***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 objetivo del trabajo es aplicar los conceptos de SQL avanzado vistos en clase. Para ello se modelaron funciones en SQL para poder trabajar con un archivo .csv proporcionado por el Gobierno de la Ciudad de Buenos Aires que contiene datos sobre los recorridos realizados por bicicletas públicas durante el año 2016. El objetivo final era volcar los datos pedidos por la consigna en una tabla final cumpliendo ciertas condiciones establecidas.

Para ello se desarrolló el archivo *functions.sql* que contiene todas las tablas necesarias para correr las funciones incluyendo la tabla final *recorrido\_final* como se indica en la consigna. Se intentó realizar siempre las funciones de la manera más eficiente posible, para que la migración de datos sea lo más rápida posible.

A continuación, se detallarán los roles de los miembros del grupo y la explicación de como se aplicaron las restricciones junto con los problemas encontrados para la realización de cada una.

***Roles asignados:***

Para la realización del TP nos dividimos las tareas para facilitarnos el trabajo:

* Della Sala, Rocío: Encargada del funcionamiento global del proyecto, encargada de estilo y optimización de código, encargada de importación.
* Giorgi, María Florencia: Encargada de las funciones, encargada de eficiencia.
* Rodríguez, Ariel Andrés: Encargado del trigger, encargado de funciones auxiliares.
* Santoflaminio, Alejandro: Encargado de cursores, encargado de investigación, encargado del informe.

***Funcionamiento***

En un principio se crean cuatro tablas: *datos\_recorrido* (hacia donde se importan los datos del .csv), *auxi*, *auxi2* y *recorrido\_final*. Los datos de *datos\_recorrido* se pasan a *auxi* y se filtran mediante un trigger, para aplicar la condición 1. Luego en *auxi* se trabajan esos datos y se pasa a otra tabla *auxi2* aplicando la condición2. Finalmente se pasan a *recorrido\_final* aplicando la tercera condición. Se crea un trigger al final para detectar el caso de solapamiento.

***Restricción 1:***

Para cumplir con lo que la consigna pide para la restricción 1, decidimos hacer uso de la implementación de un trigger que funciona de la siguiente manera:

* Primero revisa que cada campo no sea NULL a través de la función isNULL. Para todo campo en la tupla a agregar que sea NULL, se especificó que se imprima 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 isCorrectTime. 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. Esta opción terminó siendo muy ineficiente y se optó por cambiarla. Finalmente, se realizó lo siguiente:

***Restricción 3:***

Completar

***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