UNIVERSIDAD DE MÁLAGA

ETSI Informática (Gestión)
Administración de B.D.
Relación de Problemas 6
Vistas, Disparadores y Tablas Mutantes

En esta relación de problemas usaremos la **BD de la Relación de Problemas 5**.

1. Cree una vista llamada **`ACTIVIDADES_POR_PAISES`** quesimplifiqueconsultarlasactividades adoptadas por cada país. La vista tendrá 3 atributos: Nombre del País, Capital y Nombre de la Actividadadoptada.

a) Intente **INSERTAR** información en esa vista. ¿Qué ocurre?
b) Intente **ACTUALIZAR** información en esa vista. ¿Qué ocurre?
c) Intente **BORRAR** información en esa vista. ¿Qué ocurre?

Cuestiones:
- ¿Por qué es posible que dicha vista tenga filas repetidas?
- ¿Cómo podría evitarse sin añadir nuevos atributos a la vista?
- Si añade el atributo del nombre de la Región, ¿podrá haber filas repetidas?

2. Programe un disparador de sustitución (**`INSTEAD OF`**) sobre la vista anterior. Este disparador se ejecutará al **insertar** una fila sobre dicha vista. Entendemos que insertar una fila significa que queremos que la región indicada en la nueva fila adopte la actividad que se indica en esa fila, en la fecha actual y con una duración de 1 año. O sea, la sentencia **`INSERT`** sobre la vista se traduce en unasentencia **`INSERT`** sobre la tabla **Adoptan**.

a) Intente **INSERTAR** información en esa vista. Ahora no debería producirse un error.
b) Intente **INSERTAR** de nuevo la misma información y observe el tipo de error que debe producirse: No debe permitir insertar filas duplicadas en la tabla **Adoptan**.
c) Intente **INSERTAR** una fila en la que el nombre del país o el nombre de la actividad no existan en sus tablas correspondientes. Observe que se produce la excepción **`NO_DATA_FOUND`**. Modifique el *trigger* para que capture esta excepción y muestre un mensaje en español: **`'Pais o Actividad no existente.'`**
d) Observe que si **inserta** una fila con la capital mal puesta, esto no genera error, ya que el *trigger* no utiliza ese valor.
e) Por último, si al **insertar** en la vista lo que está mal es el nombre de la región, entonces el error que se produce es por incumplimiento de la llave externa de la tabla donde se está insertando realmente. Para corregir eso puede, simplemente, buscar si dicha región existe en la tabla de **Regiones**. Si no existe se generará también la excepción **`NO_DATA_FOUND`**, que se capturará en la sección **`EXCEPTION`** ya existente: Sólo tiene que cambiar el mensaje para añadir esta otra posibilidad.

3. Añada en la tabla de **Regiones** un atributo **`POBLACION_CENSADA`** que almacene el total de personas censadas en dicha región, y actualice manualmente los datos para cada región. Naturalmente, si se modifica la tabla censo puede que ese atributo se tenga que modificar. Para ello, cree un ***trigger*** **PL/SQL** llamado **`POBLACION_REGION`** que se encargue de esa tarea automáticamente. Este *trigger* se disparará en 3 casos:

a) Al insertar una fila en el censo: Debe sumar 1 a la población censada en la región en la que vivaelnuevoindividuo.
b) Al borrar una fila en el censo: Restará 1.
c) Al actualizar la región en la que vive una persona: Restará 1 a la antigua región y sumará 1 a la nueva región.

Cuestionesaconsiderar:
- Observe que como tenemos que saber los datos de cada fila modificada, este *trigger* debe ser a nivel de: _____
- Pruebe el *trigger* insertando, borrando y actualizando filas en la tabla **Censo**.
- Si inserta una persona en la que el atributo del código del país en el que vive vale **`NULL`**:
  a) ¿Se disparará el *trigger*? _____
  b) ¿Se modifica el atributo **`POBLACION_CENSADA`**? _____
- Observe que si, por algún motivo, el valor de **`POBLACION_CENSADA`** es incorrecto para una región, ese valor seguirá incorrecto siempre, ya que el *trigger* no cuenta la población censada en una región, sino simplemente suma y/o resta 1. La solución podría ser que dicho *trigger* en vez de sumar/restar lo que hiciera fuera contar las personas censadas en la región o regiones afectadas (usando **`COUNT`** sobre la tabla **Censo**).
- Borre el *trigger* anterior y cree uno nuevo similar, para que efectúe la misma operación pero contando las personas censadas en la región o regiones afectadas. ¿Puede crear dicho *trigger*? _____ ¿Funciona correctamente? _ NO _ ¿Por qué?
- El error que se produce se solucionará en el siguiente ejercicio.

4. **Amartya K. Sen** nació en Santiniketan (India) en 1933. En 1998 la academia sueca le concede el premio Nobel de Economía por sus estudios sobre la desigualdad social. En su obra "Nuevo Examen de la Desigualdad"[1] (1992) **Sen** insiste en que el nivel de ingresos no es un buen criterio para valorar el grado de pobreza, sino que es necesario utilizar otros criterios tales como analfabetismo (teniendo en cuenta el sexo), esperanza de vida... sin embargo, los ingresos suelen ser un mecanismo bastante fácil de obtener y utilizar. La **medida de la pobreza P** de **Sen** se obtiene de: **P = H (I + (1- I) G)**

- **H** es el porcentaje de pobres (considerando como tales los que tengan unos ingresos menores al umbral de pobreza de la región en la que vivan).
- **I** es una medida de la cantidad de pobreza que lo interpretaremos como la *media* de las diferencias entre los ingresos y el umbral de pobreza, entre los pobres.
- **G** es el **coeficiente de Gini** que mide la desigualdad que existe entre los pobres, siendo un valor 0 para el caso en el que no exista desigualdad entre los pobres.

Observe que, a pesar de que **H** es muy utilizado, no tiene en cuenta en qué medida son pobres los pobres ni la desigualdad entre ellos. Por eso, son necesarios tanto **I** como **G**.

a) Añada en la tabla de **Regiones** un atributo llamado **`P_SEN`** para el valor **P** de **A.K. Sen**, con la restricción de que sea mayor que cero.

b) Escriba un ***trigger*** **PL/SQL** que calcule los valores de los atributos **`P_SEN`** y **`POBLACION_CENSADA`**, para mantenerlos actualizados. Dichos valores pueden variar *después* de modificar los datos de la tabla **Censo:** al insertar o borrar una persona, o bien si varían los ingresos de alguna persona.

[1] Un resumen de este libro puede encontrarse en: **http://www.lcc.uma.es/~ppgg/libros**

El *trigger* calculará la nueva población censada y los valores **H**, **I** y **G** (usandosentencias **`SELECT..INTO`**). Tras eso modificará la región afectada con los nuevos valores de los atributos **`POBLACION_CENSADA`** y **`P_SEN`**.

- Observe que para calcular estos valores debe conocer la región afectada (debe conocer el valor de los atributos que identifican a la región en la tabla **Censo** de la fila insertada, borrada o actualizada). Esto hace que sea a nivel de: _____
- Ahora bien, para calcular esos valores necesita acceder a la tabla Censo, que es precisamente la tabla sobre la que actúa el *trigger*. Por lo tanto, esa tabla es *mutante* y no puede ser accedida. Compruébelo.

SOLUCIÓN:

1. Crear una tabla llamada **`Region_Modificada`** que almacene la región de la persona modificada (nombre de la región y código del país).
2. Crear un *trigger* a nivel de *fila* que guarde en esa tabla la región que se está modificando (en la sentencia DML sobre la tabla **Censo**). Observe que este *trigger* no tiene problemas de tabla mutante porque NO accede a la tabla **Censo**.
3. Crear un *trigger* de tipo **`AFTER`** a nivel de *sentencia* que:
   a) Lea la región afectada que haya en tabla **`Region_Modificada`**.
   b) Calcule los nuevos datos de los atributos **`POBLACION_CENSADA`** y **`P_SEN`**. para esa región: Al ser a nivel de sentencia, este *trigger* sí puede consultar la tabla Censo.
   c) Modifique la tabla de **Regiones** con esos datos.
   d) Borre todos los datos de la tabla **`Region_Modificada`**.

Inserte y borre ciudadanos del **Censo**, así como actualice sus ingresos para ver si el disparador funciona modificando la tabla de **Regiones**.

NOTA 1: Observe que la **Población censada** y el valor **P** también cambian cuando en la tabla **Censo** se modifica (con **UPDATE**) la región en la que está censada una persona. Además, en ese caso se ven afectadas dos regiones (la vieja y la nueva). Por razones de simplicidad no consideraremos ese caso en este ejercicio. Hay dos soluciones:
   a) Crear otra tabla en la que insertar la segunda región (solución sin usar un cursor).
   b) Insertar las dos regiones afectadas en la tabla **`Region_Modificada`** y tratar ambas con un **cursor** en el disparador a nivel de sentencia. Tenga en cuenta que si en una tabla hay varias filas no pueden tratarse todas con **SELECT..INTO**, sino que hay que usar un **cursor**.

NOTA 2: Observe que si una única sentencia modifica varias filas entonces la tabla **`Region_Modificada`** albergará varias filas y el disparador a nivel de sentencia dará error al producirse la excepción **`TOO_MANY_ROWS`**. La solución es usar un **cursor**.

NOTA 3: El problema de este ejercicio para calcular automáticamente el valor **P** de **Sen**, puede resolverse de otra forma que quizás resulta más sencilla e incluso puede resultar más completa. Se trata de resolver el problema a través de **vistas**. Inténtelo y aprenderá más de lo que se imagina.

5. Añada una columna en la tabla de **Países**, donde se almacene la **`Extensión`** (en Km^2) de cada país. Actualice la información de esa columna en algunos países. Se desea comprobar que la suma de las **extensiones** de todas las **regiones** de cierto país no supere la **extensión** de dicho **país**. Permitiremos que la extensión del país sea mayor que dicha suma, ya que un país puede tener regiones que no estén almacenadas en nuestra base de datos.

   a) Haga una **consulta SQL** que muestre aquellos países que incumplen esa norma tan básica. **Pista:** Usar la función de grupo **`SUM`** y las cláusulas **`GROUP BY`** y **`HAVING`**.

   b) Corrija los datos de la base de datos para que todos los países cumplan la norma y compruebe que es correcto ejecutando la consulta anterior nuevamente, la cual debe obtener un resultado vacío.

6. Las comprobaciones y correcciones del ejercicio anterior no garantizan que una situación errónea pueda volver a producirse. Para garantizarlo deberíamos hacer esa comprobación cada vez que se **inserta** una nueva región, y cada vez que se **actualice** la extensión de una región o de un país. Observe que controlar la modificación de la extensión de un país no sufre del problema de tablas mutantes.

   a) Programe **un disparador sobre Países** que no permita **actualizar** la extensión de un país si esa actualización no cumple la norma anterior. Observe que el disparador debe ser a nivel de fila y se creará sobre la tabla de **Países**. Además, el disparador accederá a la tabla de las **Regiones** para hacer la suma pertinente.

   b) Programe **otro disparador sobre Regiones** que controle que no se incumpla la norma anterior cuando se **inserta** una región y cuando se **actualice** la extensión de una de ellas. El disparador accederá a la tabla de **Países** para hallar la extensión del país involucrado y a la tabla de las **Regiones** para hacer la suma pertinente:

   - ¿Se puede acceder en este disparador a ambas tablas? o, dicho de otra forma, ¿es mutante alguna de ellas? Compruébelo y razónelo.

   - Para resolver el problema de la tabla mutante de **Regiones** debe crear dos disparadores de tipo **AFTER**:
     - Uno a nivel de fila (en el que se guardará qué país o países han sido afectados por algún cambio en sus regiones),
     - Otro a nivel de sentencia (en el que se harán las comprobaciones pertinentes).

NOTA: Fíjese que cuando se borra una región no es necesario controlar que se cumpla esa condición sobre la extensión, ya si antes del borrado se cumplía, también se cumplirá forzosamente después. Supongamos que eso no es así y que el disparador también se ejecuta al borrar una región. En ese caso, si su base de datos tuviera la restricción **`ON DELETE CASCADE`** en la llave externa de **Regiones**, esto implica que al borrar un país se borrarán todas las regiones de ese país. En ese caso, el disparador se ejecutaría al borrar un país, por lo que *en ese caso* la tabla de **Países** sería también mutante: No podríamos acceder a ella para consultar la extensión del país.