BASES DE DATOS Fecha Entrega 7-04-2016

Profesor: Héctor Gómez Gauchía

Práctica 2: USO de Triggers, Funciones Pipelined y PLSQL Dinámico

APARTADO 0.- Preparación para la ejecución

- Esta práctica se realiza sobre la Base de Datos de Restaurantes tal y como la tienes después de tus pruebas de la práctica anterior.
- Los conceptos aplicados son los vistos en clase con los ejemplos del Campus Virtual y de las transparencias de teoría.

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

APARTADO 1.-

Uno de las aplicaciones más útiles de los triggers es conservar la consistencia de datos relacionados que no puede ser mantenida por las restricciones estándar de Oracle. Queremos mantener la consistencia entre los valores de tabla `Contiene` y el importe total en tabla `Pedidos`. Para ello crea un trigger llamado `actualiza_total_pedidos` que se asocie a todas las operaciones posibles (insertar, eliminar y modificar) sobre la tabla `Contiene`. Al insertar una nueva fila en esta tabla, se deberá incrementar el valor del campo importe_total de `Pedidos`. Si se produce la eliminación de una fila o una actualización, el importe se debe rectificar según la modificación introducida. Este trigger es una simplificación del apartado 3 de la práctica anterior y nos sirve como base para el resto de la práctica actual.

Entregar: a) Fichero `actualiza_total_pedidos.sql`. b) Un bloque anónimo para probar el procedimiento para cada uno de los casos posibles (insertar, eliminar y modificar). c) El resultado de la ejecución con lo que devuelve el bloque anónimo.

APARTADO 2.-

Se planea lanzar una campaña de cupones de descuento para cada cliente basados en cada línea de sus pedidos (en tabla `Contiene`). Para ello queremos almacenar los cupones de cada restaurante en una tabla diferente. Además, como esperamos tener más restaurantes que se incorporen al sistema, queremos automatizar el proceso: cada vez que haya una fila de pedido nueva en `Contiene` queremos que se cree para cada restaurante, en tiempo de ejecución, la tabla para sus cupones y el procesamiento necesario mediante los siguientes elementos **(debes crear exactamente estos elementos y generar datos adecuados para probar todos los casos posibles razonables)**:

El concepto `ValorCompleto` que usaremos es igual al precio_con_comisión * unidades (ambos de la fila en tratamiento de `Contiene`)

Fila actual: es la fila en `Contiene` que activó el trigger `actualiza_total_pedidos`

Procedimiento `Gestion_Cupones`

Se llama desde el trigger `actualiza_total_pedidos`, y solo en el caso de insertar una fila en `Contiene`. Genera los cupones a partir del `ValorCompleto` obtenido de los datos mencionados de esa fila.

PASOS dentro del procedimiento:

1. Llama al procedimiento `crea_tabla_cupones` del restaurante "actual"(al que pertenece el pedido de la fila actual).
2. Llama al procedimiento `crea_sec_cupones`: crea la secuencia del restaurante actual
3. Llama al procedimiento `crea_trigger_borra_caducados`, los bonos del restaurante
4. Llama al procedimiento `crea_trigger_extiende_caducidad` de los bonos del restaurante
5. Queremos controlar la cantidad de cupones que tenemos activos siguiendo este proceso
    - Si la suma actual de los cupones del restaurante actual es mayor del 10% del `ValorCompleto`: habilitar el trigger `borra_caducados` y deshabilitar el `extiende_caducidad`.
    - Si dicha suma es menor del 1% del importe total de sus pedidos hacer la operación inversa.
6. Llama a la función pipelined `genera_cupones` de la fila actual con sus datos y el `ValorCompleto`.

7. Cada fila devuelta por la función `genera_cupones` se inserta en la tabla de cupones del restaurante actual. Después se imprime por consola para ver el resultado en las pruebas.

Procedimiento `crea_sec_cupones`

- Comprueba que no esté creada la secuencia (con la Tabla "user_objects"). Si está creada, termina sin hacer nada.
- Crea una secuencia `sec_XXXXXX` para ese restaurante, donde `XXXXXX` es el nombre del restaurante del pedido actual. Se usa para dar un valor único a cada cupón de ese restaurante.

Procedimiento `crea_tabla_cupones`

- Comprueba que no esté creada la tabla (con la Tabla "tabs"). Si está creada termina sin hacer nada.
- Crea la tabla `cupones_XXXXXX` , donde `XXXXXX` es el *nombre* del restaurante del pedido actual,
- Atributos de la tabla:
  `NumCupon` es numérico, tiene un valor secuencial único, generado con `sec_XXXXXX`. Es la clave primaria.
  `dniCiente` es el del cliente que ha hecho el pedido
  `codPedido` es el pedido que ha generado los cupones,
  `valor` cantidad en euros de dicho cupón, sin decimales.
  `FechaCaduca` es la fecha de caducidad del cupón. Por defecto es 10 días después de la fecha actual.

Función pipelined `genera_cupones`

- Queremos generar de forma eficiente los cupones, por ello usamos la función pipelined.
- Genera un cupón de valor 1 euro por cada 5 euros de `ValorCompleto`
- Adicionalmente genera un cupón de 2 euros por cada 25 euros
- Devolver cada cupón generado en el momento de generarlo.

Procedimiento `crea_trigger_borra_caducados`

- Comprueba que no esté creado el trigger (con la Tabla "user_triggers"). Si está creado, termina sin hacer nada.
- Crea un trigger `borra_caducados_XXXXXX` para borrar los bonos caducados del restaurante actual `XXXXXX`
- Características del trigger:
  - Se activa una vez para todas las filas que se inserten en la tabla `cupones_XXXXXX`
  - Borra todos los cupones cuya fecha menor o igual a la de hoy.

Procedimiento `crea_trigger_extiende_caducidad`

- Comprueba que no esté creado el trigger (con la Tabla "user_triggers"). Si está creado, termina sin hacer nada.
- Crea un trigger `extiende_caducidad_XXXXXX` para extender la fecha de caducidad de los bonos caducados del restaurante actual `XXXXXX`
- Características del trigger:
  - Se activa una vez para todas las filas que se inserten en la tabla `cupones_XXXXXX`
  - Cada bono caducado del restaurante actual `XXXXXX`, suma 10 días a la fecha de caducidad.

Entregar: a) Entregar cada procedimiento, función o trigger en ficheros .sql separados. b) Un bloque de código anónimo para probar el proceso para cada uno de los casos posibles. c) Para cada caso probado: una breve explicación de lo que hace, indicando los cambios efectuados en la BD.

APARTADO (EXTRA para mejora la nota)

- Crear una Función de Tabla que tenga un parámetro con el código postal y devuelva el nombre del pueblo o barrio correspondiente (solo en Madrid).
- Transformar el cupón para que tenga el nombre del pueblo o barrio, llamando a la función anterior para obenerlo.

Entregar: a) ficheros .sql. b) Un bloque de código anónimo para probar los elementos creados, para cada uno de los casos posibles. c) Para cada caso probado: una breve explicación de lo que hace, indicando los cambios efectuados en la BD.

APÉNDICE: Ayudas

- Tabla "tabs" contiene todas las tablas del usuario. Con "desc tabs" puedes ver todos los atributos
- Tabla "user_objects" tiene todos los objetos del usuario, incluidas las secuencias.
- Tabla "user_triggers" tiene todos los triggers del usuario.
-

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))
```