AMPLIACION de BASES DE DATOS

(Profesor : Héctor Gómez Gauchía)

Práctica 3 Apartado 2: (CONT. Transacciones), niveles aislamiento, bloqueos implícitos y explícitos, junto con aspectos de administración (privilegios, usuarios, tablas públicas, gestión espacio libre en tablespaces, etc.)

- ***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 el Examen Final

- ***Conservar Respuestas para ponerlas en el Cuestionario*: 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. Poner solo texto, NO poner capturas de pantalla.*
  - 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.

MATERIALES para estudiar la práctica (ver más detalles en *ABD-MATERIALES de cada semana.pdf***)**
- Ayudas: ver en la carpeta de la práctica: prac3enu-Apdo-2-Transacciones-PISTAS.pdf
- Teoría: Tema Transacciones / Teoria-Transacciones-y-Concurrencia.pdf
  / Transacciones-Posibles-Situaciones.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.zip

→ **Recuerda durante toda la práctica usar en el editor** `set serveroutput on` y el `set autocommit off`

→ **Ejecuta la BDejemplo.sql que está en PRAC3-Apdo-1 para empezar con la BD adecuada**

APARTADO 2.- Rollback, savepoints, bloqueos, transacciones autónomas, y aspectos de administración (privilegio, usuarios, espacio libre en tablespaces, etc.)

a.- ¿Cómo funcionan varios aspectos: usuarios, gestión de espacio, tablas públicas, triggers, procedimientos?

Queremos dar una comisión a **todos** los alumnos de ABD por cada nueva inversión (inserta en tabla **`Invierte`**) que hagas en tu usuario. Para probar, antes, lo haremos solo para tu usuario. Cada alumno tiene una tabla **`Notificaciones`** donde se inserta una fila por cada comisión que reciba. La tabla tiene el mismo nombre en todos los usuarios.

Para ello se pide lo siguiente:

--- PREPARACIÓN ---

a.1— Desde el editor del SQLDeveloper, dentro de tu usuario, crea una tabla:

```
Notificaciones(Usuario_Origen, Fecha, DNI_cli, NombreE, Tipo , Comision)
```

Para tener todos el mismo formato, ejecuta **crearNotificaciones.sql** , que está en la carpeta de esta práctica. Recuerda que funciona con la BDejemplo.sql de la PRAC3-Apdo-1 que debes haber ejecutado

Donde:

`Usuario_Origen` es el usuario que está enviando la notificación (el que ejecuta el procedimiento)
`Fecha`: se obtiene con la función systimestamp , la fecha y hora del sistema del momento de crear la notificación
`Comision`: Calculada respecto a la cantidad de la inversión como se indica debajo.
El *resto de atributos* son los de la inversión que activa el trigger.

a.2-- Para que cualquiera pueda insertar filas en esa tabla, da permiso al público.

--- DESARROLLO y PRUEBAS ---

(para funciones que no conoces: ver el fichero prac3enu-Apdo-2-Transacciones-PISTAS)

a.3-- Hacer un trigger **`prac32`** que se ejecuta después de que se crea una inversión en la tabla **`invierte`**. Este trigger solo llama al procedimiento **`autoRegalaComisiones (parámetros necesarios)`**, ver detalles de este procedimiento a continuación. El trigger puedes probarlo después de probar el procedimiento.

a.4—Hacer el procedimiento **`autoRegalaComisiones(parámetros necesarios)`**, en el que te das a tí mismo una comisión así: escribe una fila en la tabla **`Notificaciones`**, con comisión del 2% del importe de la inversión actual (la que dispara el trigger) siempre que cumplas estas condiciones siguientes:

- (para NOTA) Si tiene menos 1800 bloques de espacio libre en su tablespace.

PRUEBA este procedimiento, haciendo que se dispare el trigger **prac32**: haz un insert en la tabla invierte.

a.5-- Hacer el procedimiento **`RegalaComisiones(parámetros necesarios)`**, para **todos** los usuarios de ABD. Es una generalización del **`autoRegalaComisiones`**. Debe hacer lo siguiente:

- Obtener todos los usuarios de Oracle de la asignatura ABD.
- A todos le damos una comisión por cada nueva inversión, siempre que se cumplan ciertas condiciones.

Así que hacemos lo mismo que en el apartado anterior, solo que en vez de ser para tu usuario, lo hacemos para cada usuario obtenido: escribiendo una fila en la tabla **`Notificaciones`** que hay dentro de *cada usuario*, siempre que el usuario cumpla las condiciones siguientes:

- Si existe la tabla **`Notificaciones`** de ese usuario (hay usuarios que no tendrán la tabla).
- (para NOTA) Si tiene menos 1800 bloques de espacio libre en su tablespace.
- (para NOTA) Si hacemos un sorteo con un número aleatorio (que obtenemos una vez para cada usuario) y coincide con la última cifra del nombre de usuario de Oracle.

PRUEBA este procedimiento: antes, recuerda cambiar en el trigger prac32, la llamada a **RegalaComisiones**

b.- ¿Cómo funcionan los Savepoints, Rollback y Pragma autonomous Transactions?

Copia el procedimento **`RegalaComisiones`** a uno nuevo **`RegalaComisiones_b`** , y cambia su nombre en el create, para que incluya lo siguiente:

- Asumiendo que cada usuario en el bucle, está recibiendo simultáneamente comisiones de varios usuarios: si, después de insertar nuestra comisión comprobamos que tiene ya comisiones que suman más de 100 euros, entonces deshacemos la nuestra con rollback.

Problema: Como el procedimento **`RegalaComisiones_b`** se ejecuta desde el trigger prac32, no se puede hacer un rollback.

Solución: (para ver cómo se hace, consulta apdo. **f.-** : Hago la transacción *autónoma*, modificando el trigger prac32 y terminando dicha transacción en el trigger. Además incluye en el bloque que uses de prueba (después del insert en **`Inversiones)`**, en el trigger y en el **`RegalaComisiones_b`** esto: imprimir en qué transacción está (cópialo desde pone_linea_autonoma.sql).

Así podrás ver que si son transacciones diferentes o iguales.

PRUEBA este procedimiento: antes, recuerda cambiar en el trigger prac32, la llamada a **RegalaComisiones_b**

c.- ¿ Cómo funcionan los Rollback?

Copia el procedimento **`RegalaComisiones_b`** a uno nuevo **`RegalaComisiones_c`** , y cambia su nombre en el create, para que incluya lo siguiente:

- Después de haber hecho el proceso para todos los usuarios, si hemos dado más de 1000 euros de comisiones en total, deshacemos todo y volvemos a repetir el proceso entero (como mucho hasta tres veces en total) hasta que el total sea menor de 1000 euros.

PRUEBA este procedimiento: antes, recuerda cambiar en el trigger prac32, la llamada a **RegalaComisiones_c**

d.- ¿Cómo funcionan los bloqueos explícitos de tabla?

Copia el procedimento **`RegalaComisiones_c`** a uno nuevo **`RegalaComisiones_d`** y cambia su nombre en

el create, para que incluya lo siguiente:
-- Mientras se está haciendo el proceso del apartado anterior (*si hemos dado más de 1000 euros..)* no queremos que nadie modifique la tabla, así que bloquea tú explícitamente esa tabla Notificaciones del usuario al que escribimos, para que nadie la modifique mientras haces el proceso.

Cuando pruebes puede que tu proceso se quede esperando porque haya otros usuarios que han bloqueado alguna tabla.

e.- ¿Cuando terminan los bloqueos de tabla?

En el apartado anterior, el problema es que se quedan bloqueadas todas las tablas. ¿con que instrucciones lo resuelves?. Es razonable resolverlo?

f.- Para entender: ¿cómo funcionan las transacciones autónomas?

-- Ejecuta el procedimiento TRAB_T_1_LINEA_AUTONOMA.sql (instrucciones ejecución al final de ese mismo archivo script) que llama al procedimiento PONE_LINEA_AUTONOMA.sql
- ¿Qué números de transacción da?
- ¿Porqué hay varias transacciones?

g.- (para NOTA) **¿ Cómo funcionan las transacciones autónomas?**
Modifica el trigger **`prac32`** para que incluya lo siguiente:
g.1.- Queremos que, aunque se hiciera rollback de la inversión (en tabla Invierte), la comisión no se anulara
g.2.- ¿Qué tendrías que poner para que este trigger se active después de otro supuesto trigger **`tr_anterior`** sobre la misma tabla Invierte (deja comentada la instrucción, puesto que tr_anterior no existe)?
g.3.- Haz un procedimiento **`Prueba_Autónoma`** para comprobar resultado: inserta en tabla invierte para que se active el trigger y consultando datos de número de transacción en varios momentos, después haz rollback y comprueba qué se mantiene y qué se ha quitado.

---------- **APLICACIÓN de LA TEORÍA : Responde a estas preguntas** ----------

h.- Qué características de una instalación con varias transacciones hacen que una transacción deba ser serializable

i.- Qué características de una instalación con varias transacciones hacen que una transacción deba ser read commited

j.- ¿Qué nivel de aislamiento u otro recurso aplicarías para estas situaciones?:

j.1.- Unas transacciones actualizan la tabla ValoresBolsa. Qué nivel de aislamiento se necesita en otra transacción para que vea cuanto antes los últimos cambios? ¿Harías algo en esas transacciones para que se vean los cambios cuanto antes? . Explica el porqué.

j.2.- Unas transacciones actualizan la tabla ControlPuerta, que puede estar abierta, cerrada o bloqueada. ¿Qué harías para que cada transacción vea el estado inmediatamente después de la actualización de la otra transacción? Explica el porqué.

j.3.- Unas transacciones actualizan la tabla MovtosVisa. Nuestra transacción hace el cierre fin de mes a las doce de la noche del último día, contabiliza y marca cada movimiento como procesado. Y al final suma el total de los movimientos marcados. Qué nivel de aislamiento necesitamos? Explica el porqué.

j.4.- Unas transacciones actualizan la tabla MovtosVisa. Nuestra transacción solo lista los movimientos entre dos fechas. Qué tipo de transacción definiremos para agilizar el proceso? Indica la instrucción.

k.- ¿Qué resultado dará cada situación descrita?:

k.1.- En este trozo de procedimiento: ¿qué sucede con el segundo rollback?
Comentas las posibles situaciones que se pueden dar según el valor de fallo1 y fallo2

```
Savepoint SP1;
    Proc-update1();
Savepoint SP2;
     Proc-update2();
IF fallo1 then rollback savepoint SP1 end if;
IF fallo2 then rollback savepoint SP2 end if;
```

k.2.- En este trozo de procedimiento: ¿qué sucede con el segundo rollback?

```
Savepoint SP1;
 Update . . . .
  Rollback;
 Update. . . . .
 Rollback to savepoint SP1;
```