CLASE PRACTICA

CARGA DE DIMENSIONES A PARTIR DE LA BASE DE DATOS TRANSACCIONAL

Buenas tardes estudiantes, el objetivo de esta practica es que nos familiaricemos con el procedimiento para crear la carga de las dimensiones. Para ello vamos a usar una base de datos transaccional de ejemplo de la clase (Demo.bak), la cual vamos a restaurar en el servidor de Azure.

Crearemos la base de datos que albergará el modelo estrella, la cual llamaremos DemoDW. En esta base de datos crearemos varias cosas, que son necesarias para la metadata de carga del DW.

1. Crearemos un Esquema **[int]** en la base de datos que hemos creado llamada DemoDW con las respectivas tablas que estaremos usando para llevar registros de las fechas en que se han realizado cargas y su respectiva tabla para el Linaje de la data.

```
CREATE SCHEMA [int]
    AUTHORIZATION [dbo];


--Creamos la tabla que sostiene el detalle de las fechas en que se han hecho
las cargas de datos anteriores.
CREATE TABLE [int].[IncrementalLoads] (
    [LoadDateKey] INT           IDENTITY (1, 1) NOT NULL,
    [TableName]   NVARCHAR (100) NOT NULL,
    [LoadDate]    DATETIME       NOT NULL,
    CONSTRAINT [PK_LoadDates] PRIMARY KEY CLUSTERED ([LoadDateKey] ASC)
);
```

En un proyecto de *data warehouse*, una tabla de *lineage* o trazabilidad permite llevar un control detallado del proceso de carga de datos, registrando cuándo se inició y finalizó la carga de cada tabla, qué tipo de datos se cargaron (hechos o dimensiones), y en qué estado quedó la operación (éxito, fallo, pendiente, etc.). Esta información es clave para detectar errores, validar que los datos están actualizados y garantizar la calidad del sistema. Sin esta tabla, sería difícil saber si una tabla está desactualizada o si una carga falló silenciosamente, lo cual podría llevar a decisiones equivocadas basadas en información incompleta o incorrecta.

```
-- Creamos la tabla para llevar la trazabilidad de los datos

CREATE TABLE [int].[Lineage] (
    [LineageKey]     INT           IDENTITY (1, 1) NOT NULL,
    [TableName]      NVARCHAR (200) NOT NULL,
    [StartLoad]      DATETIME       NOT NULL,
    [FinishLoad]     DATETIME       NULL,
    [LastLoadedDate] DATETIME       NOT NULL,
    [Status]         NVARCHAR (1)   CONSTRAINT [DF_Lineage_Status] DEFAULT (N'P') NOT NULL,
    [Type]           NVARCHAR (1)   CONSTRAINT [DF_Lineage_Type] DEFAULT (N'F') NOT NULL,
    CONSTRAINT [PK_Integration_Lineage] PRIMARY KEY CLUSTERED ([LineageKey] ASC)
);
```

Implementación del Linage

En la primera imagen a continuación tenemos registros de ejemplo de la tabla Linaje, podemos observar que en la imagen siguiente (color azul) tenemos ejemplos de registros cargados en la dimensión, donde hacemos uso del campo [Lineage key] como un atributo más, para determinar en caso de ser necesarios a a que proceso de carga están vinculados esos registros.

Lineage key	Table name	Start	End	Type	Status
10	Dim_Product	2019-03-04 20:39:10.000	2019-03-04 20:54:35.000	F	S
11	Dim_Employee	2019-03-04 20:45:21.000	2019-03-04 20:59:12.000	F	S
...	...	...	...		...
28	Dim_Product	2019-03-15 20:07:10.000	2019-03-15 20:12:33.000	I	S

Product key	Product name	...	Lineage key
1	Cotton candy		10
2	Green tea		10
3	Grape juice		28
4	Banana bread		28

Employee key	Employee name	...	Lineage key
1	Mary Baker		11
2	Jack Peanut		11
3	Gigi Knopper		11
...	...		...

Como parte de la implementación del linaje de datos tenemos este procedimiento almacenado

Get_LineageKey.sql

```
CREATE   PROCEDURE [int].[Get_LineageKey]
@LoadType nvarchar(1),
@TableName nvarchar(100),
@LastLoadedDate datetime
AS
BEGIN
           SET NOCOUNT ON;
           SET XACT_ABORT ON;

       -- La carga para @TableName comienza ahora
       DECLARE @StartLoad datetime = SYSDATETIME();

       /*
       Se inserta una nueva fila en la tabla Lineage, con el nombre de la
       tabla que será cargada,
       la fecha de inicio de la carga, el tipo de carga y el estado de la
       carga.
       Valores posibles para el Tipo:
       - F = Carga completa
```

```
        - I = Carga incremental

        Valores posibles para el Estado:
        - P = En progreso
        - E = Error
        - S = Éxito
        */
        INSERT INTO [int].[Lineage](
                [TableName]
               ,[StartLoad]
               ,[FinishLoad]
               ,[Status]
               ,[Type]
               ,[LastLoadedDate]
               )
        VALUES (
                @TableName
               ,@StartLoad
               ,NULL
               ,'P'
               ,@LoadType
               ,@LastLoadedDate
               );

        -- Si estamos haciendo una carga inicial, elimina la fecha de la carga
        más reciente para esta tabla
        IF (@LoadType = 'F')
               BEGIN
                      UPDATE [int].[IncrementalLoads]
                      SET LoadDate = '1753-01-01'
                      WHERE TableName = @TableName

                      EXEC ('TRUNCATE TABLE ' + @TableName)
               END;

        -- Select the key of the previously inserted row
        SELECT MAX([LineageKey]) AS LineageKey
        FROM [int].[Lineage]
        WHERE
               [TableName] = @TableName
               AND [StartLoad] = @StartLoad

        RETURN 0;
END;
```

Además de las tablas, haremos uso del siguiente procedimiento almacenado que su trabajo es devolvernos la fecha de la ultima carga de una tabla en particular.

Get_LastLoadedDate.sql

```
CREATE   PROCEDURE [int].[Get_LastLoadedDate]
@TableName nvarchar (100)
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

       -- Si la tabla no se encuentra en la estrella, lanzamos un error.
       IF NOT EXISTS (SELECT 1 FROM master.sys.tables
              WHERE name = @TableName AND type = N'U')
       BEGIN
        PRINT N'La tabla no existe en nuestro modelo estrella.';
        THROW 51000, N'La tabla no existe en nuestro modelo estrella.', 1;
        RETURN -1;
       END

--Si la tabla existe, pero nunca ha sido cargada antes, no estará en IncrementalLoads, así
-- agregamos un registro para la tabla que viene como parámetro (@TableName), ------ usando
--la fecha más baja en la columna LoadDate
IF NOT EXISTS (SELECT 1 FROM [int].[IncrementalLoads] WHERE TableName =
@TableName)
       INSERT INTO [int].[IncrementalLoads]
       SELECT @TableName, '1753-01-01'

-- Seleccionamos la fecha de Carga (LoadDate) para el parámetro correspondiente al
-------- parámetro la variable @TableName
       SELECT
              [LoadDate] AS [LoadDate]
    FROM [int].[IncrementalLoads]
    WHERE [TableName] = @TableName;

    RETURN 0;
END;
```

Con estos procedimientos en su lugar (DemoDW), Vamos a empezar nuestra carga del DW por la dimensión Fecha.

2. La Dimensión Fecha

La dimensión Fecha es una dimensión especial y que no depende de ningún origen de datos, así que puede ser cargada de forma independiente. En esta dimensión cada registro representa un día, así que, si cargamos solamente 1 año, tendríamos en esta tabla 366 registros. La dimensión contiene mucha información sobre fecha, pre-calculada, para que las consultas que se hagan que requieran algún cálculo sobre un campo fecha, tengas disponible una columna que ya fue calculada, solamente de incluirse en la consulta, no de calcularla hasta el momento en que se recupera.

Contenido de DimDate.sql

```
CREATE TABLE [dbo].[Dim_Date] (
    [Date Key]                  INT          NOT NULL,
    [Date]                      DATE         NOT NULL,
    [Day]                       TINYINT      NOT NULL,
    [Day Suffix]                CHAR (2)     NOT NULL,
    [Weekday]                   TINYINT      NOT NULL,
    [Weekday Name]              VARCHAR (10) NOT NULL,
    [Weekday Name Short]        CHAR (3)     NOT NULL,
    [Weekday Name FirstLetter] CHAR (1)      NOT NULL,
    [Day Of Year]               SMALLINT     NOT NULL,
    [Week Of Month]             TINYINT      NOT NULL,
    [Week Of Year]              TINYINT      NOT NULL,
    [Month]                     TINYINT      NOT NULL,
    [Month Name]                VARCHAR (10) NOT NULL,
    [Month Name Short]          CHAR (3)     NOT NULL,
    [Month Name FirstLetter]    CHAR (1)     NOT NULL,
    [Quarter]                   TINYINT      NOT NULL,
    [Quarter Name]              VARCHAR (6)  NOT NULL,
    [Year]                      INT          NOT NULL,
    [MMYYYY]                    CHAR (6)     NOT NULL,
    [Month Year]                CHAR (7)     NOT NULL,
    [Is Weekend]                BIT          DEFAULT ((0)) NOT NULL,
    [Is Holiday]                BIT          DEFAULT ((0)) NOT NULL,
    [Holiday Name]              VARCHAR (20) DEFAULT ('') NOT NULL,
    [Special Day]               VARCHAR (20) DEFAULT ('') NOT NULL,
    [First Date Of Year]        DATE         NULL,
    [Last Date Of Year]         DATE         NULL,
    [First Date Of Quater]      DATE         NULL,
    [Last Date Of Quater]       DATE         NULL,
    [First Date Of Month]       DATE         NULL,
    [Last Date Of Month]        DATE         NULL,
    [First Date Of Week]        DATE         NULL,
    [Last Date Of Week]         DATE         NULL,
    [Lineage Key]               INT          NULL,
    PRIMARY KEY CLUSTERED ([Date Key] ASC)
);
```

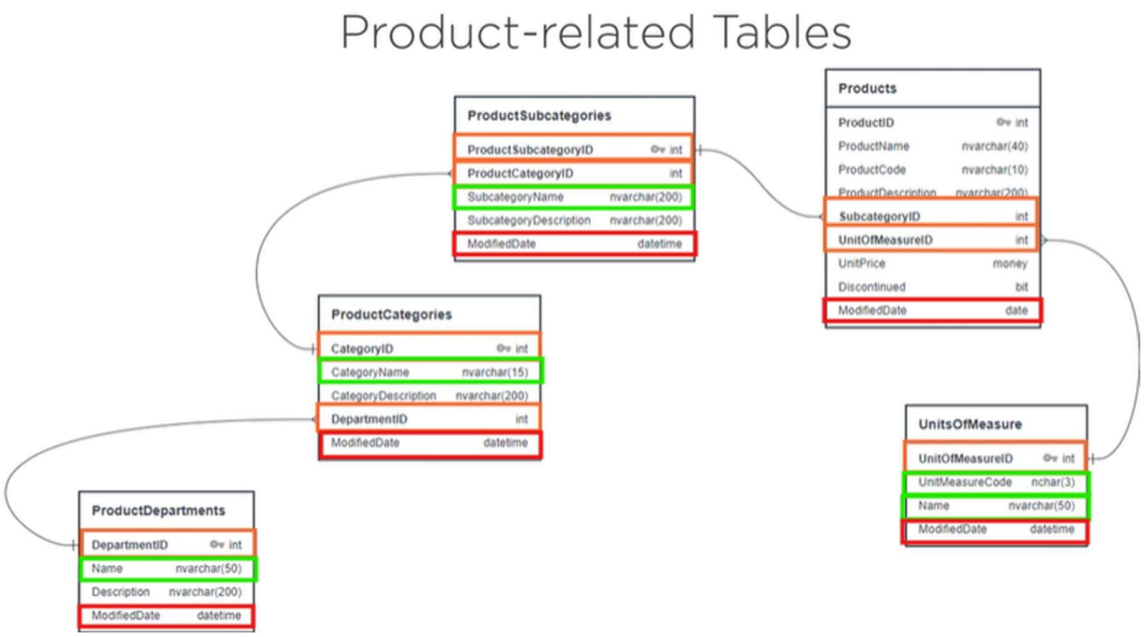
El procedimiento almacenado (Load_DimDate.sql) se encarga de la lógica de los datos que se deben insertar en la dimensión, permitiendo parametrizar un desde y un hasta, haciendo una carga según las necesidades del DW.

Parámetros que podemos cambiar en la cabecera del procedimiento

```
CREATE    PROCEDURE [dbo].[Load_DimDate]
        @StartDate DATE = '2020-01-01',
        @EndDate DATE = '2025-12-31'


EXEC dbo.Load_DimDate
```

3. Dimensión Producto: Jerarquías a desnormalizar en la dimensión producto.

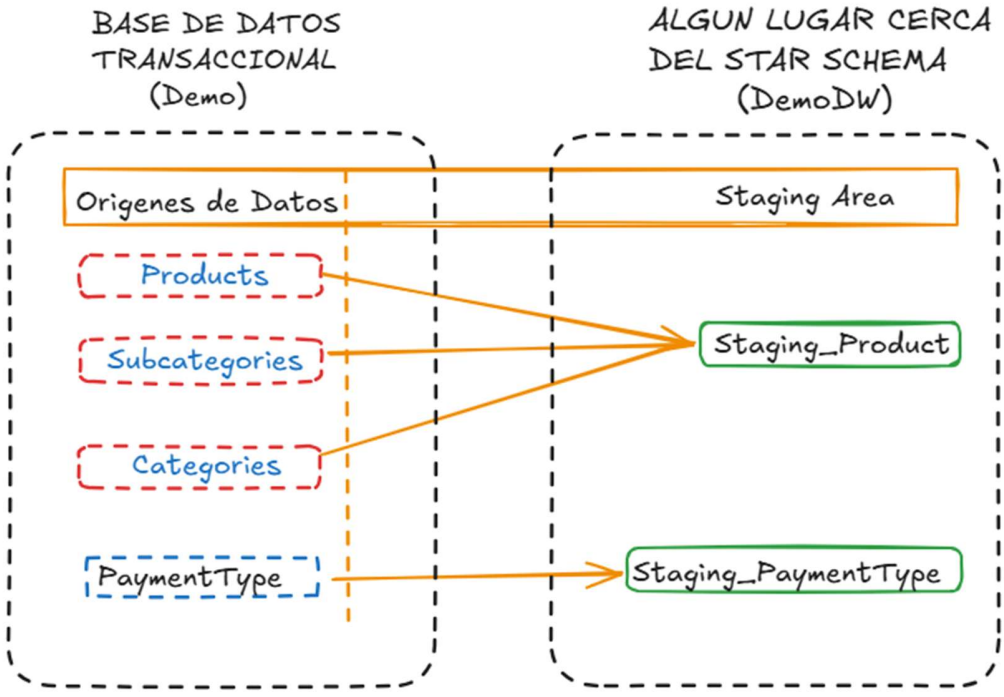


El resultado de la desnormalización, agregando campos propios de la metadata como [Lineage Key] para tener acceso a la información del registro de carga que inserta información en la tabla, así mismo, la llave desde el origen de datos de donde proviene el registro [_Source Key], también las respectivas columnas que dan paso a la gestión de los cambios que puedan ocurrir, conocido como SCD (dimensiones lentamente cambiantes), que serian [Valid From] y [Valid To], la dimensión productos quedaría de la siguiente manera.

Dim_Product
[Product Key]
[_Source Key]
[Product Name]
[Product Code]
[Product Description]
[Product Subcategory]
[Product Category]
[Product Department]
[Unit Of Measure Code]
[Unit Of Measure Name]
[Unit Price]
Discontinued
[Valid From]
[Valid To]
[Lineage Key]

Staging_Product
[Product Key]
[_Source Key]
[Product Name]
[Product Code]
[Product Description]
[Product Subcategory]
[Product Category]
[Product Department]
[Unit Of Measure Code]
[Unit Of Measure Name]
[Unit Price]
Discontinued
[Product Modified Date]
[Subcategory Modified Date]
[Category Modified Date]
[Department Modified Date]
[UM Modified Date]
[Valid From]
[Valid To]

La tabla productos al ser una tabla que no se puede llenar directamente a como lo hicimos con la dimensión Fecha, vamos a simular el área de Staging creando otra tabla en el propio DW, en este caso la llamaremos [Staging_Product], la cual tiene algunas diferencias con la dimensión producto a como se aprecia en la imagen anterior, por ejemplo, no contiene atributo de linaje de datos, pero en cambio contiene fechas de modificación usada para detectar cambios en la dimensión. Por cada dimensión en nuestro DW tendremos una tabla correspondiente en la ***staging_area*** (ubicada en el DemoDW), a como se aprecia en la imagen.



En la misma base de datos donde van a residir nuestras dimensiones (DemoDW) crearemos nuestras tablas de la stagingArea, en este caso primero crearemos la dimensión Productos y su correspondiente tabla de staging.

Dim_Product.sql

```
CREATE TABLE [dbo].[Dim_Product] (
    [Product Key]           INT             IDENTITY (1, 1) NOT NULL,
    [_Source Key]           NVARCHAR (50)   NOT NULL,
    [Product Name]          NVARCHAR (200)  NOT NULL,
    [Product Code]          NVARCHAR (50)   NOT NULL,
    [Product Description]   NVARCHAR (200)  NOT NULL,
    [Product Subcategory]   NVARCHAR (200)  NOT NULL,
    [Product Category]      NVARCHAR (200)  NOT NULL,
    [Product Department]    NVARCHAR (200)  NOT NULL,
    [Unit Of Measure Code] NVARCHAR (10)   NOT NULL,
    [Unit Of Measure Name] NVARCHAR (50)   NOT NULL,
    [Unit Price]            DECIMAL (18, 2) NOT NULL,
    [Discontinued]          NVARCHAR (10)   NOT NULL,
    [Valid From]            DATETIME        NOT NULL,
    [Valid To]              DATETIME        NOT NULL,
    [Lineage Key]           INT             NOT NULL,
    CONSTRAINT [PK_Dim_Product] PRIMARY KEY CLUSTERED ([Product Key] ASC)
);
```

Staging_Product.sql

```
CREATE TABLE [dbo].[Staging_Product] (
    [Product Key]               INT             IDENTITY (1, 1) NOT NULL,
    [_Source Key]               NVARCHAR (50)   NOT NULL,
    [Product Name]              NVARCHAR (200)  NOT NULL,
    [Product Code]              NVARCHAR (50)   NOT NULL,
    [Product Description]       NVARCHAR (200)  NOT NULL,
    [Product Subcategory]       NVARCHAR (200)  NOT NULL,
    [Product Category]          NVARCHAR (200)  NOT NULL,
    [Product Department]        NVARCHAR (200)  NOT NULL,
    [Unit Of Measure Code]      NVARCHAR (10)   NOT NULL,
    [Unit Of Measure Name]      NVARCHAR (50)   NOT NULL,
    [Unit Price]                DECIMAL (18, 2) NOT NULL,
    [Discontinued]              NVARCHAR (10)   NOT NULL,
    [Product Modified Date]     DATETIME        NOT NULL,
    [Subcategory Modified Date] DATETIME        NOT NULL,
    [Category Modified Date]    DATETIME        NOT NULL,
    [Department Modified Date]  DATETIME        NOT NULL,
    [UM Modified Date]          DATETIME        NOT NULL,
    [Valid From]                DATETIME        NOT NULL,
    [Valid To]                  DATETIME        NOT NULL
);
```

Usaremos el procedimiento Almacenado **Load_StagingProduct.sql** para cargar la tabla que acabamos de crear, este procedimiento estará ubicado en la base de datos Demo y recibe dos parámetros, la fecha de la ultima carga, y la fecha de la carga actual.

```
EXEC    [dbo].[Load_StagingProduct]
        @LastLoadDate='2018-01-01',@NewLoadDate='2025-01-01'
```

En este punto empieza a tener sentido, las tablas que hemos definido anteriormente en el esquema [int], como “Incremental loads” además de las columnas para soportar el SCD tipo 2 que hemos elegido, para las dimensiones de nuestro DW. En la imagen podemos ver que un producto ha cambiado el nombre, así que se agrega otro registro para ese producto, se desactiva el registro anterior, estableciendo su validez (columna [valid to]) al día que ocurrió el cambio, y el nuevo registro con el nombre modificado estará activo de aquí en adelante.

_Source key	Name	Valid from	Valid to
387	Cherry toffee	2019-01-01	2019-09-16
387	Super cherry toffee	2019-09-16	9999-12-31
104	Banana bread	2019-01-01	9999-12-31
105	Grape juice	2019-01-01	9999-12-31

Imagen de la tabla que registra las cargas “Incremental Loads”

Load date key	Table name	Load date
1	Dim_Product	2019-04-13
2	Dim_Employee	2019-01-01
...	...	
16	Dim_Product	2019-04-15

Esta información la necesitamos por la siguiente razón, en nuestro ejemplo que vamos a cargar datos a la dimensión productos, por ejemplo, si estamos cargando datos el día que está seleccionado en el recuadro, la última vez para esta fecha que cargamos datos fue el 13 de Abril, cuando los datos se recuperan desde la base de datos de origen, agregaremos una condición a la consulta que solamente traiga los registros recientes, esto lo haremos usando la fecha de modificación del registro, donde la condición establezca que esta fecha sea mayor que la fecha de la última carga (13/04/2019), pero menor o igual que a la fecha en que estamos haciendo nuestra carga que como ejemplo hemos tomado el ultimo registro con Id 16(15/04/2019).

```
SELECT *
FROM Products
WHERE
ModifiedDate > '2019-04-13'
AND ModifiedDate <= '2019-04-15'
```

Para la carga de esta dimensión necesitamos tirar de dos procedimientos almacenados el primero será creado en la base de datos transaccional (Demo) llamado **Load_StagingProduct** cuyo objetivo es mover datos desde la base de datos transaccional hasta el área de Staging de la dimensión Producto.

```
CREATE PROCEDURE [dbo].[Load_StagingProduct]
@LastLoadDate datetime,
@NewLoadDate datetime
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

--SELECT @LastLoadDate, @NewLoadDate

SELECT
     'Demo|' + CONVERT(NVARCHAR, prod.[ProductID])          AS [_SourceKey]
    ,CONVERT(nvarchar(200), prod.[ProductName])             AS [Product Name]
    ,CONVERT(nvarchar(50), prod.[ProductCode])              AS [Product Code]
    ,CONVERT(nvarchar(200), prod.[ProductDescription])      AS [Product Description]
    ,CONVERT(nvarchar(200), subcat.[SubcategoryName])       AS [Subcategory]
    ,CONVERT(nvarchar(200), cat.[CategoryName])             AS [Category]
    ,CONVERT(nvarchar(200), dep.[Name])                     AS [Department]
    ,CONVERT(nvarchar(10), um.[UnitMeasureCode])            AS [Unit of measure Code]
    ,CONVERT(nvarchar(50), um.[Name])                       AS [Unit of measure Name]
    ,CONVERT(decimal(18,2), prod.[UnitPrice])               AS [Unit Price]
    ,CONVERT(nvarchar(10), CASE prod.[Discontinued]
        WHEN 1 THEN 'Yes'
        ELSE 'No'
     END)                                                   AS [Discontinued]
    ,CONVERT(datetime, ISNULL([prod].ModifiedDate, '1753-01-01'))   AS [Product Modified Date]
    ,CONVERT(datetime, ISNULL([subcat].ModifiedDate, '1753-01-01')) AS [Subcategory Modified Date]
    ,CONVERT(datetime, ISNULL([cat].ModifiedDate, '1753-01-01'))    AS [Category Modified Date]
    ,CONVERT(datetime, ISNULL([dep].ModifiedDate, '1753-01-01'))    AS [Department Modified Date]
    ,CONVERT(datetime, ISNULL([um].ModifiedDate, '1753-01-01'))     AS [UM Modified Date]
    ,(SELECT MAX(t) FROM
                          (VALUES
                            ([prod].ModifiedDate)
                          , ([subcat].ModifiedDate)
                          , ([cat].ModifiedDate)
                          , ([dep].ModifiedDate)
                          , ([um].ModifiedDate)
                          ) AS [maxModifiedDate](t)
                        )                                   AS [ValidFrom]
    ,CONVERT(datetime, '9999-12-31')                        AS [ValidTo]

FROM [dbo].[Products] prod
LEFT JOIN [dbo].[ProductSubcategories] subcat ON prod.SubcategoryID = subcat.ProductSubcategoryID
LEFT JOIN [dbo].[ProductCategories] cat ON subcat.ProductCategoryID = cat.CategoryID
LEFT JOIN [dbo].[ProductDepartments] dep ON cat.DepartmentID = dep.DepartmentID
LEFT JOIN [dbo].[UnitsOfMeasure] um ON prod.UnitOfMeasureID = um.UnitOfMeasureID
WHERE
    ([prod].ModifiedDate > @LastLoadDate AND [prod].ModifiedDate <= @NewLoadDate) OR
    ([subcat].ModifiedDate > @LastLoadDate AND [subcat].ModifiedDate <= @NewLoadDate) OR
    ([cat].ModifiedDate > @LastLoadDate AND [cat].ModifiedDate <= @NewLoadDate) OR
    ([dep].ModifiedDate > @LastLoadDate AND [dep].ModifiedDate <= @NewLoadDate) OR
    ([um].ModifiedDate > @LastLoadDate AND [um].ModifiedDate <= @NewLoadDate)

    RETURN 0;
END;
```

Este procedimiento almacenado devuelve el conjunto de datos correspondiente a los productos que se han movido en ese rango de fechas.

```
EXEC [dbo].[Load_StagingProduct]
     @LastLoadDate='2018-01-01',@NewLoadDate='2025-01-01'
```

Results | Messages

_SourceKey	Product Name	Product Code	Product Description	Subcategory	Category	Department	Unit of measure Code	Unit of measure Name	Unit Price	Discontinued
Demo|45	Coffee & Toffee	PROD-0045	Coffee & Toffee	Toffee	Candies	Sweets	PC	Piece	14.00	No
Demo|158	Toffeeness	PROD-0158	Toffeeness	Toffee	Candies	Sweets	PC	Piece	45.60	No

El segundo procedimiento lo crearemos en la base de datos DemoDW llamado Load_DimProduct en la base de datos DemoDW.

Load_DimProduct.sql

```
CREATE PROCEDURE [dbo].[Load_DimProduct]
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime =  '9999-12-31';
    DECLARE @LastDateLoaded datetime;

    BEGIN TRAN;

    -- Obtiene el linaje de la carga actual de Dim_Product
    DECLARE @LineageKey int = (SELECT TOP(1) [LineageKey]
                               FROM int.Lineage
                               WHERE [TableName] = N'Dim_Product'
                               AND [FinishLoad] IS NULL
                               ORDER BY [LineageKey] DESC);

    --Agrega un empty row al proceso de carga, necesario en toda dimension.
    IF NOT EXISTS (SELECT * FROM Dim_Product WHERE [_Source Key] = '')
    INSERT INTO [dbo].[Dim_Product]
           ([_Source Key]
           ,[Product Name]
           ,[Product Code]
           ,[Product Description]
           ,[Product Subcategory]
           ,[Product Category]
           ,[Product Department]
           ,[Unit Of Measure Code]
           ,[Unit Of Measure Name]
           ,[Unit Price]
           ,[Discontinued]
           ,[Valid From]
           ,[Valid To]
           ,[Lineage Key])
     VALUES
           ('', 'N/A', 'N/A','N/A','N/A','N/A','N/A','N/A','N/A', -1, 'N/A', '1753-01-01', '9999-12-31', -1)

    -- Actualiza la fecha de validez de los productos modificados en Dim_Product
    -- Los registros no estaran activos, porque la tabla de stagin sostiene las versiones nuevas.
    UPDATE prod
    SET prod.[Valid To] = mprod.[Valid From]
    FROM
        Dim_Product AS prod INNER JOIN
        Staging_Product AS mprod ON prod.[_Source Key] = mprod.[_Source Key]
    WHERE prod.[Valid To] = @EndOfTime
```

1

2

3

1. Obtiene el linaje que corresponde a la carga actual para la dimensión.
2. Si no existe un registro vacío “Empty Row” entonces lo agrega.
3. Actualiza validez de los productos modificados en Dim_Product, básicamente con la fecha [Valid From] de la tabla de Staging, en otras palabras, marca los registros con cambios como expirados.

```
    -- Insertar nuevos Registros desde la tabla staging de Productos hacia  Dim_Product.
    INSERT Dim_Product
            ([_Source Key]
           ,[Product Name]
           ,[Product Code]
           ,[Product Description]
           ,[Product Subcategory]
           ,[Product Category]
           ,[Product Department]
           ,[Unit Of Measure Code]
           ,[Unit Of Measure Name]
           ,[Unit Price]
           ,[Discontinued]
           ,[Valid From]
           ,[Valid To]
           ,[Lineage Key])
    SELECT [_Source Key]
           ,[Product Name]
           ,[Product Code]
           ,[Product Description]
           ,[Product Subcategory]
           ,[Product Category]
           ,[Product Department]
           ,[Unit Of Measure Code]
           ,[Unit Of Measure Name]
           ,[Unit Price]
           ,[Discontinued]
           ,[Valid From]
           ,[Valid To]
           ,@LineageKey
    FROM Staging_Product;

    --Actualizar la tabla linaje para la carga actual, con la fecha de finalizacion y el estado de la carga
    -- en este caso 'S' para successfully
    UPDATE [int].Lineage
        SET
            FinishLoad = SYSDATETIME(),
            Status = 'S',
            @LastDateLoaded = LastLoadedDate
    WHERE [LineageKey] = @LineageKey;

    --Actualizar la tablas de cargas incrementales con la fecha de la carga actual para Dim_Product
    UPDATE [int].[IncrementalLoads]
        SET [LoadDate] = @LastDateLoaded
    WHERE [TableName] = N'Dim_Product';

    -- Uso de commit para finalizar la transaccion.
    COMMIT;

    RETURN 0;
END;
```

4. Insertamos nuevos registros desde la tabla de staging de productos hacia la dimensión.

5. Actualizamos en la tabla linaje la fecha de finalización de carga y el estado, en este caso ‘S’ es para successfully.

	LineageKey	TableName	StartLoad	FinishLoad	LastLoadedDate	Status	Type
1	1	Dim_Date	2025-05-05 03:53:24.183	2025-05-05 03:53:24.670	2025-05-05 03:53:24.183	S	F
2	5	Dim_Product	2025-05-05 12:05:40.723	2025-05-05 12:11:08.487	2025-05-05 12:05:40.710	S	F

6. Actualizamos la tabla de las cargas incrementales con la fecha de la carga actual para la tabla Dim_Product.

LoadDateKey	TableName	LoadDate
1	Dim_Product	2025-05-05 12:05:40.710

Al final podemos hacer uso de este Script para la carga de la dimensión Dim_Product.

```
DECLARE @LoadType nvarchar(1) = 'F';
DECLARE @TableName NVARCHAR(100)='Dim_Product';
DECLARE @Prev_LastLoaded datetime;
DECLARE @LastLoadedDate datetime;
DECLARE @LineageKey int;

DECLARE @lineage TABLE (lineage int)
DECLARE @lastload TABLE (load_date datetime)

--Establecer la fecha para la carga actual.
SELECT @LastLoadedDate = GETDATE()

--Insertamos un nuevo registro en la tabla Linaje
--Paso 3: Almacenamos la el id del nuevo registro en la variable
@LineageKey para uso futuro
INSERT INTO @lineage EXEC [int].[Get_LineageKey] @LoadType,
@TableName, @LastLoadedDate
SELECT TOP 1 @LineageKey = lineage from @lineage

--Paso 4
--Nos aseguramos que la tabla de staging esta vacía antes de cargar
nueva informacion.
TRUNCATE TABLE Staging_Product

--PASO 5 Recuperamos la fecha de cuando la Dim_Product fue cargada por
última vez.
--PASO 6 Almacenamos esta fecha en @Prev_LastLoadedDate variable
INSERT INTO @lastload EXEC [int].[Get_LastLoadedDate] @TableName
SELECT TOP 1 @Prev_LastLoaded = load_date FROM @lastload

--PASO 7 Cargar en la tabla de Staging los nuevos productos
--o productos que fueron modificados desde la última carga de
Dim_Product.
INSERT INTO [dbo].[Staging_Product]
       EXEC [Demo].dbo.[Load_StagingProduct] @Prev_LastLoaded,
@LastLoadedDate

--PASO 8 Transferir informacion de la tabla de staging a la dimensión
actual Dim_Product
EXEC [dbo].[Load_DimProduct]
```

4. Dimension Location

Dimensión	Objetos a crear en las bases de datos		Tipo
	Demo	DemoDW	
Dim_Location	Load_StagingLocation	Staging_Location Dim_Location Load_DimLocation	Tabla Tabla Procedimiento

```
--Script para la carga de datos a la dimension Location.
DECLARE @LoadType nvarchar(1) = 'F';
DECLARE @TableName NVARCHAR(100)='Dim_Location';
DECLARE @Prev_LastLoaded datetime;
DECLARE @LastLoadedDate datetime;
DECLARE @LineageKey int;

DECLARE @lineage TABLE (lineage int)
DECLARE @lastload TABLE (load_date datetime)

--Establecer la fecha para la carga actual.
SELECT @LastLoadedDate = GETDATE()

--Insertamos un nuevo registro en la tabla Linaje
--Paso 3: Almacenamos el id del nuevo registro en la variable @LineageKey
para uso futuro
INSERT INTO @lineage EXEC [int].[Get_LineageKey] @LoadType, @TableName,
@LastLoadedDate
SELECT TOP 1 @LineageKey = lineage from @lineage

--Paso 4
--Nos aseguramos que la tabla de staging esta vacia antes de cargar nueva
informacion.
TRUNCATE TABLE Staging_Location

--PASO 5 Recuperamos la fecha de cuando la Dim_Product fue cargada por última
vez.
--PASO 6 Almacenamos esta fecha en @Prev_LastLoadedDate variable
INSERT INTO @lastload EXEC [int].[Get_LastLoadedDate] @TableName
SELECT TOP 1 @Prev_LastLoaded = load_date FROM @lastload

--PASO 7 Cargar en la tabla de Staging los nuevos productos
--o productos que fueron modificados desde la última carga de Dim_Location.
INSERT INTO [dbo].[Staging_Location]
       EXEC [Demo].dbo.[Load_StagingLocation] @Prev_LastLoaded,
@LastLoadedDate

--PASO 8 Transferir informacion de la tabla de staging a la dimension actual
Dim_Location
EXEC [dbo].[Load_DimLocation]
```

5. Dimension Customer

Dimensión	Objetos a crear en las bases de datos		Tipo
	Demo	DemoDW	
Dim_Customer	Load_StagingCustomer	Staging_Customer Dim_Customer Load_DimCustomer	Tabla Tabla Procedimiento

6. Dimension Employee

Dimensión	Objetos a crear en las bases de datos		Tipo
	Demo	DemoDW	
Dim_Employee	Load_StagingEmployee	Staging_Employee Dim_Employee Load_DimEmployee	Tabla Tabla Procedimiento

7. Dimension PaymentType

Dimensión	Objetos a crear en las bases de datos		Tipo
	Demo	DemoDW	
Dim_PaymentType	Load_StagingPaymentType	Staging_PaymentType Dim_PaymentType Load_DimPaymentType	Tabla Tabla Procedimiento

8. Dimension Promotion

Dimensión	Objetos a crear en las bases de datos		Tipo
	Demo	DemoDW	
Dim_Promotion	Load_StagingPromotion	Staging_Promotion Dim_Promotion Load_DimPromotion	Tabla Tabla Procedimiento

9. Tabla de Hechos

Hechos	Objetos a crear en las bases de datos		Tipo
	Demo	DemoDW	
Fact_Sales	Load_StagingSales	Staging_Sales Fact_Sales Load_FactSales	Tabla Tabla Procedimiento