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

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`

Crea un *trigger* de tipo instead of que permita crear un producto directamente con una orden `insert` en la vista `V_PRODUCTOS`. El *trigger* ignorará el valor del identificador de producto, y llamará al procedimiento `CREAR_PRODUCTO`.

```
1  create or replace trigger INSERTAR_PRODUCTO
2  instead of INSERT on V_PRODUCTOS
3  for each row
4  declare
5  -- VARIABLES QUE HAGAN FALTA
6  begin
7  --
8  -- Ignora el identificador y llama a CREAR_PRODUCTO con el nombre
9  --
10 end;
11 /
12
13
14 insert into v_productos(nombreproducto) values ('Zapatos magnolia');
```

Listado 1: Creación de producto con `insert`

Ejercicio 2 : Creación rápida de productos

Para compras de productos, se desea que se pueda insertar directamente en la vista `V_EXISTENCIAS`.

- Si se indica un `idproducto`, se insertará un producto con ese identificador. Si no, se utilizará la secuencia de la práctica anterior.
- El nombre del producto será `OFERTA`.
- Se creará una entrada de producto con la cantidad indicada en `existencias`, al precio marcado en `ultimopreciocompra`. Si alguno de estos en `NULL` se lanzará el error `-20103`.
- Si se indica un `ultimoprecioventa`, se lanzará el error `-20104`

```
1  -- Se añade un productro de nombre OFERTA, id 1234, con una entrada de 10
       ↪ unidades a 20 euros
2  insert into v_existencias(idproducto,existencias,ultimopreciocompra)
3               values (1234,10,20);
4
5  -- Se añade un productro de nombre OFERTA, con identificador sacado de la
       ↪ secuencia, con una entrada de 10 unidades a 20 euros
6  insert into v_existencias(existencias,ultimopreciocompra)
7               values (10,20);
8
9  -- Error -20103
10 insert into v_existencias(ultimopreciocompra) values (20);
11
12 -- Error -20104
13 insert into v_existencias(existencias,ultimopreciocompra,ultimoprecioventa)
14              values (10,20,30);
```

Listado 2: Ejemplos de insert

Ejercicio 3 : 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.
- Cuando se haga una nueva compra para un producto, se comprobará si va a ser la última (si no hay entradas posteriores).
- Si la entrada va a ser la última, no se podrá guardar si su precio difiere en más de 10€(por arriba o por abajo) de la que hasta ese momento era la última. En ese caso, se lanzará el error `-20200` con `RAISE_APPLICATION_ERROR`.
- Si nunca ha habido una entrada para ese producto, siempre 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, 'Precio fuera de rango');
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, 1, 9); -- COMPRO 2 A 9€, DEBERIA DAR ERROR
22 end;
23 /
```

Listado 3: 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`

Entorno de pruebas

En hay accesible un servidor Oracle (`alvarogonzalez.no-ip.biz`), con un usuario para cada alumno. El profesor pasará unas pruebas automáticas, que dejarán el resultado en http://alvarogonzalez.no-ip.biz:8088.

Instrucciones de entrega

La entrega se realizará en el servidor de pruebas (`alvarogonzalez.no-ip.biz`). 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 fichero se corregirá de forma semiautomática, por lo que es necesario seguir la nomenclatura propuesta en el ejercicio.
- El fichero se cargará en un usuario recién creado con permisos necesarios para crear todos los elementos necesarios (tablas, vistas, funciones, secuencias...)
- Si tiene **errores** de compilación podría no corregirse. Si no se siguen los **nombres de objetos** pedidos podría no corregirse.

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