Trabajo Práctico Especial Base de Datos I: Informe del proyecto 1C 2023

Fecha de entrega: 15/06/2023

Alumnos:

Camila Sierra Pérez	60242
Martín E. Zahnd	60401
Magdalena Flores Levalle	60077

Docentes:
Leticia Irene Gomez
Cecilia Rodriguez Babino

Grupo 5

Índice

1. Introducción

El objetivo de este Trabajo Práctico Especial es aplicar los conceptos de SQL Avanzado (PSM, Triggers) vistos a lo largo del curso, para implementar funcionalidades y restricciones no disponibles de forma estándar (que no pueden resolverse con Primary Keys, Foreign Keys, etc.).

2. Roles

- **Encargado del informe:** Magdalena Flores Levalle
- **Encargado de las funciones:** Camila Sierra Pérez; Martín E. Zahnd y Magdalena Flores Levalle
- **Encargado del trigger:** Camila Sierra Pérez
- **Encargado del funcionamiento global del proyecto:** Camila Sierra Pérez; Martín E. Zahnd y Magdalena Flores Levalle
- **Encargado de investigación:** Martín E. Zahnd

3. Aspectos que debieron investigarse

En primera instancia, fue necesario investigar cómo se puede realizar la carga de datos, para lo cual decidimos utilizar el comando `\copy`, que explicaremos posteriormente en la sección de *Proceso de importación de los datos*.

También fue necesaria la fórmula para calcular años bisiestos, que presentamos a continuación:

$$bisiesto(n) : n \mod 4 \equiv 0 \wedge (n \mod 100 \not\equiv 0 \vee n \mod 400 \equiv 0) \tag{1}$$

Que escribimos como la SQL function `esBisiesto`:

Listing 1: Función esBisiesto(anio)

```
CREATE OR REPLACE FUNCTION esBisiesto(anio INT) RETURNS BOOLEAN AS $$
    BEGIN
        RETURN anio % 4 = 0 AND (anio % 100 != 0 OR anio % 400 = 0);
    END;
$$ LANGUAGE plpgSQL;
```

Para el formateado de la tabla de la función ReporteConsolidado, buscamos las funciones de padding RPAD y LPAD. Las mismas se utilizan de la siguiente manera:

Listing 2: Función (R,L)PAD

```
[R,L]PAD(string, length[, fill])
```

Siendo que se le pasa la función left padding (LPAD) si se quiere agregar padding del lado izquierdo o right padding (RPAD) para el lado derecho. Los parámetros de la misma son el string deseado, la cantidad de caracteres del padding y el parámetro de fill que es opcional. Este último no se completó ya que las funciones tienen por default el relleno del length con espacios vacíos que era el resultado pretendido.

Además, cómo realizar cursores y triggers fue un tema que debimos repasar, si bien para esto pudimos utilizar los apuntes de la cátedra.

Por último, se investigó el uso del tipo de dato `RECORD` para utilizar en la función ReporteConsolidado(n) con el objetivo de guardar los datos de los cursores y así poder imprimir el output. A su vez, fue necesario indagar cómo utilizar el comando `FETCH INTO` para recuperar el resultado de los cursores creados y asignarlo a la variable creada con el tipo de dato record. También averiguamos sobre el uso de `RAISE EXCEPTION` para lanzar las excepciones pertinentes y `RAISE NOTICE` para poder imprimir el output.

4. Dificultades encontradas

La primera dificultad con la que nos encontramos fue cómo interceptar adecuadamente la inserción de los datos a través de un trigger, para generar las nuevas tablas con los campos deseados. La solución a este problema fue simple: investigamos más en profundidad cómo hacerlo.

Además, no podíamos realizar el COPY a la tabla definitiva dado que la misma debía tener menos campos que el csv de los datos. Para solucionar esto creamos una tabla temporal; en la cual insertamos los datos que luego serán necesarios para poblar las diferentes tablas.

El segundo obstáculo que se nos presentó fue la función `ReporteConsolidado(n)`, en particular, su desarrollo. Su extensión, la necesidad de declarar varias variables y records, y de crear cursores para realizar las consultas pertinentes a cada categoría sumaron a la complejidad de la misma. Brindando mayor detalle con respecto a las variables y records, en un principio los nombres de las variables resultaban ambiguos, motivo por el cual decidimos utilizar el prefijo `r_` para los records y `c_` para los cursores.

Una vez solucionadas las dos dificultades anteriores surgió la última dificultad: mostrar el output de la forma deseada, con el formato y las unidades correspondientes. Para obtener un correcto formato, optamos por aplicar un *padding* con las funciones `RPAD` y `LPAD` para acomodar los valores en sus columnas de manera que sea más legible para el usuario.

5. Proceso de importación de los datos

En primera instancia, para poder importar los datos, fue necesario crear una serie de tablas para contener los mismos, primero las tablas de las dimensiones: `ESTADO` , `ANIO` y `NIVEL_EDUCACION` y luego la tabla definitiva: `NACIMIENTOS` . A su vez, decidimos hacer uso de una tabla temporal `TABLA_TEMPORAL_NACIMIENTOS` para copiar la totalidad de los datos del csv. Los datos para la adecuada creación y población de estas tablas fueron brindados por la cátedra a través de un archivo en formato CSV. Para realizar el copiado de datos en su tabla correspondiente se utilizó el ya mencionado comando `COPY`, cuya estructura está dada por

Listing 3: Función COPY

```
COPY table_name [ ( column_name [, ...] ) ]
    FROM { 'filename' | PROGRAM 'command' | STDIN }
    [ [ WITH ] ( option [, ...] ) ]
    [ WHERE condition ]


Donde option puede ser
    FORMAT format_name
    FREEZE [ boolean ]
    DELIMITER 'delimiter_character'
    NULL 'null_string'
    HEADER [ boolean | MATCH ]
    QUOTE 'quote_character'
    ESCAPE 'escape_character'
```

```
    FORCE_QUOTE { ( column_name [, ...] ) | * }
    FORCE_NOT_NULL ( column_name [, ...] )
    FORCE_NULL ( column_name [, ...] )
    ENCODING 'encoding_name'
```

En nuestro caso particular, las *options* que utilizamos fueron `FORMAT` , `DELIMITER` , y `HEADER` , para establecer el formato de archivo CSV , el caracter `,` (coma) como separador e indicar que el archivo contiene una línea que actúa como encabezado y debe ser ignorada al momento de copiar los datos.

Para poblar las distintas tablas de las dimensiones y la tabla definitiva, creamos el trigger *insertarDatos()*, el cual realiza dicha acción antes de que se inserten los datos en la tabla temporal con el comando `COPY`.

6. Ejecución del programa

Para ejecutar el programa correctamente se pueden seguir las siguientes instrucciones:

6.1. Requisitos previos

Para poder ejecutar el programa, debe contar con los archivos `us_births_2016_2021.csv`, brindado por la cátedra, y el archivo `funciones.sql`.

6.2. Ejecución

1. Desde una terminal, deben enviarse los archivos mencionados anteriormente a Pampero utilizando el comando:

```
scp funciones.sql us_births_2016_2021.csv {user}<at>pampero.itba.edu.ar:/home
    /{user}/
```

   Donde `<at>` debe ser reemplazado por el símbolo `@` y `{user}` por su nombre de usuario.

2. Luego, en otra terminal, conectarse a pampero (al servidor de la BD de la catedra):

```
ssh {user}<at>pampero.itba.edu.ar -L 5432:bd1.it.itba.edu.ar:5432
```

3. Una vez conectado, debe crear las tablas, los triggers y las funciones en la base de datos:

```
psql -h bd1.it.itba.edu.ar -U {user} -f funciones.sql PROOF
```

4. Por último, debe conectarse a la DB de itba utilizando el siguiente comando:

```
psql -h bd1.it.itba.edu.ar -U {user} PROOF
```

El programa se encuentra ahora listo para ser usado, puede probar las funcionalidades ejecutando la siguiente función:

```
SELECT ReporteConsolidado(cantidad_deseada);
```