BASE DE DATOS I

TPE – 2018 Primer Cuatrimestre

Grupo 12:

Della Sala, Rocío 56507

Giorgi, María Florencia 56239

Rodríguez, Ariel Andrés 56030

Santoflaminio, Alejandro 57042

Introducción:

El trabajo práctico nos permitió aplicar los conceptos de SQL avanzado (PSM y Triggers), con el fin de implementar funcionalidades no disponibles de forma estándar.

Para ello fuimos provistos un archivo .csv perteneciente al Gobierno de la Ciudad de Buenos Aires, el cual contiene datos acerca de los recorridos realizados por las bicicletas públicas durante el año 2016. El objetivo era volcar los datos pedidos en una tabla final cumpliendo ciertas condiciones establecidas.

Roles asignados:

Si bien los roles fueron variando y nos ayudamos entre todos para realizar las diferentes tareas, el rol que cada uno cumplió fue:

1. *Della Sala, Rocío:* Encargada del funcionamiento global del proyecto, del estilo y optimización del código y de la importación.
2. *Giorgi, María Florencia:* Encargada de las funciones y de la eficiencia.
3. *Rodríguez, Ariel Andrés:* Encargado del trigger y de las funciones auxiliares.
4. *Santoflaminio, Alejandro:* Encargado de los cursores, de la investigación e informe.

Funcionamiento

Con respecto al funcionamiento global del trabajo, detallaremos como se abarcó tanto cada etapa del trabajo como también cada restricción pedida.

Importación

Se llevó a cabo mediante el comando \copy 'tabla-a-importar' PASTE. En el archivo importacion.sql se detallan claramente los pasos seguidos.

Conversiones de tipos

Se crearon múltiples funciones para convertir los tipos entre las distintas tablas, como TEXT a TIMESTAMP, para llevar el tiempo de uso al formato correcto, calcular el valor del atributo `fecha_hora_dev` a partir de `fecha_hora_ret` y `tiempo_uso`, entre otras.

Tablas

Creamos cuatro tablas en total:

1. `datos_recorrido:` Tabla hacia la cual se importan los datos del .csv.
2. `auxi:` Tabla donde se insertan los datos que fueron filtrados por la condición 1.
3. `auxi2:` Tabla donde se insertan los datos que fueron filtrados por la condición 2.
4. `recorrido_final:` Tabla donde se insertan los datos que fueron filtrados por la condición 3.

Las tablas se encuentran en el archivo importacion.sql.

Restricción 1:

Para cumplir con la restricción 1, decidimos implementar un trigger que realiza múltiples chequeos previos a la inserción.

Primero revisa que cada campo no sea NULL a través de la función `esNULL`. Para todo campo en la tupla a agregar que sea NULL, se imprime un mensaje especificando que campos de la tupla son NULL.

Luego se chequea que el valor de la columna `tiempo_uso` corresponda a una fecha y que además ésta represente un intervalo positivo de tiempo a través de la función `esCorrectoTime`. De no ser una fecha correcta, se especifica a través de un mensaje cual es el error en los datos ingresados en dicho campo.

Finalmente, si no se encontró ningún error en los datos de los campos de la tupla, dicha tupla se agregará a la tabla. De lo contrario, si algún dato estaba mal, se imprimirá un mensaje indicando que tupla es la que no se pudo agregar

Decidimos hacer uso de un trigger, ya que era la manera más eficiente, en cuanto a tiempo y espacio, de cumplir con lo pedido en la restricción 1, ya que no creamos tablas auxiliares y no recorremos la tabla de nuevo. Lo único que hacemos con este trigger es chequear los campos para decidir si dicha tupla debe pertenecer o no a la tabla final.

Restricción 2:

En un primer lugar se había ideado la solución del problema utilizando dos funciones separadas que en total utilizaban tres cursores y generaban la tabla final que cumplía la condición 2. Esta opción terminó siendo muy ineficiente, dándonos tiempos de ejecución altísimos y se optó por cambiarla.

Finalmente, se optó por reemplazar parte de la lógica de cómo se estaba encarando el problema. Mediante un primer cursor, agrupando la tabla `auxi` por id del usuario y fecha de retiro (solo en el caso de que tenga dichos atributos repetidos) se accedió a cada tupla y se llamó a una segunda función. La misma, mediante un segundo cursor, agrega a una tabla aquellas tuplas que son las segundas en cada conjunto. Una vez hecho esto, se creó una tercera función que inserta en esta misma tabla aquellas tuplas que no coincidan con el id y fecha de retiro de las que ya existían allí. De esta forma se evitó volver a usar cursores para la tabla, lo cual resultó en un impacto muy positivo sobre la eficiencia de nuestro trabajo.

Se probó la ejecución hasta esta condición y para las más de 716.000 tablas el vuelco a la nueva tabla `auxi2` fue de 2 minutos.

Restricción 3:

Para la última restricción realizamos lo siguiente:

Primero se buscan todos los usuarios distintos de `auxi2` y con un cursor se los recorre a aquellos que tengan más de una tupla (sino se los agrega directo) y se ejecuta la función problemaSolapados. En esa función se utiliza un cursor para recorrer todas las tuplas correspondientes a tal usuario (ordenadas por fecha de retiro).

Se loopea entre todas esas tuplas y se hace otro loop adentro. El loop interno se fija si para esa tupla con esos valores no hay solapamiento (en tal caso sale y la inserta). Si hubiera solapamiento loopea hasta quedarse con la mayor fecha de devolución.

Es importante destacar que se hace uso de estructuras para ir guardando valores en las diferentes funciones.

La prueba realizada con el archivo .csv completo arrojó que la ejecución no termina (al menos no a un tiempo razonable). A raíz de las diversas pruebas que realizamos, el problema está justamente relacionado con la implementación de la restricción 3. Si aplicamos hasta la condición 2, la ejecución dura alrededor de dos minutos. Con archivos de menor tamaño la ejecución si termina aplicando todas las condiciones. Se probaron diversas alternativas para solucionar el error sin ningún éxito.

Conclusión

El trabajo práctico especial nos resultó útil para afianzar conocimientos sobre SQL Avanzado. Además, trabajar con grandes volúmenes de datos nos pareció que es una buena práctica para aprender la ineficiencia de nuestro código y trabajarlo hasta llevarlo al punto de disminuir el tiempo de ejecución.

Fuentes consultadas

- https://my.vertica.com/docs/7.0.x/HTML/Content/Authoring/SQLReferenceManual/Statements/SET/SETDATESTYLE.htm
- https://www.postgresql.org
- Apuntes de la materia