BASES DE DATOS Fecha Entrega 17-03-2016

Profesor: Héctor Gómez Gauchía

Práctica 4: USO de Bloques PL/SQL

APARTADO 0.- Preparación para la ejecución

- Prepara tu cuenta de Oracle
  - siguiendo los pasos en el fichero: `instruccionesABD-2016.pdf`

- Cargar los datos de la Base de Datos de Restaurantes:
  1.- ejecutar el `crea_tablas_restaurantes.sql`
  2.- ejecutar el `carga_datos_restaurantes.sql`

- Para informarte sobre los detalles la BD que usamos: mira al final de este documento

- Información útil para uso del PL/SQL y la Depuración: (se recomienda repasar)

  - ver fichero depuración-en-SQLDeveloper-2016.pdf

- AL PRINCIPIO DE CADA sesión con el SQLDeveloper, ejecutar en el editor:

  - `SET AUTOCOMMIT ON` (comprobarlo con `SHOW AUTOCOMMIT`)
  - Para ver los mensajes producidos por `PUT_LINE`:

```
SET SERVEROUTPUT ON SIZE 100000;
```

Probar con este bloque

```
DECLARE      -- se ejecuta y debe salir "Hola"
V VARCHAR(10);
BEGIN
V:='Hola';
DBMS_OUTPUT.PUT_LINE(V);
END;
```

NOTA: Esta práctica debe estar implementada en tu usuario de Oracle de la facultad.

NOTA sobre gestión de FECHAS: existen dos funciones to_char y to_date: ver ejemplos el fichero `crea_tablas_restaurantes.sql`

APARTADO 1.-

Hacer un procedimiento almacenado llamado PEDIDOS_CLIENTE que reciba como parámetro el DNI de un cliente y muestre por pantalla sus datos personales, junto con un listado con los datos de los pedidos que ha realizado (código de pedido, fecha, fecha de entrega, estado e importe del pedido), ordenados crecientemente por fecha. En caso de error (DNI no existe, no hay pedidos para ese cliente, etc..), deberá mostrarse por pantalla un mensaje de advertencia explicando el error, usando excepciones. Al finalizar el listado se deberá mostrar la suma de los importes de todos los pedidos del cliente (debes usar el mismo cursor que usaste antes recorriéndolo de nuevo).

Entregar: a) fichero pedido_cliente.sql. b) Un bloque de código anónimo para probar el procedimiento para cada uno de los casos posibles (el cliente no existe, sí existe pero no tiene pedidos y sí existe y además tiene pedidos). c) El resultado de la ejecución con lo que devuelve el bloque anónimo.

APARTADO 2.-

Hace un procedimiento almacenado llamado REVISA_PRECIO_CON_COMISION (sin argumentos) cuya misión es comprobar la consistencia de los datos de todos los precios con comisión en la tabla Contiene. El campo "precio con comisión" (puede estar a NULL) de la tabla "Contiene" debe almacenar el precio del plato incluyendo la comisión (que es un porcentaje) de su restaurante, haciendo el cálculo adecuado. El procedimiento debe verifica que el valor que tenía en la tabla contiene corresponde con el valor calculado y, si no, actualizar el valor de modo que resulten consistentes. Al terminar, si todos los datos son correctos, se mostrará un mensaje indicando "Ningún cambio en los datos de Contiene". En caso contrario se indicará el número de filas modificadas en Contiene, y llamará a una función *guardaPlato* que las creará en una tabla

TrazaPlatos que tiene los mismos atributos que Contiene, devolviendo un cero si ha ido todo correcto o un 1 si ha fallado algo. El procedimiento dará un mensaje si la función ha devuelto un 1.
Entregar: a) fichero revisa_precio_con_comision.sql. y guardaPlato.sql b) Un bloque de código anónimo para probar el procedimiento para cada uno de los casos posibles (que haya cambios en Contiene o que no haya). c) El resultado de la ejecución con lo que devuelve el bloque anónimo.

APARTADO 3.-

Procedimiento almacenado llamado REVISA_PEDIDOS (sin argumentos) cuya misión es comprobar la consistencia de los datos de todos los pedidos. Se pide usar un cursor FOR UPDATE. El campo "importe total" de la tabla "Pedidos" debe almacenar la suma de los "precio con comisión" de los platos (en tabla "Contiene") del pedido multiplicados por las unidades servidas del plato. El procedimiento llamará una función que verifica si coinciden y, en caso contrario, actualizar estos datos para todos los pedidos, de modo que resulten consistentes. Si todos los datos son correctos, se mostrará un mensaje indicando "Ningún cambio en los datos de la tabla Pedidos". En caso contrario se indicará el número de filas modificadas en la tabla Pedidos y cada fila modificada.
Entregar: a) fichero revisa_pedidos.sql. b) Un bloque de código anónimo para probar el procedimiento para cada uno de los casos posibles (que haya cambios o que no haya). c) El resultado de la ejecución con lo que devuelve el bloque anónimo.

APARTADO 4.-

Crea un procedimiento DATOS_CLIENTES que recorra todos los Clientes con un FOR y muestre todos sus datos junto con la suma de importe total de todos sus pedidos, para ello llama a una función *pedidos_cliente_tot* que es como el procedimiento pedidos_cliente y, además de hacer los que hace ese procedimiento, devuelve el acumulado de la suma total de los importes del cliente para que, al terminar con todos los clientes se muestre un mensaje con la suma total de todos los pedidos de todos los clientes.
Entregar: a) fichero datos_clientes.sql. y pedidos_cliente_tot.sql b) Un bloque de código anónimo para probar el procedimiento. c) El resultado de la ejecución con lo que devuelve el bloque anónimo.

APARTADO 5.-

Consulta las wikis: si no hay soluciones todavía o son diferentes a la tuya, sube la tuya.
Puedes modificar las otras soluciones que haya si lo crees conveniente.

Información sobre la Base de Datos de Restaurantes

Esquema Relacional

```
Restaurantes (codigo Number(8), nombre Char(20), calle Char(30)
              , código_postal Char(5), comision* Number(8, 2))
AreasCobertura (codigoRes Number(8), codigoPostal Char(5))
Horarios (codigoRes Number(8), dia_semana Char(1)
          , hora_apertura Date, hora_cierre Date)
Platos (restaurante Number(8), nombrePlato Char(20), precio* Number(8,2)
        , descripcion* Char(30), categoria* Char(10))
Pedidos (codigo Number(8), estado Char(9), fecha_hora_pedido Date
         , fecha_hora_entrega* Date, importeTotal* Number(8,2)
         , cliente Char(9), codigodescuento* Number(8))
Contiene (restaurante Number(8), plato Char(20), pedido Number(8)
          , precio_con_comision* Number(8,2), unidades Number(4))
Descuentos (codigodescuento Number(8), fecha_caducidad* Date
            , porcentaje_descuento Number(3))
Clientes (DNI Char(9), nombre Char(20),apellido Char(20), calle* Char(20)
          , numero Number(4), piso* Char(5), localidad* Char(15)
          , código_postal* Char(5), telefono* Char(9)
          , usuario Char(8), contraseña Char(8))
```

Descripción de la base de datos

"Restaurante en casa S.A." es una compañía que distribuye pedidos de comida desde restaurantes a casas particulares y oficinas. La aplicación debe almacenar la siguiente información:

- De cada restaurante se conoce su nombre, dirección, horario detallado para cada día de la semana y áreas de cobertura, que serán las localidades a las que se pueden servir pedidos desde el restaurante. Para facilitar la gestión de datos, cada restaurante tiene un código identificativo único.
- Cada restaurante ofrece un conjunto de platos distintos que se pueden elegir de manera individual en cada pedido. Cada plato tiene un nombre, descripción y un precio, al que se debe agregar la comisión que la compañía aplica a cada restaurante. En el sistema, los platos se agrupan en categorías comunes a todos los restaurantes (e.g. pescados, arroces, etc.).
- Los clientes de la empresa facilitan su DNI, nombre, apellidos, dirección (calle, número, piso, localidad, código postal) y un número de teléfono de contacto la primera vez que utilizan el servicio de "Restaurante en casa S.A.", de modo que quedan registrados en el sistema. Para consultar el estado de sus pedidos, cada cliente puede disponer de un usuario y una contraseña.
- Los pedidos que cada cliente realiza se componen de la siguiente información: código de pedido (automáticamente generado a partir de una secuencia que se inicia a 1), fecha del pedido, fecha de entrega, estado del pedido, importe total y cliente. Interesa además registrar los platos que componen el pedido, indicando las unidades y precio de cada uno de ellos (comisión incluida).
- La compañía distribuye cupones descuento cuyos datos son su código (no se repite), fecha de caducidad y porcentaje de descuento. Al hacer un pedido, el cliente indicará, si dispone de alguno, el código de su cupón. En ese caso, el pedido debe incluir la información de descuento, aplicándolo, en consecuencia, al importe final del pedido.

- Otras consideraciones:
  - El descuento de los cupones debe ser mayor que cero y menor o igual a cien.
  - Se necesita un índice que optimice la búsqueda de platos según su categoría.
  - Existen clientes que no han personalizado su contraseña porque no han accedido nunca por internet (sus pedidos son telefónicos). En ese caso, la contraseña por defecto es “Nopass”. El nombre de usuario es obligatorio, debiendo verificar que no se repite.
  - El estado por defecto de todos los pedidos es “REST”, y debe ser “REST”, “CANCEL”, “RUTA”, “ENTREGADO” ó “RECHAZADO”.
  - Para las claves ajenas configurar el comportamiento de la clave ajena cuando se eliminan filas referenciadas. Por ejemplo:

    -Si se borra un descuento se debe poner automáticamente el campo codigoDescuento a NULL en su pedido correspondiente

    -Si se borra un restaurante se eliminan automáticamente sus platos y sus horarios

    -Si se borra un pedido de debe eliminar automáticamente los platos que contiene

    -No se puede eliminar un cliente mientras tenga pedidos
  - -No se puede borrar un plato mientras existan pedidos con dicho plato