

Índice

Objetivo de la práctica	2
Descripción del problema	2
Ejercicio 1 : Creación de producto con insert	2
Ejercicio 2 : Existencias precalculadas	3
Ejercicio 3 : Control de precios	3
Entorno de pruebas	4
Instrucciones de entrega	4

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
 - La fecha y hora de entrada (timestamp)
 - El producto
 - La cantidad de producto
 - El precio pagado al proveedor por unidad de producto
- De cada salida de producto del almacén se apunta
 - La fecha y hora de salida (timestamp)
 - 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.

```
create or replace trigger INSERTAR_PRODUCTO
instead of INSERT on V_PRODUCTOS
for each row
declare
    -- VARIABLES QUE HAGAN FALTA
begin
    --
    -- Ignora el identificador y llama a CREAR_PRODUCTO con el nombre
    --
end;
/

insert into v_productos(nombreproducto) values ('Zapatos magnolia');
```

Listado 1: Creación de producto con insert

Ejercicio 2 : Existencias precalculadas

Crea una tabla T_EXISTENCIAS_PRECAL que contenga el stock de cada producto de forma precalculada. Para ello:

- Crea un procedimiento INICIALIZA_EXISTENCIAS_PRECAL, que borre todas las filas de la tabla T_EXISTENCIAS_PRECAL e inserte de nuevo una fila por cada producto, con sus existencias.
- Añade los *triggers* necesarios a tus tablas para que cada vez que se produzca una entrada o salida se actualice el stock de esta tabla, sin que se vuelvan a leer del resto de entradas o salidas de ese producto.

```
create table T_EXISTENCIAS_PRECAL(idproducto number(10), existencias number(10));

create or replace procedure INICIALIZA_EXISTENCIAS_PRECAL
as
    -- VARIABLES QUE HAGAN FALTA
begin
    -- BORRA TODAS LAS FILAS DE T_EXISTENCIAS_PRECAL
    for producto in .... loop
        -- INSERTA UNA FILA POR CADA FILA EN V_PRODUCTOS
    end loop;
end;

create or replace trigger ACT_EXISTENCIAS_ENTRADA
after insert on .....
for each row
declare
    -- VARIABLES QUE HAGAN FALTA
begin
    .....
end;

create or replace trigger ACT_EXISTENCIAS_SALIDA
after insert on .....
for each row
declare
    -- VARIABLES QUE HAGAN FALTA
begin
    .....
end;
```

Listado 2: Existencias precalculadas

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.

```
create or replace trigger CONTROL_PRECIOS_ENTRADA
before insert on .....
for each row
declare
    -- VARIABLES QUE HAGAN FALTA
begin
    .....
    if ..... then
        RAISE_APPLICATION_ERROR(-20200, 'Precio fuera de rango');
    end if;
    .....
end;

-- PRUEBA DEL TRIGGER
declare
    id number;
begin
    crear_producto( 'Pera limonera', id );
    entrada_producto( id, 1, 10, systimestamp); -- COMPRO 1 A 10€, ADMITIDO POR SER
    ↪ LA PRIMERA COMPRA
    entrada_producto( id, 3, 20, systimestamp); -- COMPRO 3 A 20€, ADMITIDO
    entrada_producto( id, 1, 9, systimestamp); -- COMPRO 2 A 9€, DEBERIA DAR ERROR
end;
/
```

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).