Trabajo 1 de Bases de Datos 2 (12.5%)

2024-1

Se tiene una tabla de documentos XML que guarda datos sobre clientes así:

```
DROP TABLE cliente;
CREATE TABLE cliente(
  id NUMBER(8) PRIMARY KEY,
  c XMLTYPE NOT NULL);

INSERT INTO cliente VALUES
(1, XMLTYPE('<Cliente clNro="445">
        <Estrato>5</Estrato>
        <Genero>m</Genero>
        </Cliente>'));

INSERT INTO cliente VALUES
(2, XMLTYPE('<Cliente clNro="800">
        <Estrato>88</Estrato>
        <Genero>x</Genero>
        </Cliente>'));
```

Nota: Se garantiza que todos los documentos XML de los clientes van a tener esta estructura y que tienen valores válidos como los mostrados en los ejemplos anteriores: todos los números son enteros positivos. La columna genero contiene cualquier letra en minúscula. Se garantiza que no hay dos clientes con el mismo clNro.

Se tiene una tabla de documentos XML que guarda datos sobre productos así:

```
DROP TABLE producto;
CREATE TABLE producto(
  id NUMBER(8) PRIMARY KEY,
  p XMLTYPE NOT NULL);

INSERT INTO producto VALUES
(1, XMLTYPE('<Producto plNro="100">
        <Marca>Micerdito Azul</Marca>
        <Tipoprod>Carnico</Tipoprod>
        </Producto>'));
```

```
INSERT INTO producto VALUES
(4, XMLTYPE('<Producto plNro="111">
        <Marca>Acme</Marca>
        <Tipoprod>Mueble de Oficina</Tipoprod>
        </Producto>'));
```

Nota: Se garantiza que todos los documentos XML de los productos van a tener esta estructura y que tienen valores válidos como los mostrados en los ejemplos anteriores: todos los números son enteros positivos. Se garantiza que no hay dos productos con el mismo plNro.

Se tiene una tabla de documentos JSON que guarda datos sobre ventas así:

```
DROP TABLE venta;
CREATE TABLE venta(
 id  NUMBER(8) PRIMARY KEY,
 jventa JSON NOT NULL
);
```

Cada documento JSON representa una venta hecha a un cliente. Veamos un ejemplo de una venta:

```
INSERT INTO venta VALUES (1,
'{
 "codventa": 66,
 "fecha": "29-11-2023",
 "codcliente": 445,
 "items": [
  {
   "codproducto": 100,
   "nrounidades": 3,
   "totalpesos": 75
  },
  {
   "codproducto": 111,
   "nrounidades": 1,
   "totalpesos": 1000
  },
  {
   "codproducto": 100,
   "nrounidades": 1,
   "totalpesos": 26
  }
 ]
}'
);
```

Se insertan muchas ventas en la tabla de ventas... Note que en una misma venta puede aparecer el mismo producto en diferentes celdas del arreglo items (por ejemplo, el producto 100 aparece en dos celdas). Cada venta sigue la estructura JSON mostrada en el ejemplo y tienen valores válidos como los mostrados en el ejemplo: la columna fecha se maneja en este formato: dd-mm-yyyy, así la fecha 05-12-2022 corresponde al 5 de diciembre de 2022. Todos los números son enteros positivos. El arreglo items de cada venta puede tener cualquier número de celdas, mínimo una celda. En el ejemplo el arreglo tiene tres celdas. Cada celda es un documento JSON con tres columnas: codproducto, nrounidades y totalpesos.

- Puede haber ventas que en su columna codcliente tengan códigos de clientes que **NO existen** en la tabla de clientes (clNro). Estos son clientes **no registrados**, pero que se les puede hacer ventas.
- Puede haber ventas que en su columna codproducto tengan códigos de productos que **NO existen** en la tabla de productos (plNro). Estos son productos **no registrados** en la tabla de productos, pero que igual son vendidos.

Realice programas PL/SQL para cada uno de los siguientes puntos.

Punto 1. (30%)

El programa recibe como parámetro un mes y un año. Ejemplo: 02 y 2022 (o sea, febrero de 2022). El programa debe imprimir el total en pesos vendido a cada **estrato y género** correspondiente a todas las ventas de ese mes y año. Es decir, en pantalla debe salir lo siguiente. Ejemplo:

Informe para 02-2022:

totalpesos	estrato	género
99840	4	m
85997	Ausente	Ausente
76550	4	x
76400	5	w
75589	2	m

Etc.

El informe en pantalla (o sea, lo que está en verde) debe salir ordenado descendentemente por totalpesos y debe tener **idéntica** estructura a la mostrada. De esta forma, el usuario puede saber fácilmente, por ejemplo, cuál es el estrato y género que más compra. El totalpesos correspondiente a los clientes **no registrados**, debe salir con estrato "Ausente" y género "Ausente", como se muestra en el ejemplo. Nota: en caso de que no haya ausentes (o sea, cuando todos los clientes de las ventas están registrados), debe salir: 0 Ausente Ausente.

Punto 2. (30%)

El programa recibe como parámetro el nombre de un tipo de producto. Ejemplo: “Mueble de Oficina”. El programa debe imprimir el total en pesos vendido **por cada marca de ese tipo de producto en cada año**. El informe debe salir ordenado ascendentemente por año. En cada año cada marca sale con su total pesos. Es decir, en pantalla debe salir así (suponiendo que el año menor en la tabla venta sea 2014):

Informe para Mueble de Oficina por año y marca:

2014

marca	totalpesos
Acme	34000
Mimueblecito	24666
Ikasa	55799

Total de los productos no registrados en la tabla producto: 78990

2015

marca	totalpesos
Acme	84000
Ikasa	55799

Total de los productos no registrados en la tabla producto: 0

2016
Etc.

Note que en la última línea de cada año se debe incluir el totalpesos correspondiente a los productos **no registrados** en la tabla producto. Si el tipo de producto ingresado como párametro no existe, imprima "No existe".

Punto 3. Triggers (40%)

Se tiene una tabla llamada casamatriz así:

```
DROP TABLE casamatriz;
CREATE TABLE casamatriz(
id NUMBER(8) PRIMARY KEY,
capitalp NUMBER(8) NOT NULL CHECK(capitalp > 0)
);
```

Ejemplos:

```
INSERT INTO casamatriz VALUES(1, 1000);
INSERT INTO casamatriz VALUES(2, 2000);
INSERT INTO casamatriz VALUES(3, 1000);
```

Se tiene una tabla llamada casahija así:

```
DROP TABLE casahija;
CREATE TABLE casahija(
id NUMBER(8) PRIMARY KEY,
capitalh NUMBER(8) NOT NULL CHECK(capitalh > 0),
casapadre NUMBER(8) NOT NULL REFERENCES casamatriz
);
```

Ejemplos:

```
INSERT INTO casahija VALUES(1, 300, 1);
INSERT INTO casahija VALUES(2, 300, 1);
INSERT INTO casahija VALUES(3, 350, 1);
INSERT INTO casahija VALUES(4, 400, 2);
INSERT INTO casahija VALUES(5, 1500, 2);
```

Etc.

Realice los siguientes triggers, **cada uno vale 10%**:

- Cuando se vaya a insertar una fila en casahija, se debe verificar que la suma de los capitales de todas las hijas de una misma casa matriz no supere el capital de la casa matriz. Considerando los ejemplos anteriores, si se va a insertar la siguiente fila en casahija:
  `INSERT INTO casahija VALUES(9, 125, 2);`

El trigger **debe impedir** esta inserción porque la suma de las hijas de la casa 2 daría: 2025 y la casa matriz número 2 solamente tiene un capital de 2000. Ahora si se fuese a insertar la siguiente fila en casahija:

INSERT INTO casahija VALUES(11, 80, 2);

Esta inserción **SÍ** se acepta porque la suma daría 1980 la cual es menor que 2000.

- Cuando se vaya a disminuir el capital de una casa matriz (o sea, capitalp), se debe garantizar que el nuevo capital de la casa matriz sea mayor o igual que la suma del capital de sus hijas. En los ejemplos anteriores, si se baja el capital (capitalp) de la casa matriz 1 a 990 **se acepta** (en el ejemplo la suma de las hijas de la casa 1 es 950), pero si se baja el capital (capitalp) a 949 **se rechaza**.
- Cuando se vaya a aumentar el capital de una casa hija (o sea, capitalh), se debe controlar que la suma del capital de las hijas (de una misma casa matriz) no supere el capital de su correspondiente casa matriz. En el ejemplo, si se sube el capital de la casa hija número 1 de 300 a 400 **se rechaza**, pero si se sube por ejemplo a 349 **se acepta**.
- Se debe controlar adicionalmente para la inserción, que el máximo número de hijas que una casa matriz pueda tener es 5. Además, para la actualización no se permite que una casa hija se cambie de casa matriz.

Notas adicionales:

• Para enviar por email a ***fjmoreno@unal.edu.co*** el **lunes 18 de Marzo hasta la 1pm**. Solo se califican trabajos enviados a ese correo. No se reciben trabajos en hora posterior. No se reciben versiones "mejoradas".
• No se califican trabajos enviados "por accidente" a otros correos.
• **NO modifique la estructura de las tablas dadas**. Sin embargo, puede usar todas las tablas, colecciones y estructuras de datos **auxiliares** que desee.
• Grupos de máximo tres personas.
• Los trabajos deben ser independientes entre los grupos. **Trabajos copiados así sea en una sola parte** se califican con 0 (cero) **en su totalidad** para todos los integrantes.
• El monitor le puede ayudar con aspectos referentes al enunciado, pero su función no es hacerle el trabajo **ni está autorizado para cambiar las condiciones del trabajo.**
• Si trabaja con **otros** sistemas de bases de datos o lenguajes, así su trabajo funcione y sea "espectacular", el trabajo se califica con cero.
• Si para hacer el trabajo, acude a herramientas como ChatGPT, le recomiendo no dependa totalmente de ellas, ya que su desempeño en el examen posiblemente no sea el mejor. Recuerde, lo informado al inicio del curso, si su intención es hacer los trabajos acudiendo a estas herramientas, no le recomiendo que tome la materia.
• Si encuentra errores en el enunciado por favor avisar lo más pronto posible para hacer las correcciones respectivas.
• Favor enviar en un solo mensaje todo el trabajo. Incluya en el mensaje los nombres de los integrantes. No envíe varias versiones del trabajo, **solamente una**.
• Incluya un informe máximo de dos páginas, en el cual aclare aspectos relevantes de su solución, por ejemplo, ¿qué ocurre si se le envía a su programa (en el punto 1) un mes que no existe (por ejemplo, el mes 14)? ¿Qué ocurre si en el punto 2, se envía el nombre de un tipo de producto en mayúsculas (por ejemplo, CÁRNICO o sin tilde)?

Francisco Moreno
Febrero 28 de 2024.
Un saludo respetuoso.