AMPLIACION de BASES DE DATOS

(Profesor : Héctor Gómez Gauchía)

Práctica 2 Apdos 2 a 6: PLSQL dinámico con procedimientos y triggers

- Modo de entrega: No se Entrega

(**Se rellena el CUESTIONARIO de la práctica cuando esté disponible en el CampusV)**

- Los conceptos de esta práctica se evalúan en un examen de control (tal y como se indica en la Ficha Docente), cuya fecha se avisará a tiempo.

- RESPUESTAS: **En un fichero nuevo word**, copia el enunciado de cada apartado en él. Después Incluye cada respuesta en el apartado correspondiente**. También**:

- *Incluye las filas que están implicadas en cada apartado, ya sea porque estaban provocando un problema o porque las hayas insertado porque te lo pedía.*
- Incluye el contenido de los ficheros .sql de los apartados con PLSQL o con instrucciones sql.
- Para organizar bien el fichero, puedes insertar saltos de página.
- Siempre que trabajes, haz Lista de Dudas concretas para consultar con el profesor, online o por email o en clase o en el laboratorio.

➔ ***Conservar Respuestas para ponerlas en el Cuestionario***

MATERIALES para consultar al hacer la práctica

- **PISTAS**: en la carpeta de esta práctica / prac2enu-S4-PISTAS-CV.pdf
- Teoría: Tema PL/SQL / Desarrollo-PL-SQL.pdf
- Ejemplos de PLSQL triggers y procs.: Tema PL/SQL / Ejercicios . . . / PLSQL-ejemplos/ . . . / PLSQLejemplosEjecutablesTeoria
- Ejemplos de PLSQL Dinámico : Tema PL/SQL / Ejercicios . . . / PLSQL-ejemplos/ . . ./ PLSQL-dinamico-para-CV.rar

APARTADO 2: Preparar TABLAS:

(es el resultado del apdo 1 de semana pasada: si no lo has hecho doy una solución)

→ **Partir la tabla** `Invierte` **en las tablas** `inversiones_XXXXXX` **para cada empresa** `XXXXXX`

Si has hecho la práctica 2 apdo 1, ya tendrás las nuevas tablas `inversiones_XXXXXX` con datos.

Si **no** la has hecho: ejecuta la solución, que está subida en el campus V, en el orden adecuado. Creas todos los procedimientos y después ejecutas el `partir_tabla_invierte`

NOTA: Si has borrado el contenido de la tabla Invierte, vuelve a insertar las filas que tenías antes de hacer este apartado.

APARTADO 3: Crear un trigger para cada tabla `inversiones_XXXXXX` para acumular todo lo insertado en cada tabla en una nueva tabla `SumaEmpresa`. Sigue los pasos en el orden indicado:

1º.- Crear a mano en el editor (hoja de trabajo) una única tabla vacía `SumaEmpresa`, que tiene los atributos (`nombreE`, `Cantidad`). Queremos que tenga una fila por cada empresa con la suma de las cantidades de todas las inversiones de esa empresa. Se actualiza mediante triggers automáticamente. Siguiendo las indicaciones del paso **2º**, hay que crear un trigger para cada tabla `inversiones_XXXXXX` , ya creada en el apartado anterior.

2º.- Para crear los triggers lo hacemos en tres fases:

a) En plsql Estático, crear un trigger `trig_suma_prueba` que se activa después de insertar una fila en la tabla `inversiones_EMPRESA22`. El trigger usa la cantidad de esa fila para actualizar la fila de la `EMPRESA22` en la tabla `SumaEmpresa`. Si no existe fila para esa empresa, la crea con esa cantidad. Probar este trigger antes de continuar.

b) Hacer un procedimiento `crear_trig_suma(NombreEmpresa)` que, usando PLSQL Dinámico, generaliza la creación del trigger anterior para la empresa que tenemos en el parámetro (la llamaremos XXXXX). Es decir, el procedimento crea un trigger igual que el del apartado anterior solo que el nombre será `trig_suma_XXXXX`, que se activa al insertar fila en `inversiones_XXXXX` y la fila que actualiza en la tabla `SumaEmpresa` es la de la empresa XXXXX .

c) Hacer un procedimiento `crear_triggers` que haga lo siguiente: crear un trigger `trig_suma_XXXXXX` para cada tabla `inversiones_XXXXXX`. Para ello llama al procedimiento `crear_trig_suma(NombreEmpresa)` con el nombre de empresa `XXXXXX` adecuado.

Para saber los nombres `XXXXXX`, de las empresas que tienes actualmente, debes obtenerlas de la tabla `tabs` , donde están todas las tablas cuyo nombre empieza por 'inversiones_' y luego lo cortas para quedarte solo con los nombres de las empresas.

Antes de crear el trigger, comprueba que no existe, consultando la tabla `user_objects`.

3º.- Para probar estos triggers crea un procedimiento `pru_tri_suma` que haga lo siguiente:

- Borrar todas las filas (no la tabla) de la tabla `SumaEmpresa`.
- Ejecuta un bucle para borrar todas las filas (no la tabla) de cada tabla `inversiones_XXXXXX` .
- Ejecuta `partir_tabla_invierte` (se dispara cada trigger actualizando tabla `SumaEmpresa`)
- Imprime cada fila de la tabla `SumaEmpresa`.

APARTADO 4: hacer un trigger que sirva de log de las operaciones en la tabla `SumaEmpresa`

1º.- Crea en el editor una tabla vacía `LogSumas`, que tiene estos atributos: `fecha y hora del sistema, Operacion, Cantidad y NombreE`. La fecha y hora es del tipo DATE y las obtienes con la función sysdate. Si quieres más precisión (para usar ese atributo como PK) usa el tipo timestamp y función systimestamp. `Operacion` es el nombre de la operación "INSERT" o "UPDATE" que activa el trigger `tri_log`, descrito a continuación.

2º.- Escribe un trigger `tri_log` que

- Se active con cada inserción en la tabla `SumaEmpresa` y que haga lo siguiente:
  - Crea una fila en la tabla `LogSumas`, con los atributos adecuados indicados antes.
- Se active con cada update en la tabla SumaEmpresa y que haga lo siguiente
  - Crea una fila en la tabla `LogSumas`, con los atributos adecuados indicados antes.

En ambos casos debe dar una excepción y un mensaje estandar si hay una cantidad superior a cien millones de euros.

3º.- Para probar este trigger:

- Modifica el procedimiento `pru_tri_suma` para que lo primero que haga es comprobar si está deshabilitado el trigger `tri_log` : si lo está, debe habilitarlo.

- Ejecuta el procedimiento `pru_tri_suma` y comprueba el resultado.
- (en el editor) Añade una inversión con cantidad superior a cien millones de euros para probar la excepción.

APARTADO 5: (PARA NOTA) Provoca un ciclo en la activación de triggers

- Crea otro trigger que, junto con los anteriores ya creados provoque un ciclo en la activación.
- Prueba el ciclo ejecutando el procedimiento `pru_tri_suma`.
- Describe lo que pasa.

APARTADO 6: (PARA NOTA) No se deben repetir filas el proc. `partir_tabla_invierte`

Comprueba que, si se ejecuta dos veces `partir_tabla_invierte`, no debe insertar dos veces la misma fila (aunque tenga el número secuencial distinto, se considera que repites fila).