PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

TALLER DE BASES DE DATOS

GABRIEL REYES
JEAN RODRÍGUEZ

INFORME DE TALLER II

OCTUBRE, 2019

Índice

Listings

1. Introducción

Los modelos de negocio actuales manejan cantidades enormes de datos de los clientes en cada empresa, sea grande o pequeña, lo que evidencia una clara necesidad de un software capaz de procesar las solicitudes requeridas. Cuando una entidad desea minimizar la complejidad de sus operaciones y mantener tiempos de respuesta aceptables, se debe dejar en claro un diseño simple y eficiente que sea mantenible y facilite la realización de cambios a futuro.

Para administrar la información del usuario, se propone un modelo entidad-relación que cumple con los estándares de normalización y se entrega el script para representarlo en una base de datos. Además, se mostrará el cómo se hace el intercambio de información entre un archivo de texto separado por comas (archivo de formato .csv) con el sistema de gestión de base de datos PostgreSQL a través de un script (.sql), mediante la función COPY.

2. Preparativos

Antes de comenzar, es necesario aclarar que el archivo inicial fue trabajado en el entorno de Microsoft Excel 2019, el cual permite exportar las hojas de cálculo al formato Csv, para poder importarlo posteriormente en la base de datos. El formato del archivo adjunto posee la codificación en formato UTF-8, al igual que la base de datos que trabajo el script adjunto.

Posteriormente, se modificaron algunos datos de las tablas del csv: números sin separador mil (.) y decimales con comas (,); estados civiles con casillas vacías reemplazadas por el estado "soltero", pues la ley no contempla que un individuo no posea estado civil alguno; corrección manual de celdas que no pertenecían a ninguna tupla; corrección manual del archivo csv en el editor de textos, ya que se generaban delimitadores (;) sobrantes al final del archivo.

3. Creación de Tablas

A continuación se presenta el script en el lenguaje PostgreSQL con sus restricciones aplicadas para el adecuado funcionamiento en la base de datos. Se utilizan una serie de enumeraciones que indican los posibles estados de cada tipo de dato con múltiples valores específicos, con el fin de facilitar la integración con distintos sistemas, limitar las opciones que el cliente necesite evaluar y, en general, mejorar el rendimiento en las consultas.

```
1 CREATE TYPE RangoEtario AS ENUM ('menor␣que␣30', 'entre␣30␣y␣40',
      'entre␣40␣y␣50', 'mayor␣que␣50');
2 CREATE TYPE NivelEducacional AS ENUM ('educacion␣media', '
     educacion␣tecnica', 'educacion␣universitaria');
3 CREATE TYPE EstadoCivil AS ENUM ('soltero', 'casado', 'viudo','
     separado');
4 CREATE TYPE Actividad AS ENUM ( 'dependiente', 'estudiante', '
     empresario');
5 CREATE TYPE EstadoActual AS ENUM ('deuda␣de␣1␣mes', 'deuda␣de␣2␣
     meses', 'sin␣deuda');
6 CREATE TYPE TipoProducto AS ENUM ('A', 'B');
7 CREATE TYPE CompraPromo AS ENUM ('si', 'no');
8
9 CREATE TABLE cliente(
10        cliente_id SERIAL PRIMARY KEY,
11        rut_cliente INTEGER NOT NULL,
12        verificador_rut CHAR (1) NOT NULL,
13        p_apellido VARCHAR (50) NOT NULL,
14        s_apellido VARCHAR (50),
15        nombres VARCHAR (200) NOT NULL
16 );
17
18 CREATE TABLE detalle_cliente(
19        detalle_cliente_id SERIAL NOT NULL ,
20        rut_cliente INTEGER NOT NULL,
21        r_etario RangoEtario NOT NULL,
22        n_educacional NivelEducacional NOT NULL,
23        e_civil EstadoCivil NOT NULL,
24        actividad Actividad,
25        PRIMARY KEY (detalle_cliente_id),
26        FOREIGN KEY (detalle_cliente_id) REFERENCES cliente (
              cliente_id)
27 );
28
29 CREATE TABLE tarjeta(
30        tarjeta_id SERIAL PRIMARY KEY,
31        cliente_id SERIAL NOT NULL REFERENCES cliente (cliente_id)
              ,
```

```
32        rut_cliente INTEGER NOT NULL,
33        anio_apertura SMALLINT NOT NULL,
34        cupo_max INTEGER NOT NULL,
35        porc_uso_cupo REAL NOT NULL,
36        cant_atrasos_pago SMALLINT NOT NULL,
37        e_actual EstadoActual NOT NULL
38 );
39
40 CREATE TABLE compra(
41        compra_id SERIAL PRIMARY KEY,
42        tarjeta_id SERIAL REFERENCES tarjeta (tarjeta_id),
43        promocion CompraPromo NOT NULL
44 );
45
46 CREATE TABLE producto(
47        producto_id SERIAL PRIMARY KEY,
48        tipo_producto TipoProducto NOT NULL,
49        compra_id SERIAL REFERENCES compra (compra_id)
50 );
```

Listing 1: Creación de Tablas PostgreSQL

4. Población de Tablas

A continuación se presenta el script en el lenguaje PostgreSQL con sus restricciones aplicadas para el adecuado funcionamiento en la base de datos respecto a al script anterior. En base a la creación de tablas anterior, y a las enumeraciones que presentan las posibles opciones de valores en cada campo, se hace uso de una tabla temporal para poder poblar la base de datos en base al archivo "tablas.csv", el cual se encuentra adjunto al presente informe. Antes de ejecutar el script, se debe modificar la ruta de lectura del archivo para que apunte a la del directorio en el que se encuentre al momento de descargar el archivo.

```
1 CREATE TEMPORARY TABLE tmp(
2         rut INTEGER NOT NULL,
3         verificador CHAR(1) NOT NULL,
4         p_apellido VARCHAR(50) NOT NULL,
5         s_apellido VARCHAR(50),
6         nombres VARCHAR(200) NOT NULL,
7         r_etario RangoEtario NOT NULL,
8         n_educacional NivelEducacional NOT NULL,
9         e_civil EstadoCivil NOT NULL,
10        actividad Actividad,
11        e_actual EstadoActual NOT NULL,
12        anio_apertura SMALLINT NOT NULL,
13    cupo_max INTEGER NOT NULL,
14        porc_uso_cupo REAL NOT NULL,
15        compra_prom_anio SMALLINT NOT NULL,
16        unidades_prod_a SMALLINT NOT NULL,
17        unidades_prod_b SMALLINT NOT NULL,
18        atrasos_pago_hist SMALLINT NOT NULL,
19        compra_prom CompraPromo NOT NULL
20    );
21 COPY tmp FROM 'D:\tablas.csv' DELIMITER ';' NULL as '␣' CSV
      HEADER;
22
23 INSERT INTO cliente(rut_cliente, verificador_rut, p_apellido,
      s_apellido, nombres)
24        SELECT rut, verificador, p_apellido, s_apellido, nombres
25        FROM tmp;
26 INSERT INTO detalle_cliente(detalle_cliente_id, rut_cliente,
      r_etario, n_educacional, e_civil, actividad)
27        SELECT cliente.cliente_id, cliente.rut_cliente, r_etario,
              n_educacional, e_civil, actividad
28        FROM cliente, tmp
29        WHERE cliente.rut_cliente = tmp.rut
30        ORDER BY cliente.rut_cliente;
```

```
31 INSERT INTO tarjeta(cliente_id, rut_cliente, anio_apertura,
      cupo_max, porc_uso_cupo, cant_atrasos_pago, e_actual)
32         SELECT cliente.cliente_id, cliente.rut_cliente, tmp.
               anio_apertura, tmp.cupo_max,
33         tmp.porc_uso_cupo, tmp.atrasos_pago_hist, tmp.e_actual
34         FROM tmp
35         INNER JOIN cliente ON cliente.rut_cliente = tmp.cliente_id
               ;
36 INSERT INTO compra(tarjeta_id, promocion)
37         SELECT tarjeta.tarjeta_id, tmp.compra_prom
38         FROM tmp
39         INNER JOIN tarjeta ON tarjeta.cliente_id = tmp.cliente_id;
40 do $$
41         DECLARE
42         it RECORD;
43         cur CURSOR FOR SELECT * FROM tmp;
44         BEGIN
45                 FOR it IN cur LOOP
46                         FOR j IN 1..it.unidades_prod_a LOOP
47                                 INSERT INTO producto(tipo_producto
                                       , compra_id)
48                                 SELECT 'A', compra.compra_id
49                                 FROM compra;
50                         END LOOP;
51                 END LOOP;
52                 FOR it IN cur LOOP
53                         FOR j IN 1..it.unidades_prod_b LOOP
54                                 INSERT INTO producto(tipo_producto
                                       , compra_id)
55                                 SELECT 'B', compra.compra_id
56                                 FROM compra;
57                         END LOOP;
58                 END LOOP;
59         END;
60 $$
```

Listing 2: Población de Tablas PostgreSQL

5. Conclusiones

Del presente informe se obtuvo en detalle el proceso de organización necesario para conformar la base de datos propuesta para el cliente ficticio, de tal manera que facilita la construcción del software que debe utilizar las variables necesarias para que el negocio funcione de manera exitosa, dejándose expuesta una interpretación clara del modelo definido para entregar tiempos de respuesta breves en futuras consultas.

Así mismo, se obtuvo en detalle el proceso de población de una base de datos con la planilla antes entregada en el primer taller del presente curso. Además se muestra en detalle la función COPY, función que permite el proceso de mover datos desde múltiples fuentes y cargarlos en otra base de datos.