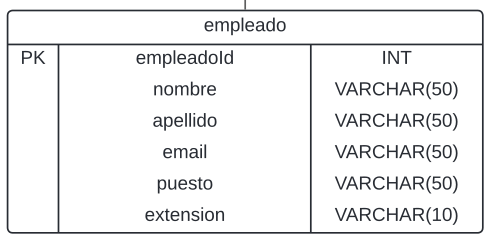
***Oficina***



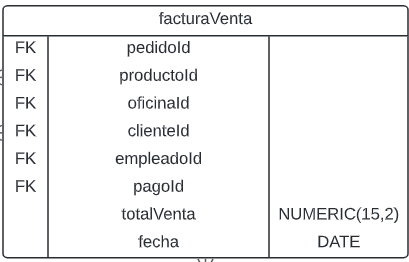
***Pago***



***Empleado***



***Tabla de Hechos***



**Tablas**

CREATE TABLE pedido (

id\_pedido INT,

cantidad INT,

fechaPedido DATE,

fechaEntrega DATE,

precioUnidad INT,

descripcion VARCHAR(70),

estado VARCHAR(15),

PRIMARY KEY (id\_pedido)

);

CREATE TABLE producto (

codigoProducto INT,

nombre VARCHAR(70),

precioVenta NUMERIC(15,2),

cantidadStock SMALLINT,

caracteristicas text,

categoria INT,

PRIMARY KEY (codigoProducto)

);

CREATE TABLE oficina (

oficinaID INT,

ciudad VARCHAR(30),

sede VARCHAR(30),

pais VARCHAR(30),

region VARCHAR(30),

PRIMARY KEY (oficinaID)

);

CREATE TABLE cliente (

clienteId INT,

nombreCliente VARCHAR(30),

contacto VARCHAR(15),

genero VARCHAR(15),

edad INT,

cedula VARCHAR(30),

direccion VARCHAR(30),

limiteCredito NUMERIC(15,2),

comportamiento VARCHAR(30),

apellido VARCHAR(30),

email VARCHAR(50),

PRIMARY KEY (clienteId)

);

CREATE TABLE empleado (

empleadoId INT,

nombre VARCHAR(50),

apellido VARCHAR(50),

email VARCHAR(50),

puesto VARCHAR(50),

extension VARCHAR(10),

PRIMARY KEY (empleadoId)

);

CREATE TABLE pago (

pagoId INT,

formaPago VARCHAR(40),

total NUMERIC(15.2),

fechaPago DATE,

PRIMARY KEY (pagoId)

);

CREATE TABLE FacturaVenta (

pedidoId INT,

productoId INT,

oficinaId INT,

clienteId INT,

empleadoId INT,

pagoId INT,

totalVenta NUMERIC(15,2),

fecha DATE,

CONSTRAINT FK\_fact\_ventaPedidoId FOREIGN KEY (pedidoId) REFERENCES pedido(id\_pedido),

CONSTRAINT FK\_fact\_ventaProductoId FOREIGN KEY (productoId) REFERENCES producto(codigoProducto),

CONSTRAINT FK\_fact\_ventaOficinaId FOREIGN KEY (oficinaId) REFERENCES oficina(oficinaID),

CONSTRAINT FK\_fact\_ventaClienteId FOREIGN KEY (clienteId) REFERENCES cliente(clienteId),

CONSTRAINT FK\_fact\_ventaPagoId FOREIGN KEY (pagoId) REFERENCES pago(pagoId),

CONSTRAINT FK\_fact\_ventaEmpleadoId FOREIGN KEY (empleadoId) REFERENCES empleado(empleadoId)

);

**Propuestas de la solución con:**

**Correcciones a la entrega 1**

Se hace la corrección de la gráfica correspondiente al modelo estrella con su respectivo script.

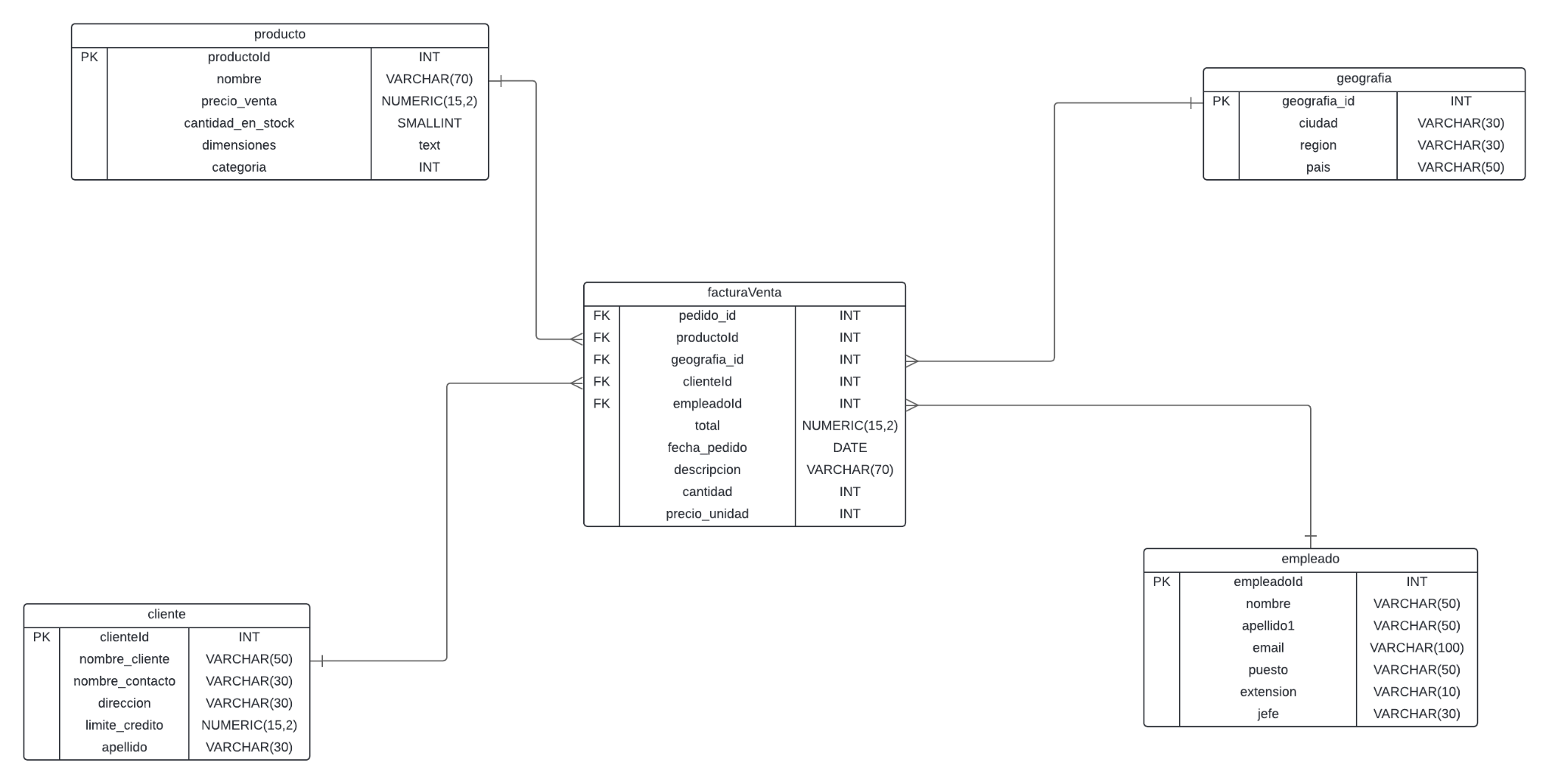
**Modelo estrella, base de datos de Jardinería**

Imagen 1. Modelo estrella de Jardinería

**Script**

Este script se utilizó como plantilla para la creación de tablas del modelo estrella

CREATE TABLE [empleado] (

[empleadoId] INT,

[nombre] VARCHAR(50),

[apellido1] VARCHAR(50),

[email] VARCHAR(50),

[puesto] VARCHAR(50),

[extension] VARCHAR(10),

[jefe] VARCHAR(30),

PRIMARY KEY ([empleadoId])

);

CREATE TABLE [cliente] (

[clienteId] INT,

[nombreCliente] VARCHAR(30),

[contacto] VARCHAR(15),

[genero] VARCHAR(15),

[edad] INT,

[cedula] VARCHAR(30),

[direccion] VARCHAR(30),

[limiteCredito] NUMERIC(15,2),

[comportamiento] VARCHAR(30),

[apellido] VARCHAR(30),

[email] VARCHAR(50),

PRIMARY KEY ([clienteId])

);

CREATE TABLE [geografia] (

[geografiaId] INT,

[ciudad] VARCHAR(30),

[region] VARCHAR(30),

[pais] VARCHAR(30),

PRIMARY KEY ([geografiaId])

);

CREATE TABLE [producto] (

[codigoProducto] INT,

[nombre] VARCHAR(70),

[precio\_venta] NUMERIC(15,2),

[cantidad\_en\_stock] SMALLINT,

[dimensiones] text,

[categoria] INT,

PRIMARY KEY ([codigoProducto])

);

CREATE TABLE [facturaVenta] (

[pedidoId] INT,

[productoId] INT,

[oficinaId] INT,

[clienteId] INT,

[empleadoId] INT,

[total] NUMERIC(15,2),

[fecha\_pedido] DATE,

[descripcion] VARCHAR(70),

[cantidad] INT,

[precio\_unidad] INT,

CONSTRAINT [FK\_facturaVenta.oficinaId]

FOREIGN KEY ([oficinaId])

REFERENCES [geografia]([geografiaId]),

CONSTRAINT [FK\_facturaVenta.clienteId]

FOREIGN KEY ([clienteId])

REFERENCES [cliente]([nombreCliente]),

CONSTRAINT [FK\_facturaVenta.productoId]

FOREIGN KEY ([productoId])

REFERENCES [producto]([nombre])

);

**Descripción del análisis realizado a los datos *Jardinería* y cómo estos se trasladaron a la base de datos *Staging.***

* Revisión y corrección del modelo estrella de acuerdo a las observaciones dadas por el profesor, garantizando la precisión y coherencia con los requerimientos establecidos. Dicho proceso también fue replicado en el script.
* Se hizo verificación de los campos de las dimensiones del modelo estrella, para asegurarse de que estén correctamente representados en la base de datos de Jardinería, garantizando la integridad de los datos.
* Se hace identificación de las posibles consultas de SQL, para obtener los datos necesarios que se requiere en cada dimensión, todo lo anterior mediante la plataforma de SQL Management Studio.
* Creación de una base de datos de destino llamada “staging076”.
* Después de identificar las consultas que se tendrán en cuenta en el proceso, se procede a crear un nuevo proyecto en el visual Community, mediante el uso de una plantilla en blanco llamada Integration Services Project.
* Se configura la conexión a la base de datos de origen que se llama jardinería y la conexión a la base de datos de destino “staging076”.

**Descripción paso a paso**

Se crea un nuevo flujo de Datos llamado “Extracción Jardinería 076”:

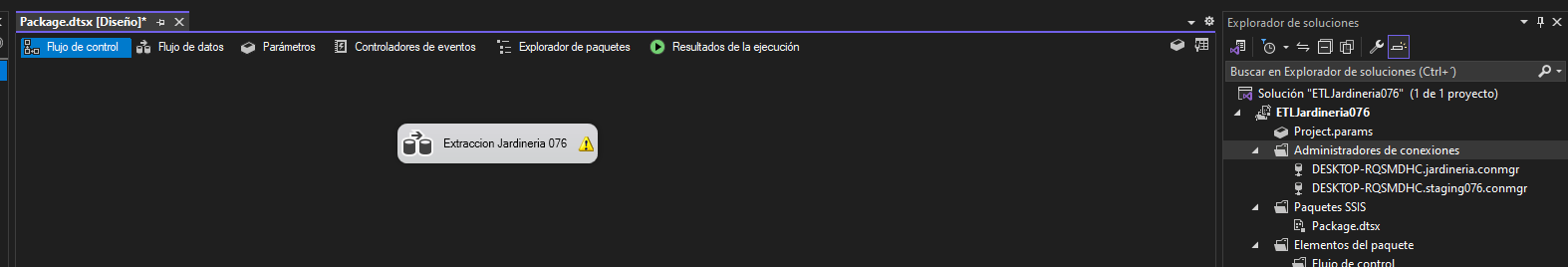


Imagen 2. Creación de Data Flow en Visual Community 2022

Se realiza la extracción de los datos proveniente de la DB de jardinería, teniendo en cuenta el modelo estrella que realizamos en la primera evidencia:

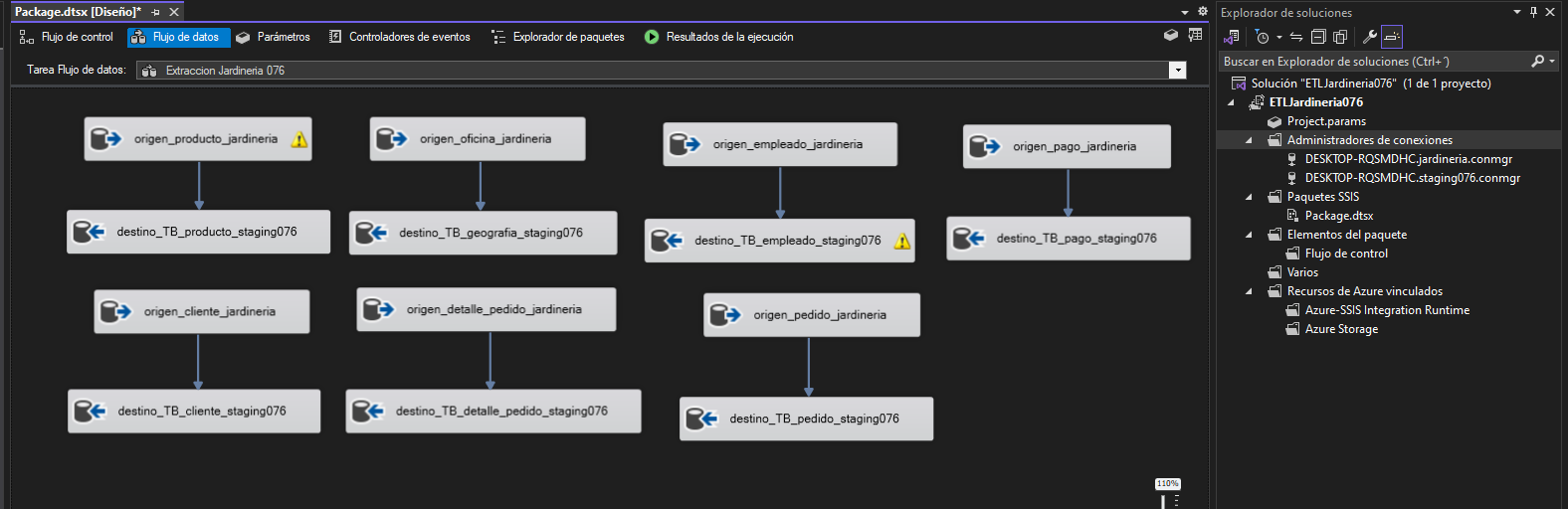


Imagen 3. Creaciones de tabla de origen y destino en Visual Community 2022

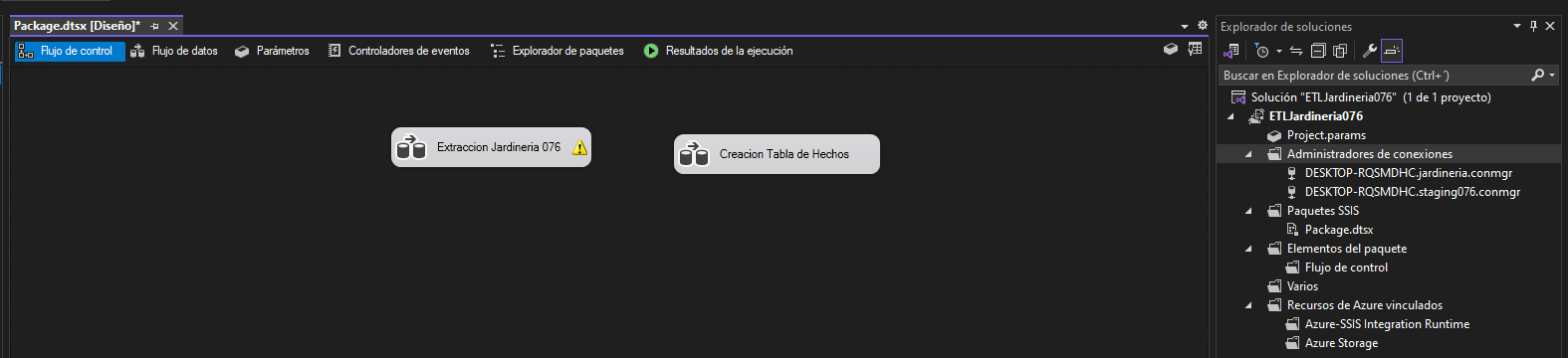
Se crea un nuevo flujo de Datos para la creación de la tabla de hechos llamada “Creación de Tabla de Hechos”:

Imagen 4. Creación de un nuevo Data Flow para la tabla de hechos

Dentro de este flujo de Datos, se extrae de las tablas de la base de datos de staging076 la información necesaria para la creación de la tabla de hechos:

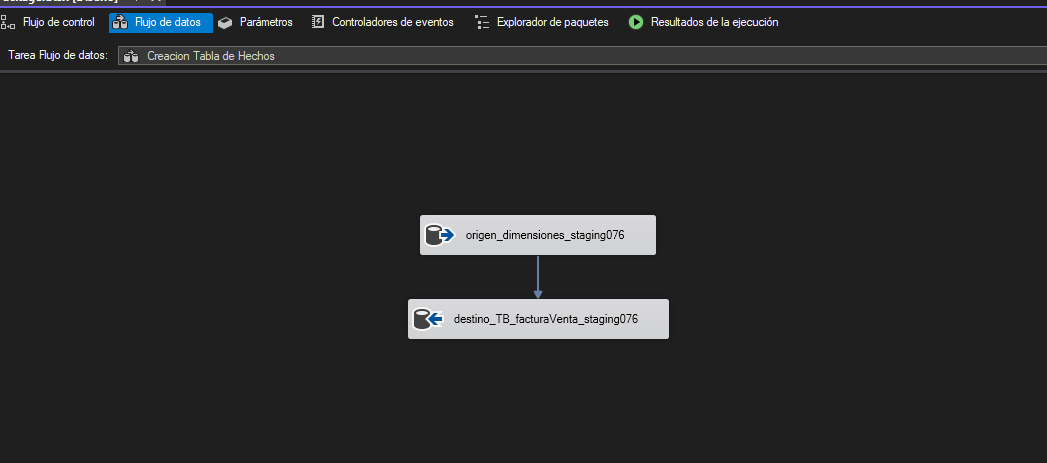


Imagen 5. Creación de tabla de destino y origen para tabla de hecho

Finalmente, luego de corroborar la validación de los datos para cada una de las tablas de la base de datos de staging076, se crea un backup de la base de datos origen, “jardinería” y de la base de datos de staging o de destino que se llama “staging076”:

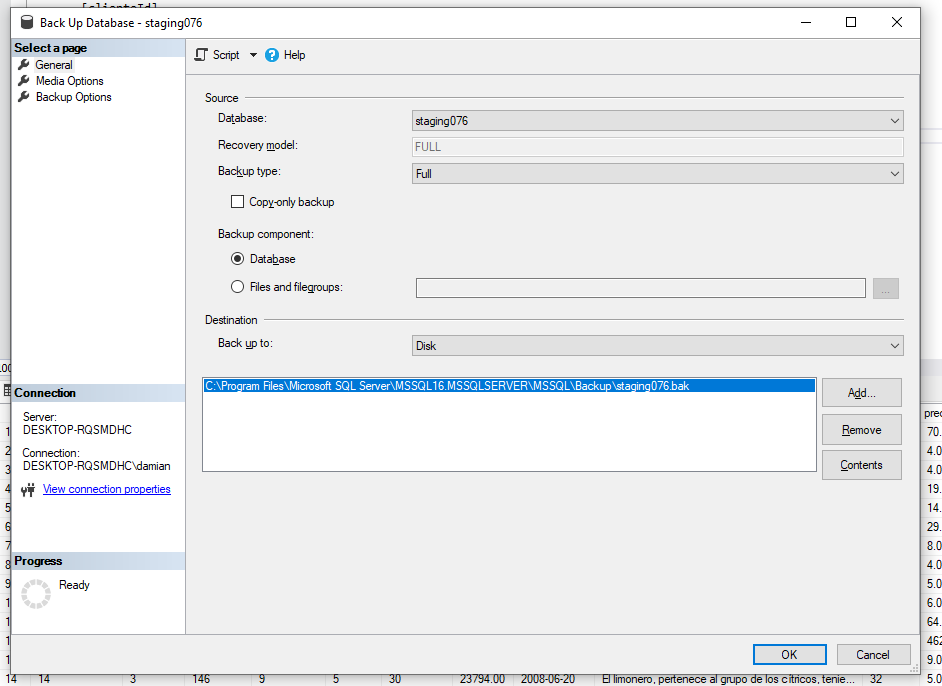


Imagen 6. Creación de Backup para base de datos “staging076”

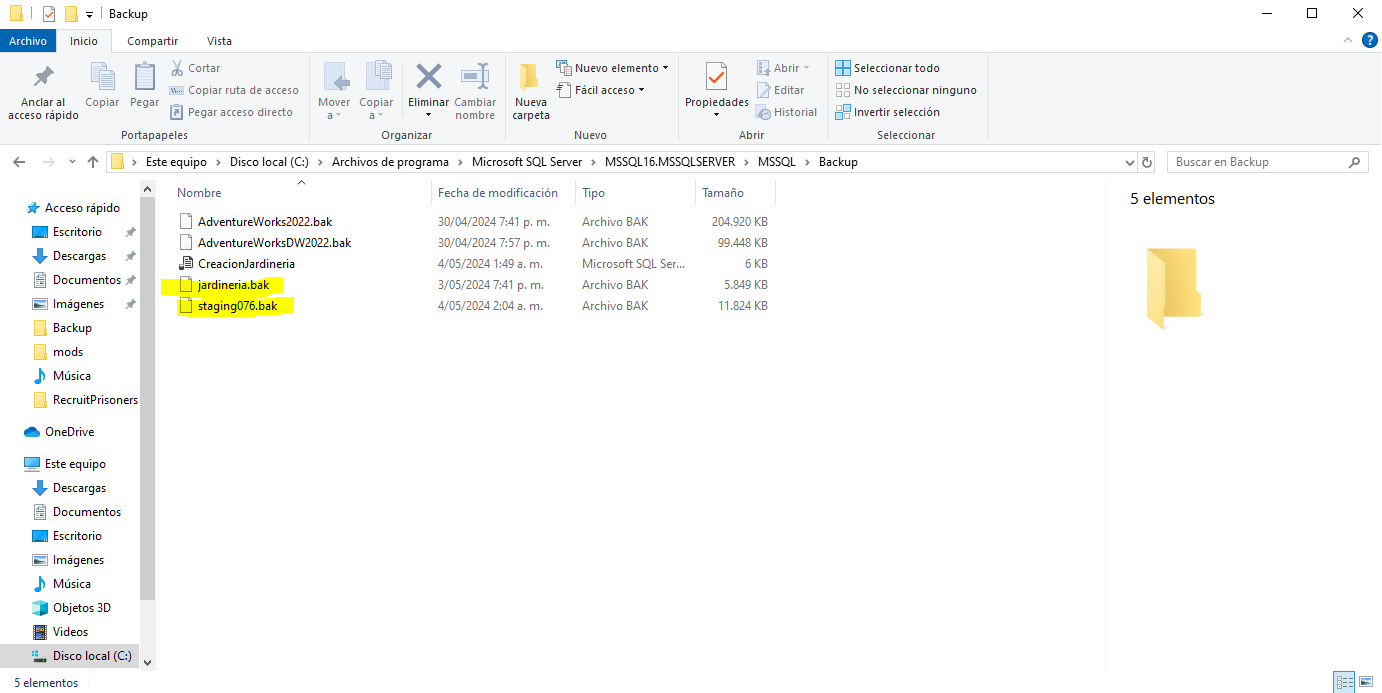


Imagen 7. Verificación de Backups creados

**Scripts**

Los siguientes queries se utilizaron para realizar las consultas necesarias para extraer los datos para ser usados en el modelo estrella. Hay que tener en cuenta que aquí no se encuentra la inserción de datos hacia las tablas ya que dicho proceso se llevo a cabo por medio del Visual Community 2022.

-- Consultas en la BD de jardineria para la creacion de tablas de destino

USE jardineria

-- Seleccion Tabla de Origen Producto

SELECT P.precio\_venta, P.nombre, P.cantidad\_en\_stock, P.dimensiones, P.Categoria, P.descripcion

FROM dbo.producto P

ORDER BY P.nombre ASC

-- Seleccion Tabla de Origen Oficina

Select O.ciudad, O.region, O.pais, O.ID\_oficina

FROM dbo.oficina O

ORDER BY O.ciudad ASC

-- Seleccion Tabla de Origen Cliente

Select C.nombre\_cliente,C.apellido\_contacto as apellido ,C.nombre\_contacto, C.linea\_direccion1 as direccion, C.limite\_credito, C.ID\_empleado\_rep\_ventas

FROM dbo.cliente C

ORDER BY C.nombre\_cliente ASC

-- Seleccion Tabla de Origen Empleado

Select E.nombre, E.apellido1,E.email , E.puesto, E.extension, E.ID\_jefe, E.ID\_oficina

FROM dbo.empleado E

ORDER BY E.nombre ASC

-- Seleccion Tabla de Origen Detalle Pedido

Select PE.ID\_pedido, PE.cantidad, PE.precio\_unidad, PE.ID\_producto

FROM dbo.detalle\_pedido PE

-- Seleccion Tabla de Origen Pedido

Select P.ID\_pedido, P.ID\_cliente, P.fecha\_pedido

FROM dbo.pedido P

-- Seleccion Tabla de Origen Pago

Select P.ID\_cliente, P.total

FROM dbo.pago P

-- Creacion de la BD de destino y creacion de sus respectivas tablas

DROP DATABASE IF EXISTS staging076;

CREATE DATABASE staging076;

USE staging076;

-- Creacion Tabla de pago para DB staging076

CREATE TABLE "destino\_TB\_pago\_staging076" (

"pago\_id" int identity(1,1),

"ID\_cliente" int,

"total" numeric(15,2)

primary key("pago\_id")

)

-- Creacion Tabla de empleado para DB staging076

CREATE TABLE "destino\_TB\_empleado\_staging076" (

"empleadoId" int identity(1,1),

"nombre" nvarchar(50),

"apellido1" nvarchar(50),

"email" nvarchar(100),

"puesto" nvarchar(50),

"extension" nvarchar(10),

"ID\_jefe" int,

"ID\_oficina" int

primary key("empleadoId")

)

-- Creacion Tabla de detalle\_pedido para DB staging076

CREATE TABLE "destino\_TB\_detalle\_pedido\_staging076" (

"detalle\_pedido\_id" int identity(1,1),

"ID\_pedido" int,

"cantidad" int,

"precio\_unidad" numeric(15,2),

"ID\_producto" int

primary key( "detalle\_pedido\_id")

)

-- Creacion Tabla de pedido para DB staging076

CREATE TABLE "destino\_TB\_pedido\_staging076" (

"pedido\_id" int identity(1,1),

"ID\_pedido" int,

"ID\_cliente" int,

"fecha\_pedido" date

primary key("pedido\_id")

)

-- Creacion Tabla de geografia para DB staging076

CREATE TABLE "destino\_TB\_geografia\_staging076" (

"geografia\_id" int identity(1,1),

"ciudad" nvarchar(30),

"region" nvarchar(50),

"pais" nvarchar(50),

"ID\_oficina" int

primary key("geografia\_id")

)

-- Creacion Tabla de producto para DB staging076

CREATE TABLE "destino\_TB\_producto\_staging076" (

"productoId" int identity(1,1),

"precio\_venta" numeric(15,2),

"nombre" nvarchar(70),

"cantidad\_en\_stock" smallint,

"dimensiones" nvarchar(25),

"Categoria" int,

"descripcion" text

primary key("productoId")

)

-- Creacion Tabla de destino Cliente para DB staging076

CREATE TABLE "destino\_TB\_cliente\_staging076" (

"clienteId" int identity(1,1),

"nombre\_cliente" nvarchar(50),

"apellido" nvarchar(30),

"nombre\_contacto" nvarchar(30),

"direccion" nvarchar(50),

"limite\_credito" numeric(15,2),

"ID\_empleado\_rep\_ventas" int

primary key("clienteId")

)

-- Creacion Tabla de hecho para staging076 de Factura de Ventas

CREATE TABLE "destino\_TB\_facturaVenta\_staging076" (

"facturaVenta\_id" int identity(1,1),

"pedido\_id" int,

"productoId" int,

"geografia\_id" int,

"clienteId" int,

"empleadoId" int,

"total" numeric(15,2),

"fecha\_pedido" date,

"descripcion" nvarchar(max),

"cantidad" int,

"precio\_unidad" numeric(15,2)

primary key("facturaVenta\_id")

)

-- Seleccion de datos de diferentes tablas de staging076 para la posterior creacion

-– de la tabla de hechos de factura de venta:

SELECT PE.pedido\_id,P.productoId, G.geografia\_id,C.clienteId,E.empleadoId, PA.total,PE.fecha\_pedido, PR.descripcion, DP.cantidad, DP.precio\_unidad

FROM destino\_TB\_cliente\_staging076 C

INNER JOIN destino\_TB\_pedido\_staging076 PE ON PE.ID\_cliente = C.clienteId

INNER JOIN destino\_TB\_detalle\_pedido\_staging076 DP ON DP.ID\_pedido = PE.ID\_pedido

INNER JOIN destino\_TB\_producto\_staging076 P ON P.productoId = DP.ID\_producto

INNER JOIN destino\_TB\_empleado\_staging076 E ON E.empleadoId = C.ID\_empleado\_rep\_ventas

INNER JOIN destino\_TB\_pago\_staging076 PA ON PA.ID\_cliente = C.clienteId

INNER JOIN destino\_TB\_producto\_staging076 PR ON PR.productoId = DP.ID\_producto

INNER JOIN destino\_TB\_geografia\_staging076 G ON G.ID\_oficina = E.ID\_oficina

En el mundo de la gestión de datos, la transformación y la integración de información de diversas fuentes son tareas fundamentales para la toma de decisiones informadas y la generación de valor. El proceso ETL (Extract, Transform, Load) emerge como un enfoque crucial para lograr este cometido de manera eficiente y efectiva.

**1.Preparación**

Para esta evidencia, continuamos empleando el modelo estrella principal de datos, pero hemos introducido dos cambios significativos o dos novedades:

En la dimensión empleado, hemos suprimido el campo 'jefe' debido a restricciones en la base de datos de jardinería que hacen que no sea posible almacenarlo como un tipo de dato varchar (no existe un campo de tipo varchar en la base de datos de jardinería).

Respecto a la dimensión producto, hemos modificado el tipo de datos del campo 'categoría' de INT a VARCHAR(30). Esta modificación se ha realizado con el objetivo de mejorar el análisis de datos, ya que creemos que contar con el nombre completo de la categoría en lugar de un ID numérico facilita una comprensión más clara y detallada de los resultados (en vez de un id como lo teníamos planeado originalmente).

* Pasando de este modelo estrella:

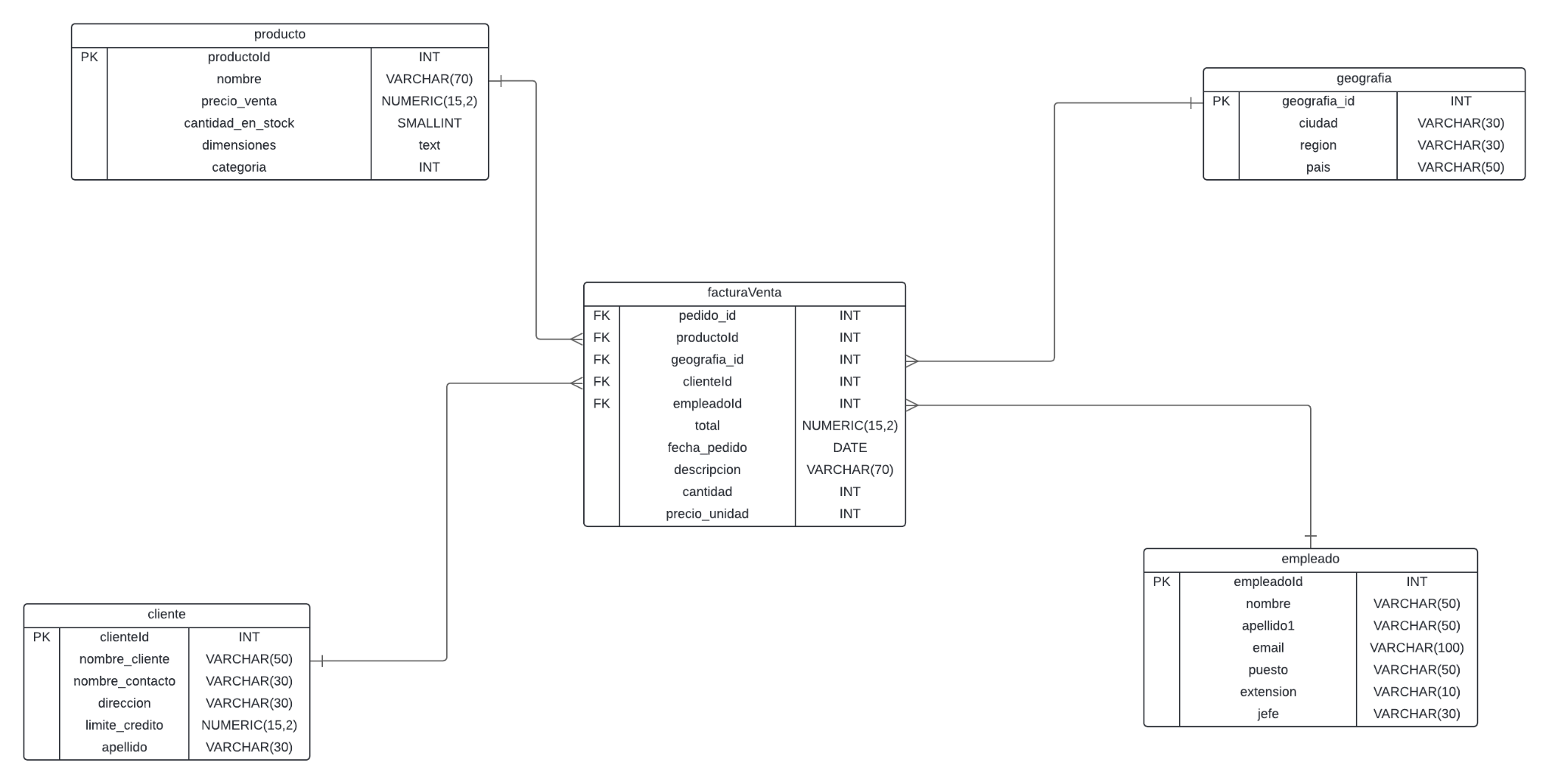


Imagen 1. Modelo estrella de Jardinería

A este:

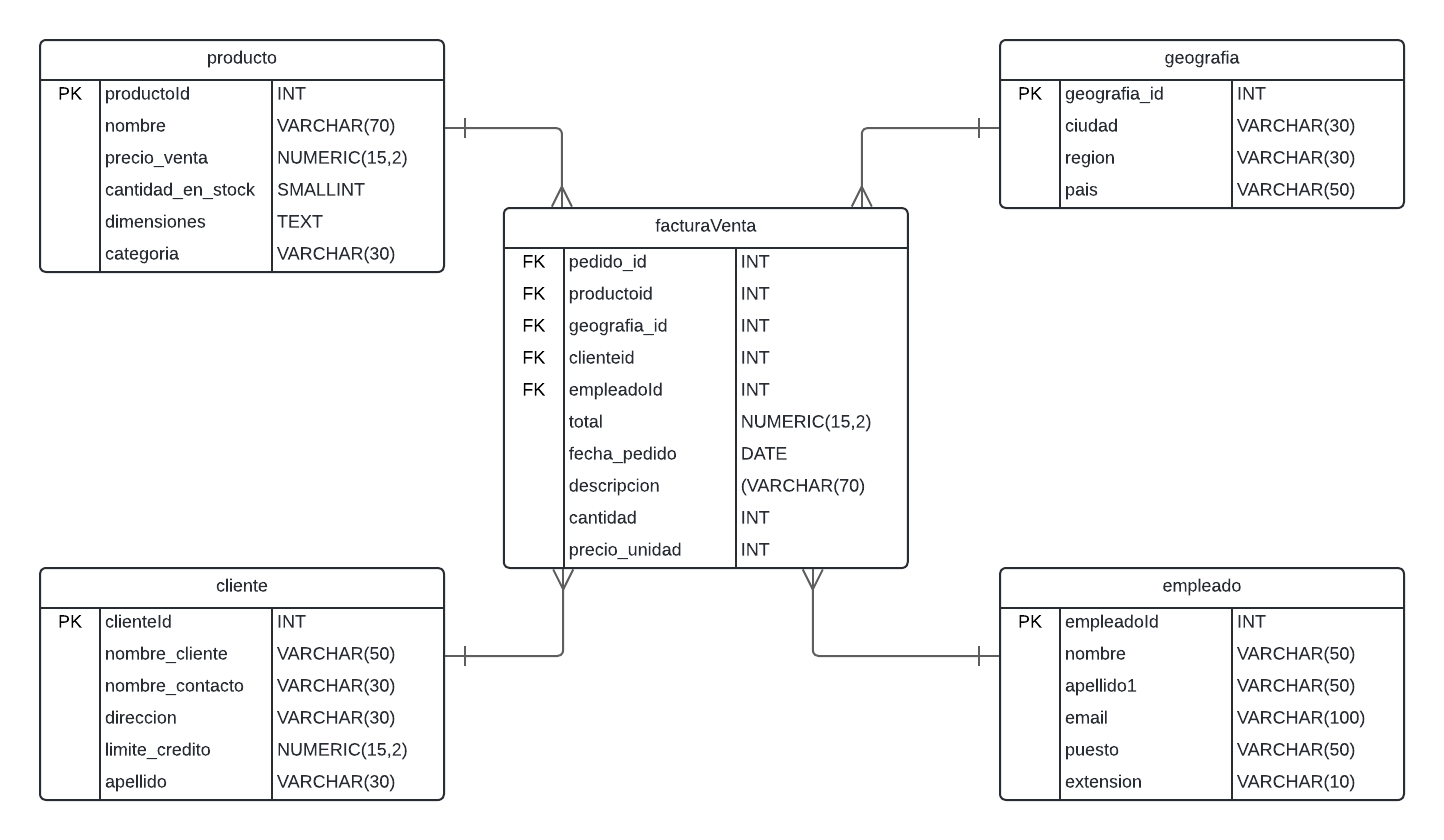


imagen 2. Modelo estrella de Jardinería modificado

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

En cuanto a la extracción de datos, es fundamental llevar a cabo el proceso de verificación para garantizar la integridad y consistencia de los datos extraídos. Esto asegura que los datos cumplan con los requisitos establecidos por el modelo estrella y así proporcionar una base sólida para el Staging.

Para realizar la extracción de la base de datos Jardinería, se utilizaron los resultados de la evidencia dos, la cual fue la creación de una base de datos Staging, donde se crearon consultas SQL para su construcción.

**Scripts:**

-- Consultas en la BD de jardineria para la creacion de tablas de destino

USE jardineria

-- Seleccion Tabla de Origen Producto

SELECT P.ID\_producto,P.precio\_venta, P.nombre,P.cantidad\_en\_stock, P.dimensiones, CP.desc\_categoria as nombre\_categoria,P.descripcion

FROM producto AS P

INNER JOIN Categoria\_producto CP ON P.Categoria = CP.Id\_Categoria

ORDER BY nombre

-- Seleccion Tabla de Origen Oficina

Select O.ciudad, O.region, O.pais, O.ID\_oficina

FROM dbo.oficina O

ORDER BY O.ciudad ASC

-- Seleccion Tabla de Origen Cliente

Select C.nombre\_cliente,C.apellido\_contacto as apellido ,C.nombre\_contacto, C.linea\_direccion1 as direccion, C.limite\_credito, C.ID\_empleado\_rep\_ventas

FROM dbo.cliente C

ORDER BY C.nombre\_cliente ASC

-- Seleccion Tabla de Origen Empleado

Select E.nombre, E.apellido1,E.email , E.puesto, E.extension, E.ID\_jefe, E.ID\_oficina

FROM dbo.empleado E

ORDER BY E.nombre ASC

-- Seleccion Tabla de Origen Detalle Pedido

Select PE.ID\_pedido, PE.cantidad, PE.precio\_unidad, PE.ID\_producto

FROM dbo.detalle\_pedido PE

-- Seleccion Tabla de Origen Pedido

Select P.ID\_pedido, P.ID\_cliente, P.fecha\_pedido

FROM dbo.pedido P

-- Seleccion Tabla de Origen Pago

Select P.ID\_cliente, P.total

FROM dbo.pago P

-- Creacion de la BD de destino y creacion de sus respectivas tablas

DROP DATABASE IF EXISTS staging076;

CREATE DATABASE staging076;

USE staging076;

-- Creacion Tabla de pago para DB staging076

CREATE TABLE "destino\_TB\_pago\_staging076" (

"pago\_id" int identity(1,1),

"ID\_cliente" int,

"total" numeric(15,2)

primary key("pago\_id")

)

-- Creacion Tabla de empleado para DB staging076

CREATE TABLE "destino\_TB\_empleado\_staging076" (

"empleadoId" int identity(1,1),

"nombre" nvarchar(50),

"apellido1" nvarchar(50),

"email" nvarchar(100),

"puesto" nvarchar(50),

"extension" nvarchar(10),

"ID\_jefe" int,

"ID\_oficina" int

primary key("empleadoId")

)

-- Creacion Tabla de detalle\_pedido para DB staging076

CREATE TABLE "destino\_TB\_detalle\_pedido\_staging076" (

"detalle\_pedido\_id" int identity(1,1),

"ID\_pedido" int,

"cantidad" int,

"precio\_unidad" numeric(15,2),

"ID\_producto" int

primary key( "detalle\_pedido\_id")

)

-- Creacion Tabla de pedido para DB staging076

CREATE TABLE "destino\_TB\_pedido\_staging076" (

"pedido\_id" int identity(1,1),

"ID\_pedido" int,

"ID\_cliente" int,

"fecha\_pedido" date

primary key("pedido\_id")

)

-- Creacion Tabla de geografia para DB staging076

CREATE TABLE "destino\_TB\_geografia\_staging076" (

"geografia\_id" int identity(1,1),

"ciudad" nvarchar(30),

"region" nvarchar(50),

"pais" nvarchar(50),

"ID\_oficina" int

primary key("geografia\_id")

)

-- Creacion Tabla de producto para DB staging076

CREATE TABLE "destino\_TB\_producto\_staging076" (

"ID\_producto" int,

"precio\_venta" numeric(15,2),

"nombre" nvarchar(70),

"cantidad\_en\_stock" smallint,

"dimensiones" nvarchar(25),

"nombre\_categoria" nvarchar(50),

"descripcion" nvarchar(max)

)

-- Creacion Tabla de destino Cliente para DB staging076

CREATE TABLE "destino\_TB\_cliente\_staging076" (

"clienteId" int identity(1,1),

"nombre\_cliente" nvarchar(50),

"apellido" nvarchar(30),

"nombre\_contacto" nvarchar(30),

"direccion" nvarchar(50),

"limite\_credito" numeric(15,2),

"ID\_empleado\_rep\_ventas" int

primary key("clienteId")

)

-- Creacion Tabla de hecho para staging076 de Factura de Ventas

CREATE TABLE "destino\_TB\_facturaVenta\_staging076" (

"facturaVenta\_id" int identity(1,1),

"pedido\_id" int,

"productoId" int,

"geografia\_id" int,

"clienteId" int,

"empleadoId" int,

"total" numeric(15,2),

"fecha\_pedido" date,

"descripcion" nvarchar(max),

"cantidad" int,

"precio\_unidad" numeric(15,2)

primary key("facturaVenta\_id")

)

-- Seleccion de datos de diferentes tablas de staging076 para la posterior creacion

-– de la tabla de hechos de factura de venta:

SELECT PE.pedido\_id,P.ID\_producto, G.geografia\_id,C.clienteId,E.empleadoId, PA.total,PE.fecha\_pedido,COALESCE(NULLIF(P.descripcion, ''), 'NA') as descripcion , DP.cantidad, DP.precio\_unidad

FROM destino\_TB\_cliente\_staging076 C

INNER JOIN destino\_TB\_pedido\_staging076 PE ON PE.ID\_cliente = C.clienteId

INNER JOIN destino\_TB\_detalle\_pedido\_staging076 DP ON DP.ID\_pedido = PE.ID\_pedido

INNER JOIN destino\_TB\_producto\_staging076 P ON P.ID\_producto = DP.ID\_producto

INNER JOIN destino\_TB\_empleado\_staging076 E ON E.empleadoId = C.ID\_empleado\_rep\_ventas

INNER JOIN destino\_TB\_pago\_staging076 PA ON PA.ID\_cliente = C.clienteId

INNER JOIN destino\_TB\_geografia\_staging076 G ON G.ID\_oficina = E.ID\_oficina

**3.Transformación de datos según las necesidades analíticas:**

Se aplicaron técnicas de transformación con el objetivo de adecuar los datos a las necesidades específicas. Primeramente se procedió a realizar una limpieza global de la base de datos Staging para eliminar inconsistencias y duplicados, asegurando la calidad y coherencia de los datos. Estas acciones fueron fundamentales para garantizar que los datos estuvieran preparados de manera óptima, esto se realizó mediante el siguiente script:

TRUNCATE TABLE destino\_TB\_geografia\_staging076

TRUNCATE TABLE destino\_TB\_cliente\_staging076

TRUNCATE TABLE destino\_TB\_detalle\_pedido\_staging076

TRUNCATE TABLE destino\_TB\_empleado\_staging076

TRUNCATE TABLE destino\_TB\_pago\_staging076

TRUNCATE TABLE destino\_TB\_pedido\_staging076

TRUNCATE TABLE destino\_TB\_producto\_staging076

TRUNCATE TABLE destinoDimGeografiaJardineria

TRUNCATE TABLE destinoEmpleadoDimJardineria

TRUNCATE TABLE destinoClienteDimJardineria

TRUNCATE TABLE destinoProductoDimJardineria

TRUNCATE TABLE destino\_TB\_hecho\_FacturaVentaJardineria

Dicho script lo agregamos como primer trabajo dentro del flujo de datos:

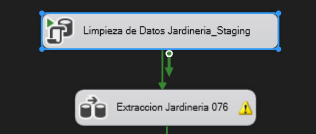


Imagen 2. Creación de la tarea “Limpieza de Datos Jardinería Staging”

Dimensión Geografía:

| Dimensión Geografía (del modelo estrella) | Tabla de Geografía en Staging |
| --- | --- |
|  |  |

Luego de un análisis exhaustivo entre la dimensión esperada de geografía del modelo estrella y la tabla de destino de geografía para staging, nos percatamos de que para poder crear la dimensión necesitamos crear una nueva tabla sin el campo de “ID\_oficina”. Fuera de esto no vimos la necesidad de realizar alguna limpieza en los datos existentes de la tabla de geografía de staging ya que los datos se encuentran en óptimas condiciones.

Por otra parte agregamos la tarea de “Sort” u orden para que los datos ingresados a la dimensión están correctamente ordenados, en nuestro caso seleccionamos un orden ascendente a partir del id de geografía.

Adicionalmente, para lograr evitar la duplicidad de datos al momento de ejecutar continuamente el flujo de transformación de datos dentro del Visual Community 2022, nos fue necesario agregar las tareas de Merge y “Conditional Split”, y de esta manera evitar repetir el ingreso de los mismos datos en la dimensión.

Se utilizó el siguiente script:

select geografia\_id, ciudad, region, pais from destino\_TB\_geografia\_staging076

Creación de flujo de transformación para la creación de la dimensión de Geografía:

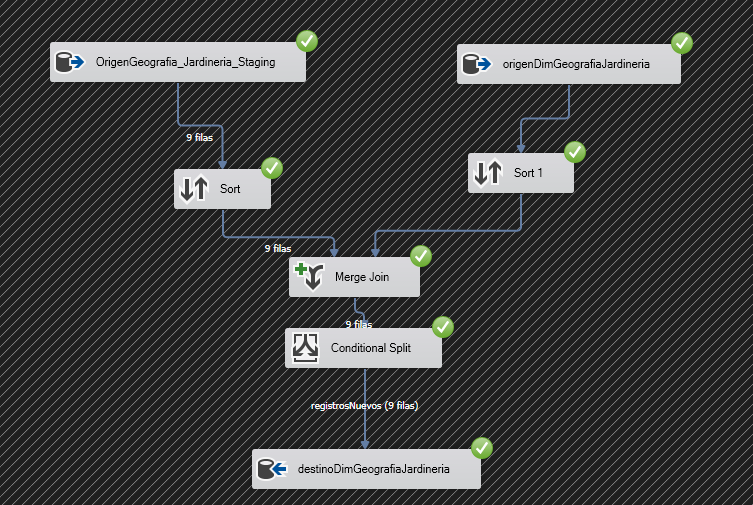


Imagen 3. Creación de la dimensión de Geografía

Observando los resultados de ejecutar dicho flujo de datos, obtenemos los siguientes resultados en la base de datos:

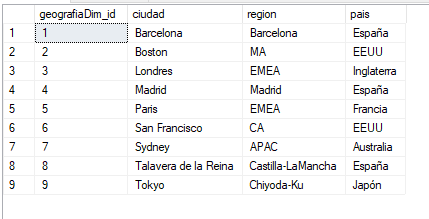


Imagen 4. Resultados de la dimensión de Geografía

Dimensión Empleado:

| Dimensión Empleado | Tabla de Empleado en Staging |
| --- | --- |
|  |  |

Analizando la dimensión empleado en el modelo estrella y la tabla de destino empleado para staging, notamos que era necesario crear una nueva tabla sin el campo ID\_jefe. Al igual que en la dimensión de geografía, en esta también se agregó la tarea “Sort” esto con el fin de que los datos ingresados estén ordenados.

Por otro lado, para evitar que los datos se dupliquen al momento de ejecutar, agregamos las tareas de Merge y “Conditional Split”.

Para la creación de la dimensión de empleado, se realizó la siguiente consulta:

SELECT

empleadoId as empleadoDimId, UPPER(nombre) as nombre,UPPER(apellido1) as apellido1, email, puesto, extension

FROM destino\_TB\_empleado\_staging076

Se realizó exitosamente el siguiente flujo:

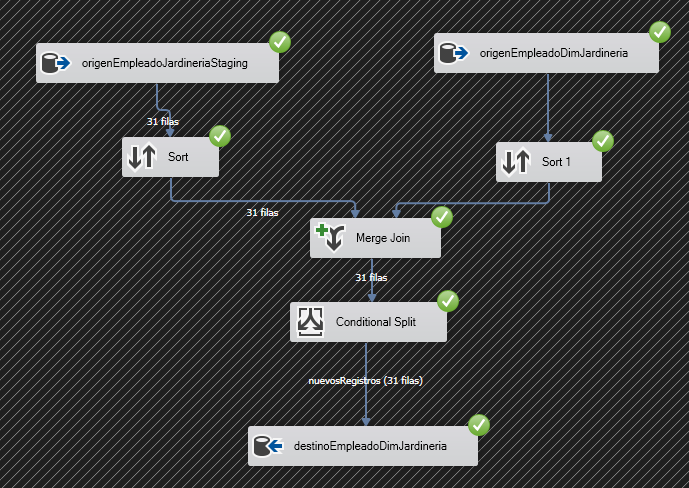


Imagen 5. Creación de la dimensión de Empleado

Observando los resultados de ejecutar dicho flujo de datos, obtenemos los siguientes resultados en la base de datos:

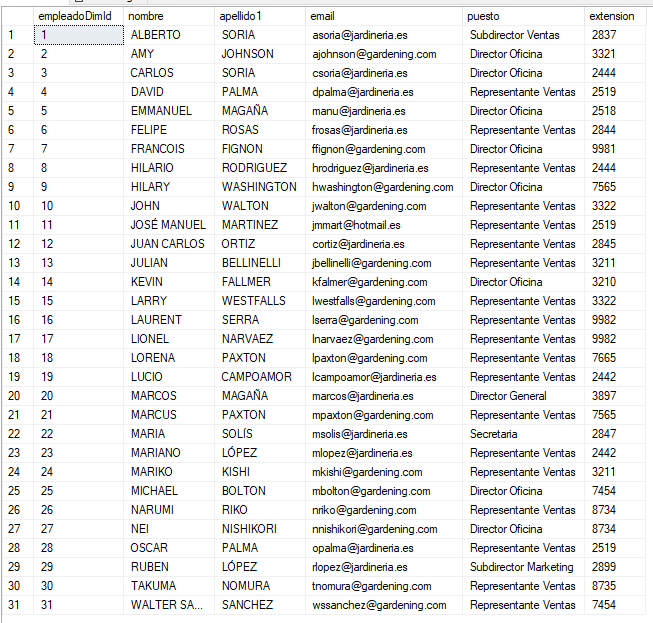


Imagen 6. Resultados de la dimensión de Empleado

Dimension Cliente:

| Dimensión Cliente | Tabla de Cliente en Staging |
| --- | --- |
|  |  |

Analizando la dimensión “Cliente” en el modelo estrella y la tabla de destino producto para staging la tabla permaneció como inicialmente se realizó ya que los campos almacenados en el staging están en el modelo estrella anteriormente construido, por lo que no existió la necesidad de depurar campos, en esta también se agregó la tarea “Sort” esto con el fin de que los datos ingresados estén ordenados.

Por otro lado, para evitar que los datos se dupliquen al momento de ejecutar, agregamos las tareas de Merge y “Conditional Split”.

Se realizó la siguiente consulta:

SELECT clienteId as clienteDimId, UPPER(nombre\_cliente) as nombre, UPPER(nombre\_contacto) as nombre\_contacto,direccion, limite\_credito, UPPER(apellido) as apellido

FROM destino\_TB\_cliente\_staging076

Se realizo exitosamente el siguiente flujo:

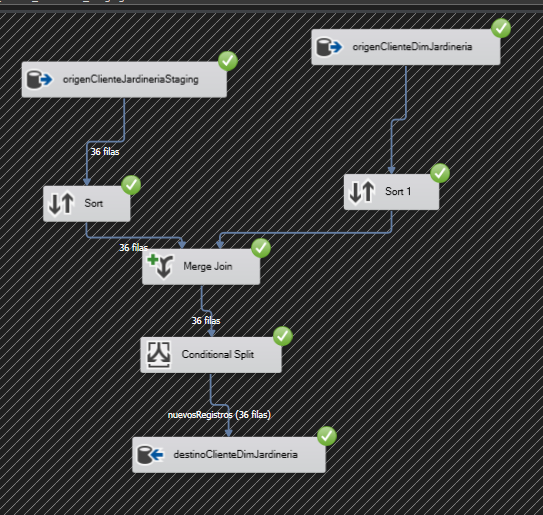


Imagen 7. Creación de la dimensión de Cliente

Observando los resultados de ejecutar dicho flujo de datos, obtenemos los siguientes resultados en la base de datos:

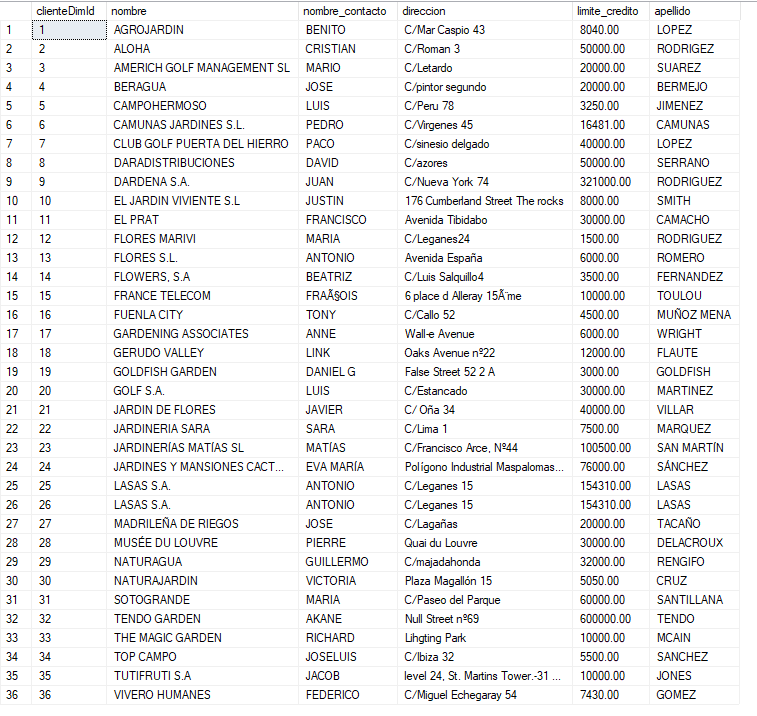


Imagen 8. Resultados de la dimensión de Cliente

Dimensión Producto:

| Dimensión Producto | Tabla Producto en Staging |
| --- | --- |
|  |  |

Analizando la dimensión producto en el modelo estrella y la tabla de destino producto para staging, la tabla permaneció como inicialmente se realizó. Se agregó la tarea “Sort” esto con el fin de que los datos ingresados estén ordenados.

Por otro lado, para evitar que los datos se dupliquen al momento de ejecutar, agregamos las tareas de Merge y “Conditional Split”. Con eso se evita el ingreso de datos repetidos.

Se realizó la siguiente consulta:

SELECT ID\_producto as productoDimId, precio\_venta, UPPER(nombre) as nombre, cantidad\_en\_stock, COALESCE(NULLIF(dimensiones, ''), 'NA') as dimensiones, nombre\_categoria as categoria

FROM destino\_TB\_producto\_staging076

Se realizó exitosamente el siguiente flujo:

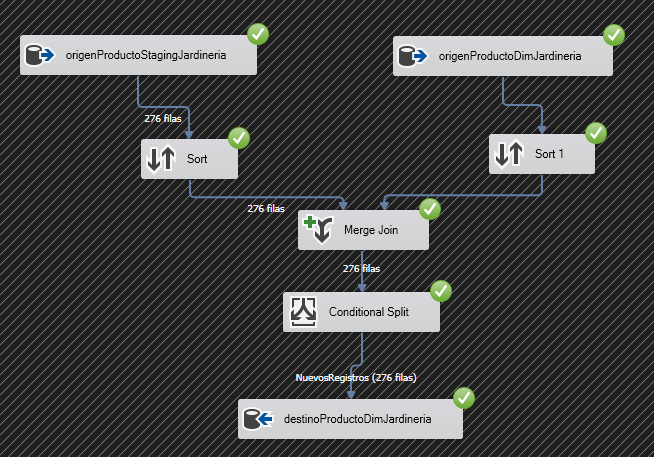


Imagen 9. Creación de la dimensión de Producto

Observando los resultados de ejecutar dicho flujo de datos, obtenemos los siguientes resultados en la base de datos:

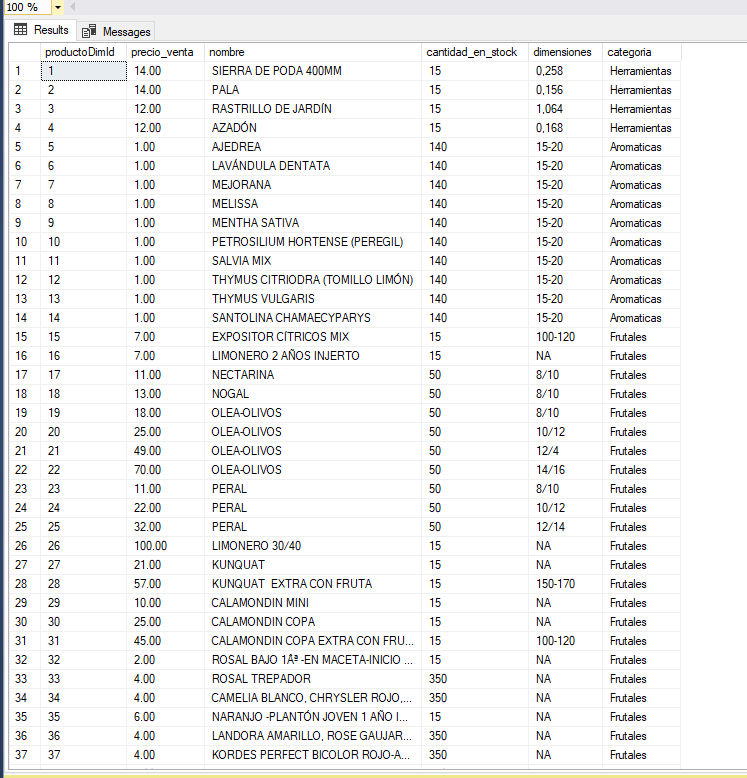


Imagen 10. Resultados de la dimensión de Producto

Tabla de hechos:

| Tabla de Hechos de la Facturas de Venta |
| --- |
|  |

Analizando la tabla de hechos en el modelo estrella y la tabla de hechos para staging la tabla permaneció como inicialmente se realizó ya que los campos almacenados en el staging están en el modelo estrella anteriormente construido, por lo que no existió la necesidad de depurar campos, en esta también se agregó la tarea “Sort” esto con el fin de que los datos ingresados estén ordenados.

Por otro lado, para

evitar que los datos se dupliquen al momento de ejecutar, agregamos las tareas

de Merge y “Conditional Split”.

Para la creación de la tabla de hechos, se utilizó la siguiente consulta:

SELECT PE.pedido\_id,P.ID\_producto, G.geografia\_id,C.clienteId,E.empleadoId, PA.total,PE.fecha\_pedido,COALESCE(NULLIF(P.descripcion, ''), 'NA') as descripcion , DP.cantidad, DP.precio\_unidad

FROM destino\_TB\_cliente\_staging076 C

INNER JOIN destino\_TB\_pedido\_staging076 PE ON PE.ID\_cliente = C.clienteId

INNER JOIN destino\_TB\_detalle\_pedido\_staging076 DP ON DP.ID\_pedido = PE.ID\_pedido

INNER JOIN destino\_TB\_producto\_staging076 P ON P.ID\_producto = DP.ID\_producto

INNER JOIN destino\_TB\_empleado\_staging076 E ON E.empleadoId = C.ID\_empleado\_rep\_ventas

INNER JOIN destino\_TB\_pago\_staging076 PA ON PA.ID\_cliente = C.clienteId

INNER JOIN destino\_TB\_geografia\_staging076 G ON G.ID\_oficina = E.ID\_oficina

Resultado de la transformación:

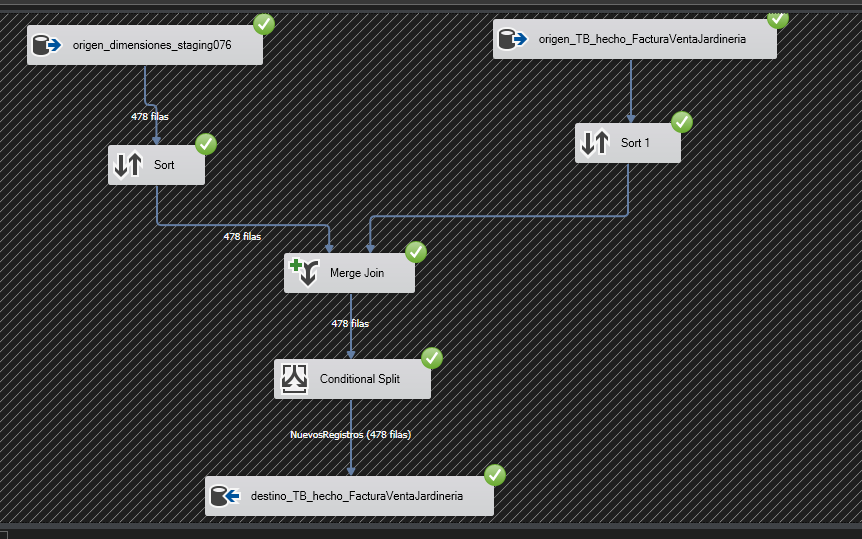


Imagen 11. Creación de la tabla de Hechos de Factura de Ventas

Observando los resultados en la base de datos, obtenemos los siguientes resultados:

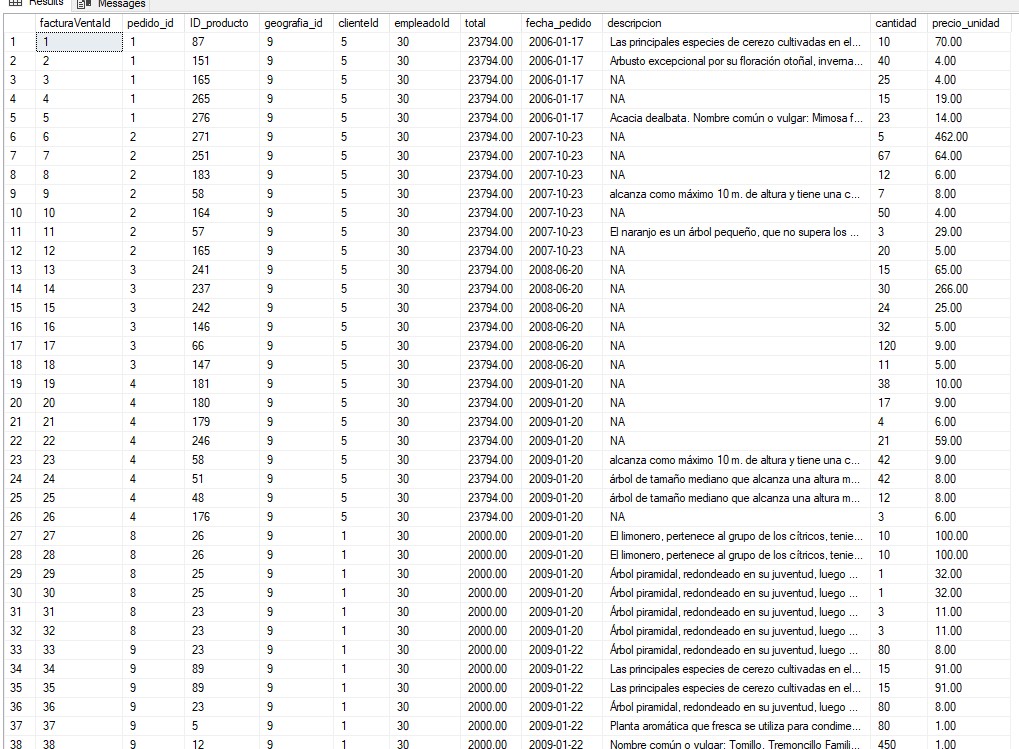


Imagen 12. Resultados de la consulta a la Tabla de Hechos de Facturas de Venta

Resultado del flujo de datos completo:

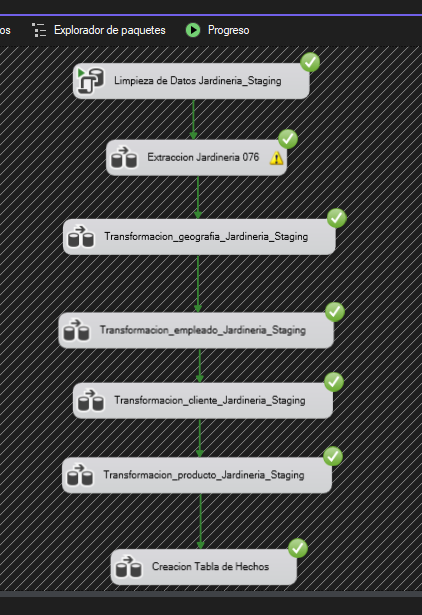


Imagen 13. Ejecución exitosa de todo el flujo de datos. Incluye la extracción y transformación de la base de datos de Jardinería

**Carga de Datos en Data Mart**

Una vez realizada la transformación de los datos, procedimos a realizar la carga de esta transformación al data mart. Primeramente creamos una nueva base de datos llamada ‘DataMartJardineria076’. Luego, procedimos a realizar la carga de cada una de las cuatro dimensiones que planteamos en la primera evidencia de la creación del modelo estrella. Que lo podemos visualizar a continuación:

Dimensión Geografia:

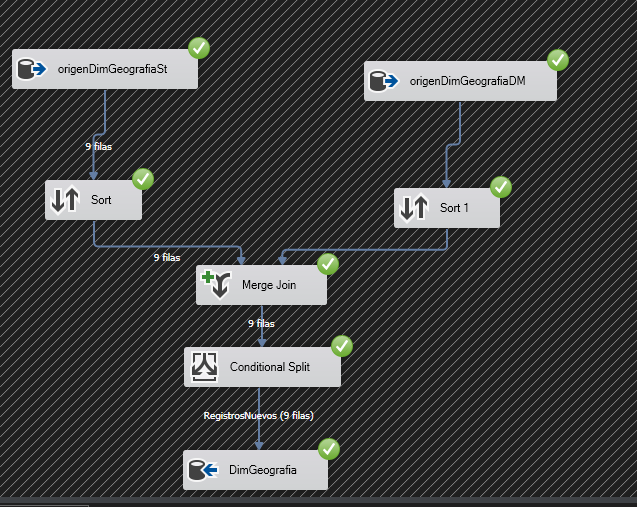


Imagen 14. Flujo de Datos exitoso sobre la dimensión de Geografía

Resultado final de la dimensión:

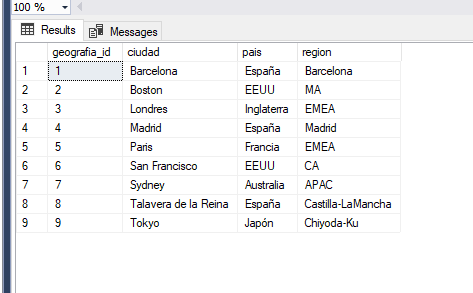


Imagen 15. Valores en la tabla de dimensión de Geografia

Dimension Empleado

Flujo de Datos en Visual

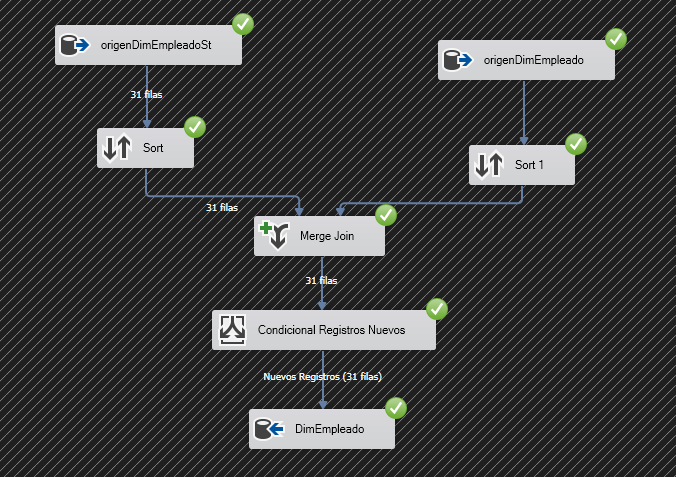


Imagen 16. Creación de la dimensión de Empleado

Resultados en la Tabla:

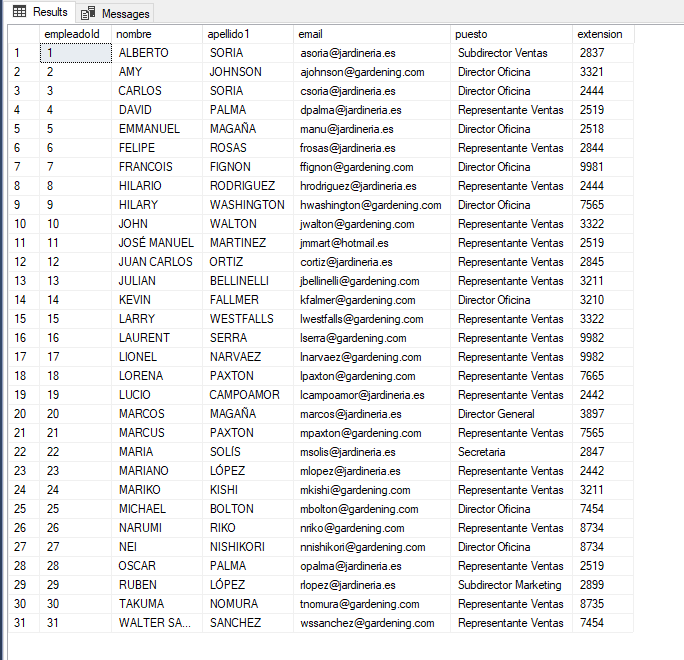


Imagen 17. Resultado de los valores de la dimensión de empleado

Dimension de Cliente

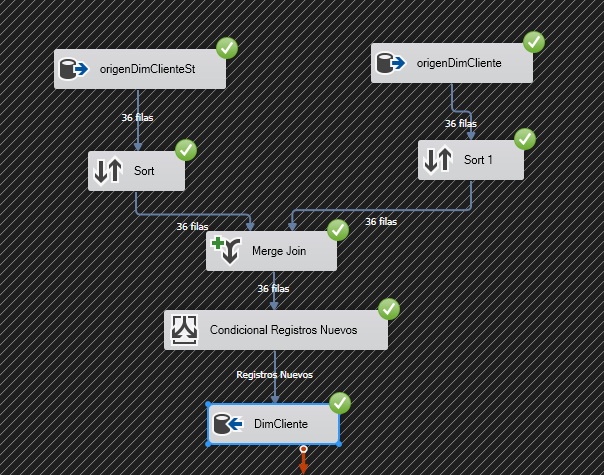


Imagen 18. Creación de la dimensión de Cliente

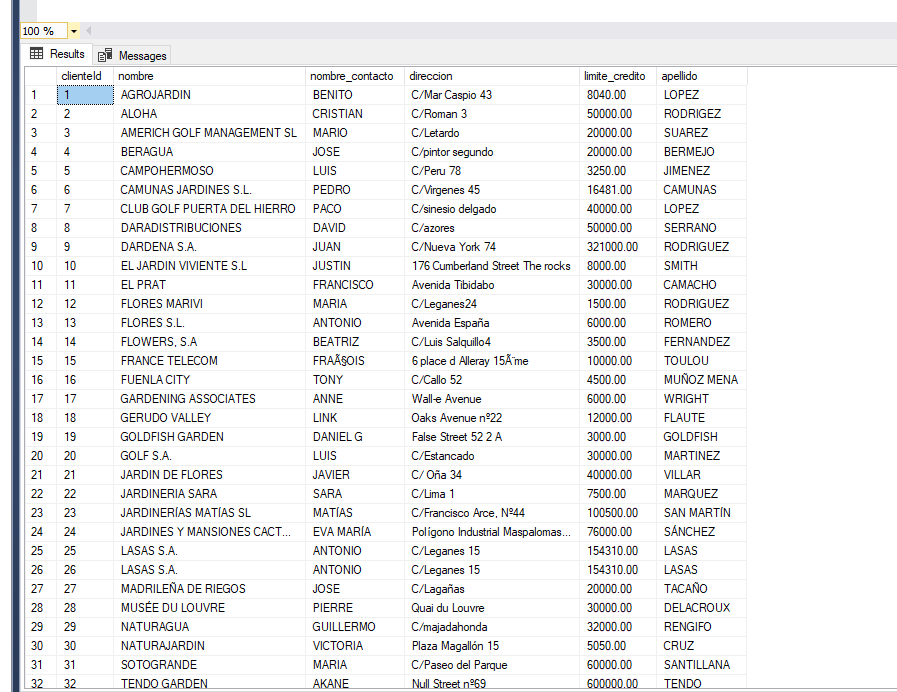
Resultados en la Tabla:

Imagen 19. Valores de la tabla de dimensión de Cliente

Dimensión de Producto

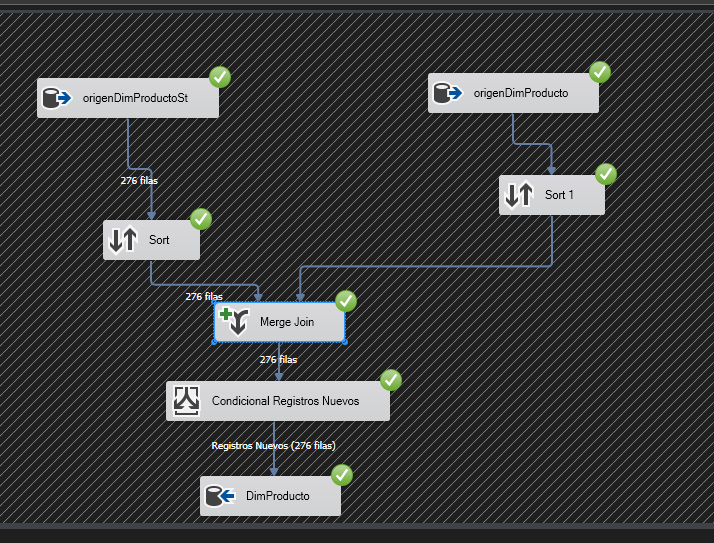


Imagen 20. Creación de Dimensión de Producto

Resultados en la Tabla:

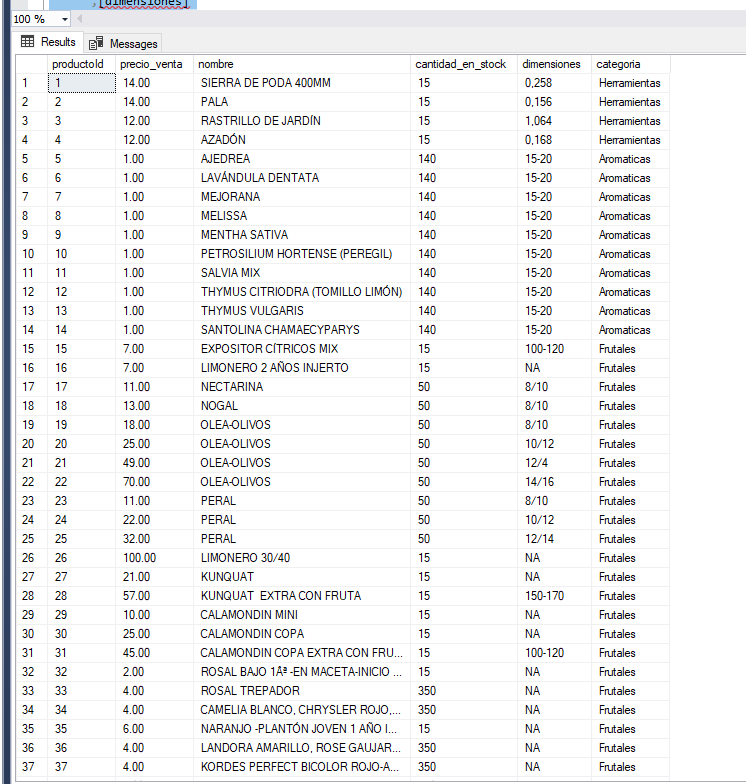


Imagen 21. Valores de la tabla de dimensión de Cliente

Creación Tabla de Hechos Factura de Venta

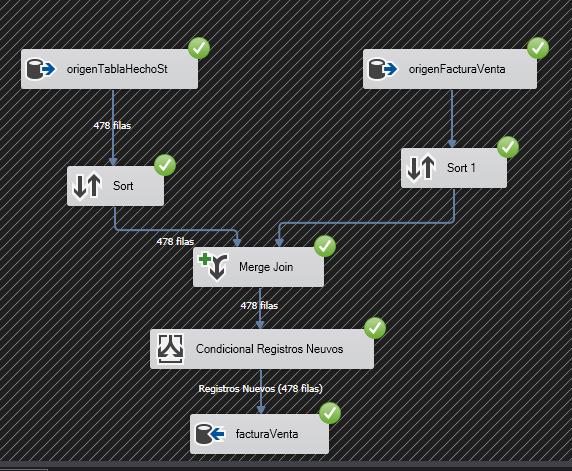


Imagen 21. Creación de Tabla de Hechos de Factura de Venta

Resultados en la Tabla:

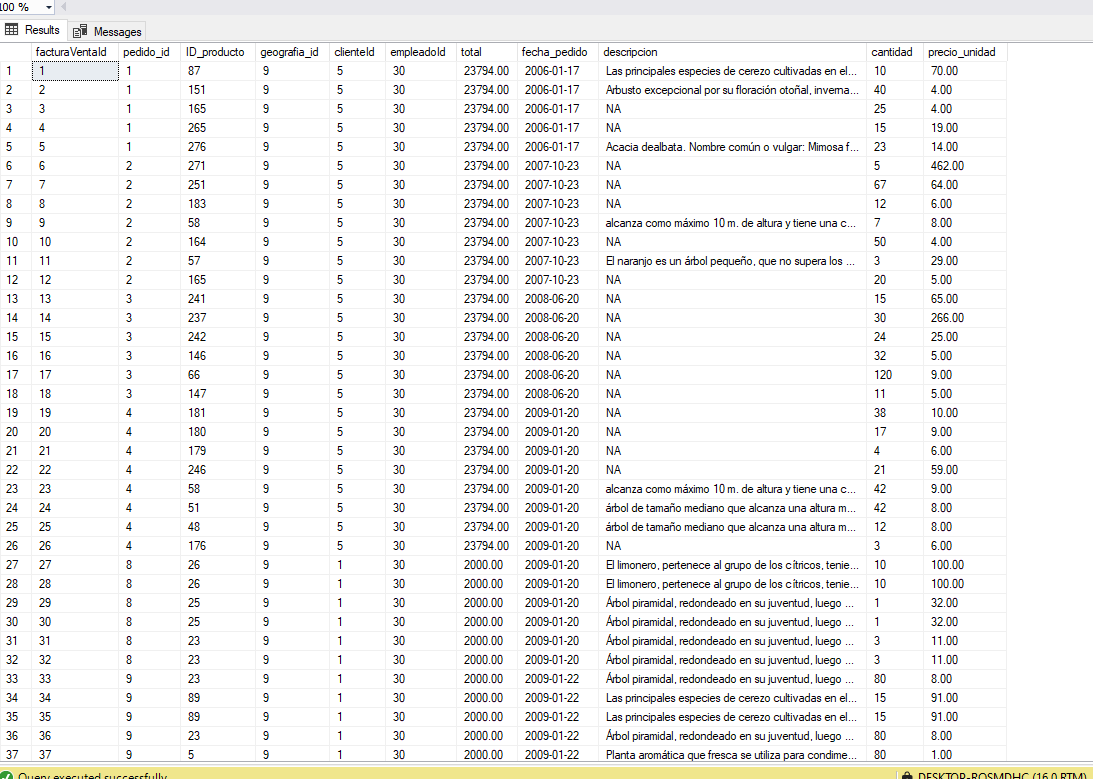


Imagen 22. Resultados de la tabla de hecho de Factura de Ventas

Después de realizar exitosamente la carga de datos en el Data Mart, nos encargamos de realizar las respectivas relaciones entre la tabla de hechos de factura de ventas y las distintas dimensiones como podemos ver a continuación:

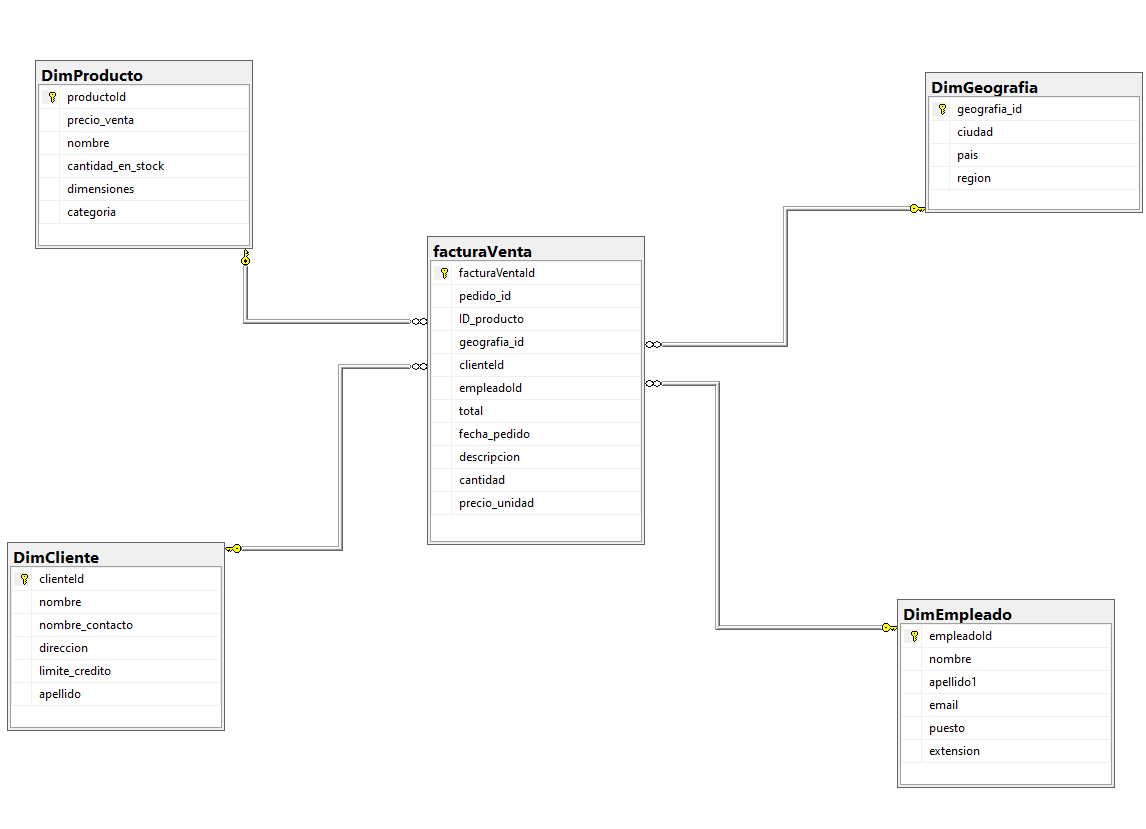
Diagrama del Data Mart

Imagen 23. Diagrama Estrella del data Mart

Finalmente podemos observar como el proceso de ETL se llevó a cabo exitosamente en visual Studio 2022:

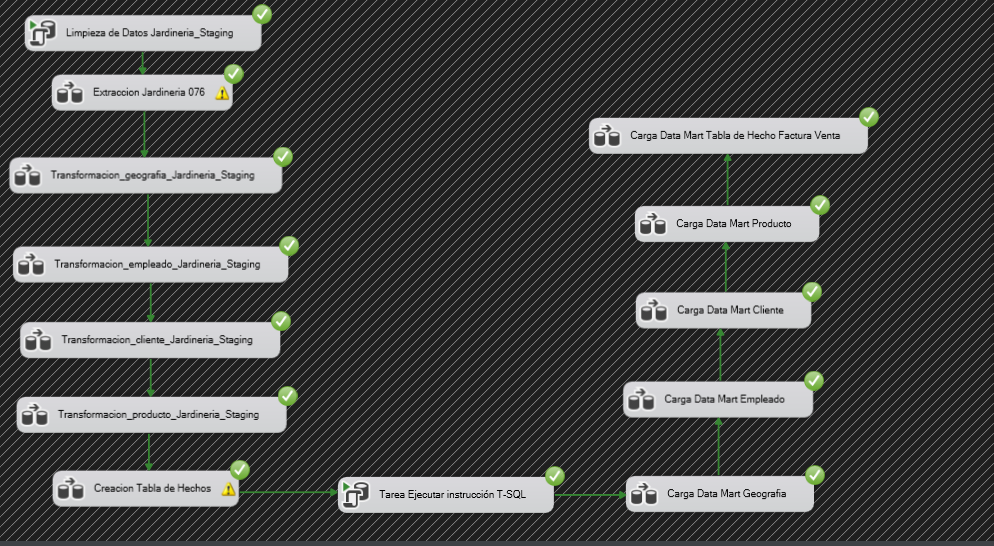


Imagen 24. Flujo de Datos exitoso en Visual Studio

**Verificación de calidad de datos**

Ya con el modelo estrella creado podemos realizar un control de la calidad de los datos

Ejemplo1: Ver todas las compras que se realizaron en la ciudad de “Tokyo”, de aqui queremos obtener, quien hizo las compras, el empleado que asesoro en cada compra, el producto, total, fecha de pedido, descripcion y cantidad:

SELECT FV.facturaVentaId, CONCAT(C.nombre\_contacto,' ', C.apellido) AS nombre\_completo, PR.nombre as producto, CONCAT(EM.nombre,' ', EM.apellido1) as nombre\_empleado,

FV.total, FV.fecha\_pedido, FV.descripcion, FV.cantidad

FROM facturaVenta FV

INNER JOIN DimGeografia G on FV.geografia\_id = G.geografia\_id

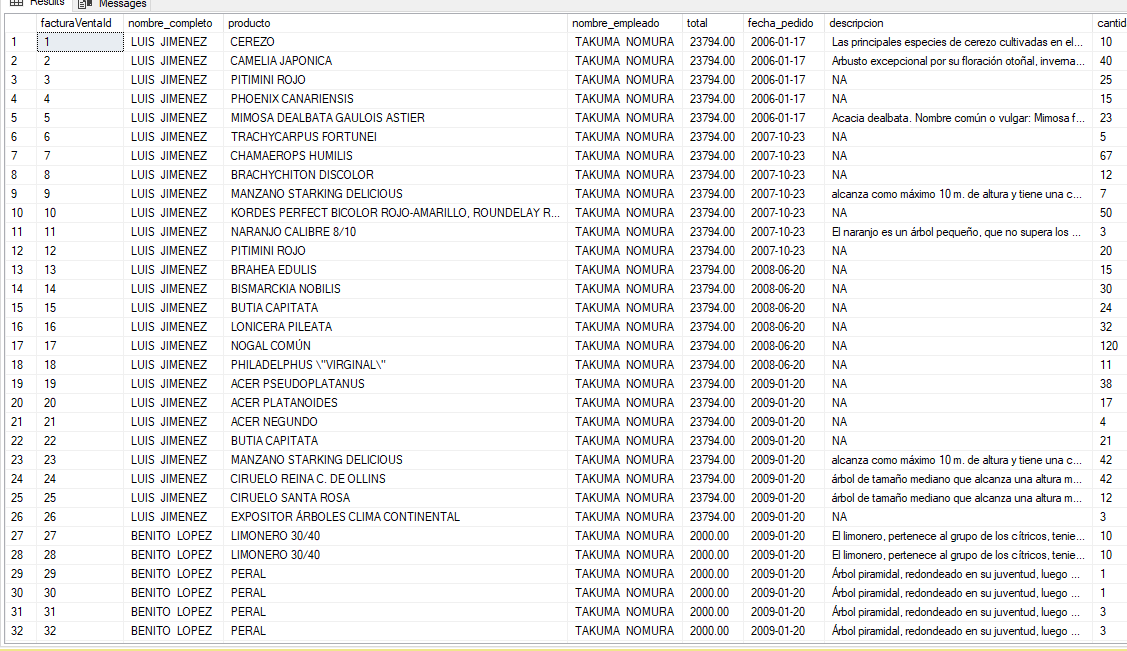
INNER JOIN DimCliente C on FV.clienteId= C.clienteId

INNER JOIN DimProducto PR on FV.ID\_producto= PR.productoId

INNER JOIN DimEmpleado EM on FV.empleadoId= EM.empleadoId

WHERE G.ciudad= 'Tokyo'

Resultado:



**Conclusión**

El proceso ETL (Extract, Transform, Load) en bases de datos se compone de tres actividades fundamentales que son vitales para la gestión efectiva de datos en entornos empresariales y/o inteligencia de negocios:

Extracción (Extract): Durante esta etapa, los datos se obtienen de una variedad de fuentes, incluyendo bases de datos, archivos planos y servicios web. Esta actividad es crucial para reunir datos crudos de múltiples fuentes y prepararlos para su procesamiento posterior.

Transformación (Transform): La transformación de datos implica una serie de operaciones destinadas a limpiar, estructurar y enriquecer los datos extraídos. Esta etapa es fundamental para garantizar la coherencia y la calidad de los datos, así como para prepararlos para su análisis y uso en aplicaciones empresariales.

Carga (Load): Finalmente, los datos transformados se cargan en el destino final, generalmente una base de datos o un almacén de datos. Esta actividad facilita el acceso y la consulta de datos por parte de usuarios y aplicaciones, permitiendo la generación de conocimiento y la toma de decisiones informadas.

**Referencias**

AWS. (s.f.). *¿ Que es un data mart?- Explicación de los data marts- AWS.* Obtenido de https://aws.amazon.com/es/what-is/data-mart/

Zerpa, H., Garcia, R., & Izquierdo, H. (2020). Datamart Basado en el modelo estrella para la implementación de indicadores clave de desempeño como salida del big data.

Base de datos Jardinería Iniciar sesión en Canvas. (s. f.). https://iudigital.instructure.com/courses/15580/files/4737073/download

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/>

AWS. (s.f.). *¿ Que es un data mart?- Explicación de los data marts- AWS.* Obtenido de <https://aws.amazon.com/es/what-is/data-mart/>

Zerpa, H., Garcia, R., & Izquierdo, H. (2020). Datamart Basado en el modelo estrella para la implementación de indicadores clave de desempeño como salida del big data.