Entrega práctica 6: Restricciones e índices.

Grupo: DG04

Autor: Manuel Ortega Salvador.
Autor: Francisco Javier Blázquez Martínez.

Apartado 1.- Disparador por tabla:

a) Creación de tablas:

```
CREATE TABLE pedidos
(
   código char(6) PRIMARY KEY,
   fecha char(10),
   importe number(6,2),
   cliente char(20),
   notas char(1024)
);

CREATE TABLE contiene
(
   pedido char(6),
   plato char(20),
   precio number(6,2),
   unidades number(2,0),
   PRIMARY KEY(pedido, plato)
);

CREATE TABLE auditoría
(
   operación char(6),
   tabla char(50),
   fecha char(10),
   hora char(8)
);
```

b) Disparador para auditar cambios:

```
CREATE OR REPLACE TRIGGER trigger_pedidos
AFTER INSERT OR DELETE OR UPDATE ON pedidos

DECLARE
fechaInsert char(10);
horaInsert char(8);

BEGIN

fechaInsert := to_char(sysdate,'dd/mm/yyyy');
horaInsert := to_char(sysdate,'hh:mi:ss');
```

```
IF INSERTING THEN
   INSERT INTO auditoría VALUES ('INSERT', 'pedidos', fechaInsert, horaInsert);
ELSIF  DELETING THEN
   INSERT INTO auditoría VALUES ('DELETE', 'pedidos', fechaInsert, horaInsert);
ELSIF UPDATING THEN
   INSERT INTO auditoría VALUES ('UPDATE', 'pedidos', fechaInsert, horaInsert);
END IF;

END;

-- Respuesta del servidor:
Trigger TRIGGER_PEDIDOS compilado

-- Para probar el correcto funcionamiento insertamos, modificamos y eliminamos una tupla:
INSERT INTO pedidos VALUES('111111', '16/10/2016', 545, 'Petrenko', 'Hola');
UPDATE pedidos SET cliente = 'Alcatraz' WHERE código = '111111';
DELETE FROM pedidos WHERE código = '111111';

-- El resto de pruebas de su correcta ejecución se realizó a lo largo de los siguientes apartados.
```

Apartado 2.- Disparador por fila:

```
/* Creación del disparador */
CREATE OR REPLACE TRIGGER trigger_contiene
/* Cuando dispararlo */
AFTER INSERT OR DELETE OR UPDATE ON contiene
/* Para cada fila modificada */
FOR EACH ROW

DECLARE
   numeroApariciones INTEGER;

-- Excepciones que vamos a controlar:
   precioPlatoNulo EXCEPTION;
   numUnidadesNulo EXCEPTION;
   pedidoNoEncontrado EXCEPTION;

BEGIN

IF INSERTING THEN

   SELECT count(*)
   INTO  numeroApariciones
   FROM pedidos
   WHERE pedidos.código=:NEW.pedido;

   IF numeroApariciones<1 THEN
      DBMS_OUTPUT.put_line('El código de pedido no se ha encontrado');
      RAISE pedidoNoEncontrado;
   ELSIF :NEW.precio IS NULL THEN
      DBMS_OUTPUT.put_line('Precio del plato no especificado');
      RAISE precioPlatoNulo;
```

```
   ELSIF :NEW.unidades IS NULL THEN
      DBMS_OUTPUT.put_line('Número de unidades no especificado');
      RAISE numUnidadesNulo;
   END IF;

   UPDATE pedidos
   SET    importe = pedidos.importe + :NEW.precio * :NEW.unidades
   WHERE pedidos.código = :NEW.pedido;

ELSIF DELETING THEN

   UPDATE pedidos
   SET    importe = pedidos.importe - :OLD.precio * :OLD.unidades
   WHERE pedidos.código = :OLD.pedido;

ELSIF UPDATING THEN

   IF :NEW.precio IS NULL THEN
      DBMS_OUTPUT.put_line('Precio del plato no especificado');
      RAISE precioPlatoNulo;
   ELSIF :NEW.unidades IS NULL THEN
      DBMS_OUTPUT.put_line('Número de unidades no especificado');
      RAISE numUnidadesNulo;
   END IF;

   UPDATE pedidos
   SET    importe = pedidos.importe - :OLD.precio * :OLD.unidades
                                    + :NEW.precio * :NEW.unidades
   WHERE  pedidos.código = :NEW.pedido;

END IF;
```

Apartado 3.- Creación y uso de índices:

a) Creación del índice:

```
CREATE INDEX index_pedidos ON  Pedidos(cliente);

-- Respuesta del servidor:
Index INDEX_PEDIDOS creado.
```

b)Inserción de tuplas para probar el índice:

```
BEGIN
FOR i IN 1..50000 LOOP

INSERT INTO Pedidos VALUES (i, '06/01/2015', 10.0, 'C' || i, ' ');

END LOOP;
END;
```

-- Respuesta del servidor:
Procedimiento PL/SQL terminado correctamente.

-- NOTA:
-- Se utilizó como límite superior 50.000 en vez de 500.000 por problemas del servidor de
-- asignación de espacio.

c) Contrastar acceso con índice y sin índice:

-- 1.- Selección de todos los atributos, no solo el campo “Cliente” indexado:

```
SET TIMING ON
SELECT * FROM Pedidos WHERE Cliente = 'C50000';
```

---- Con el índice ya creado:
>>Query Run In:Resultado de la Consulta
Transcurrido: 00:00:00.601

---- Sin el índice en funcionamiento:
>>Query Run In:Resultado de la Consulta
Transcurrido: 00:00:00.599

-- 2.- Selección solo de el atributo “Cliente” indexado:

```
DROP INDEX index_pedidos;

SET TIMING ON
SELECT Cliente FROM Pedidos WHERE Cliente = 'C50000';
```

---- Con el índice activo:
>>Query Run In:Resultado de la Consulta 2
Transcurrido: 00:00:00.339

---- Sin el índice en funcionamiento:
>>Query Run In:Resultado de la Consulta 4
Transcurrido: 00:00:00.392

-- 3.- Conclusiones:

En el caso de seleccionar atributos no indexados el efecto del índice es inapreciable o incluso ligeramente contraeficiente. Sin embargo cuando solo nos interesa trabajar sobre estos atributos con un índice que los regula se observa que el tiempo de ejecución de la consulta es menor. El hecho de que la diferencia sea tan pequeña (y de que estas conclusiones tengan poca fuerza) es debido a que la tabla sobre la que opera únicamente consta de 50.000 tuplas.

d) Índices sobre vistas:

-- 1.- Creación del índice sobre una vista no materializada:

```
CREATE VIEW CLIENTES_VIEW AS SELECT Código,Cliente FROM Pedidos;
```

---- Respuesta del servidor:
View CLIENTES_VIEW creado.

```
CREATE INDEX index_over_view on CLIENTES_VIEW(Cliente);

---- Error obtenido:
Error que empieza en la línea: 19 del comando :
create index index_over_view on CLIENTES_VIEW(Cliente)
Informe de error -
ORA-01702: una vista no es apropiada aquí
01702. 00000 -  "a view is not appropriate here"
*Cause:    Among other possible causes, this message will be produced if an
           attempt was made to define an Editioning View over a view.
*Action:   An Editioning View may only be created over a base table.

-- 2.- Creación del índice sobre una vista materializada:
CREATE MATERIALIZED VIEW CLIENTES_MAT AS
SELECT Código,Cliente FROM Pedidos;

---- Respuesta del servidor:
Materialized view CLIENTES_MAT creado.

CREATE INDEX index_over_view on CLIENTES_MAT(Cliente);

---- Respuesta del servidor:
Index INDEX_OVER_VIEW creado.

-- 3.- Conclusiones:
```
Las vistas no son propiamente datos almacenados, son expresiones SQL que nos permiten obtener unos datos a partir de unas tablas. Esto hace que tenga lógica tener índices sobre las tablas (los datos) y no sobre las consultas (expresiones SQL). En el caso de las vistas materializadas, resulta que sí que son propiamente datos almacenados (los que se obtienen de la consulta SQL que los genera), esto es a la vez causa del riesgo de inconsistencia con respecto a la tabla de la que proceden y de la ventaja de poder crear índices sobre estas para hacer consultas más rápidas.