

Serie Ejercicios Prácticos 1 Bases de Datos Activas

Determinar las instrucciones SQL necesarias, para la definición de las restricciones de integridad, reglas de negocio y triggers requeridas en cada ejercicio.

- 1)** Crear la tabla Empleados, con los siguientes atributos: DNI Empleado, Legajo, Apellido y Nombres, Cargo y Sueldo, tener en cuenta lo siguiente para cada uno de ellos:

- DNI: Clave primaria
- Legajo: cumpla con la restricción de unicidad
- Cargo: este comprendido entre los valores 50-120
- Sueldo: no sea mayor a 90000

Alternativa: crear un dominio de valor para el atributo DNI, como entero de 8 dígitos, verificando que sólo acepte valores mayores a 0.

- Crear un trigger de nombre Control_sueldo, que impida que se incremente el sueldo en más de un 20%, cada vez que se modifique dicho atributo. El evento que dispara el trigger es UPDATE (sueldo), y su tiempo de acción es BEFORE.
- Crear un trigger de nombre Adicional_cargo, para el caso en que el atributo cargo tome el valor 90, entonces deberá de incrementarse el atributo sueldo de la tabla Empleados en un 15% respecto del importe básico para ese cargo, este último (importe básico) se halla almacenado en la tabla Importes_basicos, cuya clave primaria es el atributo cargo. El evento que dispara el trigger es INSERT/UPDATE del atributo cargo, y su tiempo de acción es AFTER, de la tabla Empleados.

- 2)** Crear la tabla Seguros_autos, referido a los seguros de automotores, con los siguientes atributos: Nro.Póliza, DNI Cliente, Marca del vehículo, Año modelo del vehículo, Nro. de patente, Nro. de motor, Periodo de pago e Importe a pagar (periodo trimestral), considerar:

- Nro. póliza: Clave primaria
- DNI Cliente: tiene una integridad referencial con la tabla Clientes (DNI)
- Año modelo del vehículo: tenga como valor predeterminado '2019'
- Marca: sólo pueda contener como dato uno de los siguientes valores 'Ford, Renault, Fiat, Peugeot, VW, Toyota, Nissan'
- Periodo: 202001 (Consideraremos el año 2020 y el 1er trimestre 01)
- Importe a pagar: no puede ser nulo
- Constituir clave externa con los atributos marca y año modelo, con referencia a la tabla Vehículos(marca,año_modelo)
- Crear un trigger de nombre Facturas_mensuales, en el cual se registrarán las cuotas que se deben de abonar en cada trimestre del año 2020, para ello se insertaran en la tabla Cuotas_Seguros los comprobantes respectivos a cada trimestre, considerado para este caso práctico las cuotas del primer trimestre del año 2020 (serían 3 nuevos registros a incorporar, para este trigger en particular).
Los atributos de la tabla Cuotas_Seguros (tabla ya existente), se incorporarán de la siguiente manera:
 - o Nro_comprobante: se compone del Nro. póliza, periodo y número de cuota (01, 02 o 03).
 - o Monto facturado de cada cuota (corresponde a 1/3 del importe a pagar contenido en la tabla Seguros_autos).
 - o Fecha de pago: corresponde al día 15 de cada mes (15/01/20, 15/02/20, 15/03/20), consideraremos estas tres fechas para representar este trigger en particular, es decir los meses del primer trimestre del año 2020.

El evento que dispara el trigger es INSERT de la tabla Seguros_autos y su tiempo de acción es AFTER.

3) Crear la tabla Pedidos, con los siguientes atributos: Nro. de orden de pedido, fecha de la orden de pedido, CUIT del cliente, CUIT del proveedor, nro. de producto, tipo de pedido y cantidad pedida del producto, considerar:

- CUIT del cliente y Nro. de orden de pedido: Clave primaria
- Nro. de orden de pedido: cumpla con la restricción de unicidad
- Fecha de la orden de pedido: valor por defecto la fecha actual del sistema
- CUIT del cliente: integridad referencial con la tabla Clientes(Cuit)
- Tipo de pedido: puede contener únicamente M,C,G,H,N
- Cantidad pedida: valores mayores a 0
- Crear un trigger de nombre Control_pedido, que impida dar de alta una orden de pedido, cuando la cantidad pedida del producto sea superior al atributo stock_actual de la tabla Stock_productos cuya clave primaria es el atributo número de producto, caso contrario actualizar el stock_actual, que resulta luego de restarle la cantidad del actual pedido. El evento que dispara el trigger es INSERT, y su tiempo de acción es BEFORE, respecto de la tabla Pedidos.

4) Respecto a las tablas Pedidos y Stock_productos del ejercicio 3), agregue las siguientes restricciones:

Para la tabla Pedidos:

- "ClaveProducto" de tal manera que nro. de producto y tipo de pedido tenga una referencia externa a la tabla Productos(nro_producto,tipo_pedido)
- "CantidadPermitida" donde el atributo Cantidad pedida, sólo acepte valores comprendidos entre 10 y 2000.
- "ControlCUIT" donde el atributo CUIT del cliente no deberá de coincidir con el CUIT del proveedor, previendo deshabilitar esta comprobación en los datos ya existentes.

Para la tabla Stock_productos:

- "ControlStock" a fin de verificar que el stock actual de un determinado producto no sea inferior al stock mínimo del mismo, siendo los respectivos atributos stock_minimo y stock_actual.

5) Crear la tabla Clientes, con los atributos: código de cliente, nombre, dirección, código postal e importe base.

Crear la tabla Facturas, con los atributos: Nro. de factura, fecha de la factura, código de cliente de la factura, tipo de descuento, valor de IVA e importe de la factura.

Contemplar:

- Crear un valor de dominio de nombre CodigoCliente para el atributo código de cliente, tipo entero de 9 dígitos, que sea mayor a 0

Tabla Clientes:

- Código cliente: referenciar al valor dominio creado anteriormente, siendo clave primaria
- Nombre y dirección: no nulos
- Código postal: integridad referencial con la tabla CodPostales(cod_postal)
- Importe base no nulo

Tabla Facturas:

- Nro. Factura: clave primaria
- Código de cliente de la factura: referenciar al valor dominio creado e integridad referencial a la tabla Clientes(cliente_factura)
- Tipo de descuento: sólo puede contener valores comprendidos entre 5 y 20 respectivamente (estos valores corresponden a porcentajes %)
- IVA: sólo puede contener alguno de estos valores 15 o 21 (corresponden a porcentajes %).
- Importe de la factura, no nulo.
- Crear un trigger de nombre Calcular_importe_factura, que permita actualizar el importe de la factura que debe de abonar cada cliente de la tabla Facturas, el cual se obtiene sobre el importe base de la tabla Clientes, descontando el porcentaje

contenido en el atributo tipo de descuento, y sobre este resultado aplicar el porcentaje del IVA (15 o 21%), para obtener el valor definitivo correspondiente al importe de la factura, ambas tablas se relacionan en base al atributo código de cliente.

El evento que dispara al trigger es INSERT de una tupla o UPDATE respecto al atributo importe base de la tabla Clientes y su tiempo de acción es AFTER.

- 6) Crear la tabla Empleados_baja, con los atributos DNI Empleado, Legajo, Cargo, Usuario y Fecha.

Luego crear el trigger de nombre Empleado_eliminado, que permita insertar en la tabla Empleados_baja, los datos de aquel empleado que se elimina de la tabla Empleados (del ejercicio 1), donde las columnas usuario y fecha se grabarán con las variables del sistema USER y SYSDATE o DATE, asociados al identificador del usuario que opera el sistema en ese momento y la fecha actual en que se lleva a cabo este proceso de baja de un empleado.

Crear otro trigger de nombre Baja_usuario, que permita eliminar de la tabla Usuarios, aquella tupla cuyo DNI de esta última tabla, sea igual al DNI del empleado eliminado en el paso anterior.

El evento que dispara ambos triggers es DELETE, y su tiempo de acción es AFTER respecto a la tabla Empleados.

- 7) Modificar el trigger Baja_usuario creado en el ejercicio anterior, de tal manera que, para eliminar de la tabla de Usuarios, en lugar de hacerlo con el atributo DNI, demos la baja del empleado mediante su número de Legajo, utilizar el comando Replace trigger para registrar este cambio en el disparador.

Indicar los comandos Sql necesarios para desactivar el trigger Calcular_importe y también todos los triggers que estén asociados a la tabla Consumos.

- 8) Respecto de la tabla Facturas, creada en el ejercicio 5), incorporar a la tabla las columnas de Fecha_vto y Fecha_pago de tipo fecha (date) y la columna intereses de tipo entero.

Crear un trigger de nombre Recargo_factura, que detecte aquellos pagos fuera de término, para lo cual, si la fecha de pago es superior a la fecha de vencimiento, asignar el valor 5 al atributo intereses previamente incorporado, este recargo es el porcentaje que se vería reflejado en el cobro de la próxima factura del cliente, esta actualización sobre la tabla Facturas la accedemos teniendo en cuenta el código de cliente de las tablas del punto 5).

El evento que dispara al trigger es UPDATE respecto al atributo fecha de pago de la tabla Facturas y su tiempo de acción es AFTER.

- 9) Crear un nuevo trigger de nombre Importe_recargo respecto de la tabla Facturas, que refleje el recargo (porcentaje) que corresponde pagar a aquellos clientes que abonaron la factura fuera de término, este se obtiene de aplicar el valor del atributo intereses respecto del importe de la factura, por lo que el nuevo monto de la factura se verá incrementado con el importe resultante del recargo aplicado (Importe de la factura= Importe de la factura + (Importe de la factura con recargo)).

El evento que dispara al trigger es UPDATE respecto al atributo intereses de la tabla Facturas y su tiempo de acción es AFTER.

- 10) Crear la tabla **Cursadas** de la facultad (contiene las notas finales de todas aquellas materias que han cursado los alumnos), con los siguientes atributos: **DNI, Cód. de carrera, Cód. de materia, año de cursado, condición y Nota final obtenida**, tener en cuenta lo siguiente:

- DNI: Clave primaria
- Contemplar una restricción de integridad (**Constraint**) de nombre "Control_Materia" de tal manera que cód. de carrera y cód. de materia, tenga una referencia externa a la tabla Plan_Carreras(Cód. de carrera, Cód. de materia)
- Año de cursado: no nulo
- Condición: puede contener únicamente los valores 'R', 'P', 'D'

- Nota final obtenida: puede tener un rango de valores comprendidos entre 0 y 10 solamente
- Crear un **trigger** de nombre Condicion_alumno, que permita asignar la condición del alumno de acuerdo a la nota final obtenida, de la siguiente forma:
 - o Si la nota final obtenida es igual a 6, regularizo la materia, entonces asignar al atributo condición el valor 'R'
 - o Si la nota final obtenida es igual o superior a 7, promociono la materia, entonces asignar al atributo condición el valor 'P'
 - o Si la nota final obtenida es inferior a 6, desaprobó la materia, entonces asignar al atributo condición el valor 'D'

El evento que dispara el trigger es **INSERT** o **UPDATE** de la tabla Cursadas, y su tiempo de acción es **AFTER**.

Anexo

Disparadores o Triggers

Un trigger es un bloque de código SQL que se almacenan en la base de datos. Los bloques de código de los triggers están asociados a una tabla y se ejecutan automáticamente cuando se producen ciertos eventos asociados a la tabla.

Se suelen utilizar para prevenir transacciones erróneas y nos sirven también para implementar restricciones de integridad o seguridad.

Su formato básico es el siguiente:

```
create or replace trigger nombre_trigger

/* Tiempo de acción  Evento que activa el trigger */
{before | after} {delete | insert | update [of columna-lista_columnas]}

[or {before | after} {delete | insert | update [of columna-lista_columnas]}]

on nombre_tabla_asociada

[for each row]

/* comienza el trigger */
[declare]
  <declaraciones>
begin
  <instrucciones, sentencias>
end;
```