Índice

Objetivo de la práctica

En esta práctica el alumno utilizará las funcionalidades de PLSQL para automatizar algunas operaciones y para realizar comprobaciones sobre los datos. Estas operaciones y comprobaciones no pueden realizarse solo con SQL. Se incluyen también disparadores (*triggers*).

La última versión de esta práctica está disponible en este enlace.

Descripción del problema

Se parte de la práctica anterior. Una compañía necesita automatizar su almacén

- De cada producto se almacena su identificador, su nombre y su *stock*.
- En cada entrada de producto al almacén, se apunta
  - El producto
  - La cantidad de producto
  - El precio pagado al proveedor por unidad de producto
- De cada salida de producto del almacén se apunta
  - El producto, cantidad de producto y precio por unidad que paga el cliente
- Será necesario saber a qué precio se realizó la última compra y la última venta de cada producto

En esta práctica, se siguen utilizando las vistas `V_PRODUCTOS` y `V_EXISTENCIAS`. Los procedimientos y funciones `EXISTENCIAS_PRODUCTO`, `ENTRADA_PRODUCTO`, `SALIDA_PRODUCTO` y `SALIDA_PRODUCTO_CON_STOCK` son inicialmente los mismos. Puede comenzarse con la solución propuesta por el profesor.

Ejercicio 1 : Creación de producto con `insert`

Modifica el procedimiento `CREAR_PRODUCTO` para que si recibe un valor para el nuevo `IDPRODUCTO` lo utilice, y si el `IDPRODUCTO` ya existe, se lanzará el error `PRODUCTOYAEXISTE`. Si `IDPRODUCTO` es `NULL`, el identificador se extraerá del siguiente valor de la secuencia `SECUENCIA_PRODUCTO`.

```
1 create or replace procedure CREAR_PRODUCTO(nombreproducto IN varchar, idproducto IN
    ↪  OUT number)
2 as
3 -- VARIABLES QUE HAGAN FALTA
4 begin
5 -- SI idproducto ES NULL, USA LA SECUENCIA_PRODUCTO
6 -- INSERTA EL PRODUCTO CON ESE ID
7 -- DEVUELVO EL NUEVO ID EN idproducto
8 end;
9 /
```

Listado 1: Creación del procedimiento `CREAR_PRODUCTO`

Aviso

Si es necesario, se probarán números de la `SECUENCIA_PRODUCTO` hasta que se encuentre uno que no exista (por ejemplo, con un bucle `while`)

Crea un *trigger* de tipo instead of que permita crear un producto directamente con una orden `insert` en la vista `V_PRODUCTOS`. El *trigger* utilizará el procedimiento `CREAR_PRODUCTO`. Si no se da valor a la columna `NOMBRE_PRODUCTO` se lanzará el error `FALTANDATOS`.

```
1
2
3  create or replace trigger INSERTAR_PRODUCTO
4  instead of INSERT on V_PRODUCTOS
5  for each row
6  declare
7  -- VARIABLES QUE HAGAN FALTA
8  begin
9  --
10 -- Llama a CREAR_PRODUCTO con los parámetros necesarios
11 --
12 end;
13 /
14
15
16 insert into v_productos(nombreproducto) values ('Zapatos magnolia');
17 insert into v_productos(nombreproducto,idproducto) values ('Zapatos rosa', 25);
18
19 -- ESTO DA EL ERROR FALTANDATOS
20 insert into v_productos(idproducto) values (230);
```

Listado 2: Creación de producto con `insert`

Ejercicio 2 : Saldo precalculado

Crea una tabla `T_SALDO` que contenga la ganancia (o pérdida) de la empresa, a partir de las entradas y salidas de producto. Tendrá solo una fila, con una única columna de nombre `SALDO` con el dinero que se haya conseguido con las salidas, menos el dinero que se haya gastado con las entradas.

- Añade los *triggers* necesarios a tus tablas para que cada vez que se produzca una entrada o salida se actualice el saldo de la empresa.
- Si la tabla `T_SALDO` tiene ya una fila, esa fila se actualizará cuando haya una entrada o una salida
- Si la tabla `T_SALDO` no tiene ninguna fila, se creará la fila cuando haya una entrada o una salida

```
1  create table T_SALDO(saldo number(10,2));
```

Listado 3: Existencias precalculadas

Ejercicio 3 : Creación rápida de productos

Se modificará la tabla `V_EXISTENCIAS` para que incluya el nombre del producto.

```
1 create or replace view V_EXISTENCIAS(idproducto,nombreproducto,existencias,
    ↪ ultimopreciocompra,ultimoprecioventa) as
2 ...
```

Listado 4: Vista de existencias

El cliente necesita que se pueda dar de alta y dar entrada a un producto de forma rápida. Para ello, se desea que se pueda insertar directamente en la vista `V_EXISTENCIAS`.

- Si se indica un `idproducto`, se creará un producto con ese identificador. Si no, se utilizará la secuencia de la práctica anterior.
- El nombre del producto será el indicado en el `insert`. Si es `NULL` se lanzará el error `FALTANDATOS`
- Se creará una entrada de producto con la cantidad indicada en `existencias`, al precio marcado en `ultimopreciocompra`. Si alguno de estos es `NULL` se lanzará el error `FALTANDATOS`.
- Si se indica un `ultimoprecioventa`, se lanzará el error `SOBRANDATOS`

```
1  -- Se añade un producto de nombre OFERTA, id 1234, con una entrada de 10
     ↪ unidades a 20 euros
2  insert into v_existencias(idproducto,nombreproducto,existencias,
     ↪ ultimopreciocompra)
3  values (1234,'OFERTA',10,20);
4
5  -- Se añade un producto de nombre OFERTON, con identificador sacado de la
     ↪ secuencia, con una entrada de 10 unidades a 20 euros
6  insert into v_existencias(nombreproducto,existencias,ultimopreciocompra)
7  values ('OFERTON',10,20);
8
9  -- Error FALTANDATOS
10 insert into v_existencias(ultimopreciocompra) values (20);
11
12 -- Error SOBRANDATOS
13 insert into v_existencias(nombreproducto,existencias,ultimopreciocompra,
     ↪ ultimoprecioventa)
14 values ('producto',10,20,30);
```

Listado 5: Ejemplos de insert

Ejercicio 4 : Control de precios

Se desea evitar las variaciones muy rápidas de los precios pagados a los proveedores.

- Se pondrá un *trigger* en la tabla donde se apunten las entradas, de nombre `CONTROL_PRECIOS_ENTRADA`.
- La entrada no se podrá guardar si su precio difiere en más de 10€(por arriba o por abajo) de la entrada anterior. En ese caso, se lanzará el error `PRECIOFUERADERANGO`
- Si nunca ha habido una entrada para ese producto, se podrá guardar.

```
1  create or replace trigger CONTROL_PRECIOS_ENTRADA
2  before insert on .....
3  for each row
4  declare
5  -- VARIABLES QUE HAGAN FALTA
6  begin
7  .....
8  if ..... then
9  RAISE_APPLICATION_ERROR(-20200, 'PRECIOFUERADERANGO');
10 end if;
11 .....
12 end;
13
14 -- PRUEBA DEL TRIGGER
15 declare
16 id number;
17 begin
18 crear_producto( 'Pera limonera', id );
19 entrada_producto( id, 1, 10); -- COMPRO 1 A 10€, ADMITIDO POR SER LA PRIMERA COMPRA
20 entrada_producto( id, 3, 20); -- COMPRO 3 A 20€, ADMITIDO
21 entrada_producto( id, 2, 9); -- COMPRO 2 A 9€, DEBERIA DAR ERROR
22 end;
23 /
```

Listado 6: Control de precios de entrada

Aviso

Un *trigger* no puede acceder a los datos de una tabla que acaba de ser modificada, solo a `:new` y `:old` (ORA-04091). Por eso, este *trigger* es `BEFORE` en vez de `AFTER`

Instrucciones de entrega

La entrega se realizará en el servidor del profesor. Si no estuviera operativo, se entregará un único fichero SQL para todos los apartados con las sentencias SQL necesarias para crear las tablas, secuencias, procedimientos, funciones y vistas que el alumno necesite.

- Este ejercicio se corregirá de forma semiautomática, por lo que es necesario seguir la nomenclatura propuesta en el ejercicio.
- Si tiene **errores** de compilación podría no corregirse. Si no se siguen los **nombres de objetos** pedidos podría no corregirse.

El documento se subiría a la tarea correspondiente en el aula virtual. Presta atención al plazo de entrega (con fecha y hora).