Manipulación masiva de datos y operaciones basadas en conjunto

Las operaciones de manipulación masiva de datos abarcan:

INSERT INTO ... SELECT ... FROM: para realizar inserciones masivas basadas en un conjunto de datos de otra tabla;
UPDATE ... FROM: para realizar actualizaciones en lote basadas en un conjunto de datos de otra tabla.
DELETE FROM ... WHERE ...: eliminaciones masivas en función de un conjunto de datos o una condición.

Estas operaciones son fundamentales en procesos de sincronización, importación/integración de datos y ETL (Extracción, Transformación y Carga).

En este apunte se presentan ejemplos uso de sentencias de actualización masiva de datos de productos y marcas a partir de la lista de precios de un proveedor; y una sentencia de borrado de datos de productos no vendidos.

Algunos usos típicos de este tipo de operaciones incluye:

- Actualización de precios
- Depuración de inventario
- Sincronización de clientes

Planteo inicial

Se asume que trabajamos con la base de datos de “Gestión” que venimos utilizando en la materia.

Se asume que se reciben precios de productos provistos por los proveedores, para decidir o no su posterior compra.

Esta información se guarda en una tabla *proveedor_precio* cuya estructura se establece con el siguiente *create:*

```
CREATE TABLE proveedor_precio (
    id BIGINT NOT NULL PRIMARY KEY,      -- ID único para cada registro de la lista de precios
    version INT NOT NULL,                -- Campo para manejar concurrencia y control de versiones
    id_proveedor BIGINT NOT NULL,
    marca VARCHAR(50) NOT NULL,          -- Nombre de la marca del producto
    producto VARCHAR(50) NOT NULL,      -- Descripción del producto
    precio NUMERIC(38, 2) NOT NULL       -- Precio ofertado por el proveedor para este producto
);
ALTER TABLE proveedor_precio ADD CONSTRAINT fk_proveedor FOREIGN KEY (id_proveedor) REFERENCES persona.proveedor(id);
```

Se supone que en esta tabla se carga con información utilizando funciones o procedimientos almacenados desarrollados para tal fin.

Una vez cargada la tabla, necesitamos actualizar nuestros datos de marcas y productos. Si bien podríamos procesar cada fila de la tabla *proveedor_precio* y hacer con la misma lo que corresponda, la idea es explorar una solución de conjunto para todas las filas que compartan un mismo criterio. Esto nos servirá para:

1. Actualizar los precios (o los costos) de los productos que el proveedor propone, que ya tengamos registrados;
2. Insertar las marcas y los productos que el proveedor tiene en su lista, y nosotros no tenemos registrados;

Para plantear un ejemplo con *DELETE* se propone realizar una “limpieza” de los productos que no fueron vendidos nunca, eliminándolos de la tabla de productos. Este ejemplo no tiene que ver con la lista de precios del proveedor.

Scripts de actualización

Actualización de precios existentes

Se actualiza el costo de los productos que tenemos registrados a partir de los precios que tiene la lista de precios del proveedor:

```
UPDATE producto.producto AS p
      SET costo_unitario = pp.precio
      FROM producto.proveedor_precio pp
      JOIN producto.marca m ON pp.marca = m.descripcion
      WHERE p.id_marca = m.id
        AND pp.producto = p.descripcion;
```

Nota: Si bien hay variantes de este *update.. from*, en este caso se lo codifica con las recomendaciones de PostgreSQL, utilizando un alias en la tabla principal (p). PostgreSQL recomienda evitar el uso de la tabla que se está actualizando (producto.producto) en el FROM, ya que esto puede producir efectos inesperados al intentar emparejar todas las filas.

Filtrado: solo las filas de producto.producto que coinciden exactamente con el producto y la marca en proveedor_precio serán las que se actualicen.

Marcas que no tenemos registradas

Detección de filas en la lista del proveedor que tienen marcas que no tenemos registradas:

```
select distinct pp.marca
      FROM producto.proveedor_precio pp
      WHERE NOT EXISTS (
                      SELECT 1
                      FROM producto.marca m
                      WHERE m.descripcion = pp.marca);
```

Nota: La condición de *distinct* se agrega porque pueden ser varios los registros de la lista del proveedor que tengan la misma marca.

Inserción masiva de las filas de la lista del proveedor que tienen marcas que no tenemos registradas:

```
INSERT INTO producto.marca (id, version, codigo, descripcion)
SELECT nextval('producto.producto_sequence'), 1,
nextval('producto.producto_sequence'), pp.marca
FROM producto.proveedor_precio pp
GROUP BY pp.marca
HAVING NOT EXISTS (
    SELECT 1
    FROM producto.marca m
    WHERE m.descripcion = pp.marca
);
```

Nota: El group by y la condición establecida en el having, se establecen porque pueden ser varios los registros de la lista del proveedor que tengan la misma marca.

Productos que no tenemos registrados

Detección de filas en la lista del proveedor que tiene productos que no tenemos registrados:

```
SELECT pp.producto, pp.precio, m.descripcion AS marca

FROM proveedor_precio pp

JOIN producto.marca m ON pp.marca = m.descripcion

WHERE NOT EXISTS (

    SELECT 1

    FROM producto.producto p

    WHERE

        p.descripcion = pp.producto

        AND p.id_marca = m.id);
```

Inserción masiva de productos de la lista del proveedor:

```
INSERT INTO producto.producto (id, version, codigo, descripcion, precio_unitario,
costo_unitario, id_marca)
SELECT nextval('producto.producto_sequence'), 1,
nextval('producto.producto_sequence'), pp.producto, pp.precio, 0, m.id
FROM producto.proveedor_precio pp
JOIN producto.marca m ON pp.marca = m.descripcion  -- Asociamos la marca entre
proveedor_precio y producto.marca
WHERE NOT EXISTS (
    SELECT 1
    FROM producto.producto p
    WHERE
        p.descripcion = pp.producto
        AND p.id_marca = m.id
);
```

Productos que no han sido vendidos

Borrado de productos que no son referenciados desde ningún detalle de factura.

```
DELETE FROM producto.producto AS p
WHERE NOT EXISTS (
    SELECT 1
    FROM venta.factura_detalle AS fd
    WHERE fd.id_producto = p.id
);
```