

AMPLIACION de BASES DE DATOS

(Profesor : Héctor Gómez Gauchía)

Práctica 3 SEMANA 5: Transacciones, nivel aislamiento, bloqueos implícitos y explícitos

Resultado:

- Instrucciones y Resultados de las pruebas de cada apartado en un fichero de texto.
- los ficheros .sql de lo que desarrolles en sql y plsql,
- Haz lista de dudas concretas sin resolver sobre tus respuestas para consultar con el profesor

Modo de entrega: No se Entrega

- Si terminas alguno de los apartados indicado avisa al profesor para puntuar la parte de participación en clase (**ONLINE**: ver *prac3-AP1-semana5-INSTRUCCIONES-para-ser-evaluado.pdf*).
- Los conceptos de esta práctica se evalúan en el examen final (tal y como se indica en la Ficha Docente), cuya fecha se avisará a tiempo.

MATERIALES para estudiar la práctica

- 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

→ Recuerda durante toda la práctica usar en el editor:

`set serveroutput on y el set autocommit off`

→ Ejecuta la BDejemplo.sql que está en esta práctica para empezar con la BD original.

→ Para entender los conceptos de la práctica se recomienda hacer primero el “APARTADO ANEXO”

APARTADO 1.-

- a).- Hacer pruebas repetibles y comprobables (*(NO uses plsql dinámico en esta práctica)*)
(si haces un Diag. de Secuencia , entenderás mejor los apartados) ← *ver el .pdf*

Para ver los efectos de los mecanismos de Oracle necesitamos simular que las transacciones trabajen durante un periodo de tiempo controlado, pudiendo paralarlas y arrancarlas en los momentos precisos para provocar concurrencia a las tablas y situaciones concretas. Para ello vamos a hacer:

- Crear una secuencia (busca el tipo “cycle sequence”) *sec_Ti*, cuyo valor mínimo es 0 y máximo es 1. La uso como un semáforo. Cada vez que sumo 1: si estaba en 1 vuelve a 0. Es la que decide si el bucle del *trabajando_Ti* continúa o se detiene (ver a continuación). En realidad queremos crear una secuencia para cada transacción (T) a simular: *sec_T1*, *sec_T2*, etc. *Prueba que funciona como esperas.*
- Crear un procedimiento *trabajando_Ti* (*X es el núm.de segundos*) que se queda en un bucle infinito, simulando que trabaja, hasta que indicamos que se pare. Para cada T a simular, hacemos un procedimiento distinto: *trabajando_T1 (X)* , *trabajando_T2 (X)* , etc. El pseudocódigo de cada procedimiento es este:
Dentro de un bucle tenemos estos pasos: (empieza con el nombre *i* = 1)
 - Llama a este proc.: *ABDMIUTIL.dormir (núm.segundos)* mantiene parada la T durante esos segundos: (no necesitas ver el contenido, es un proc. que está en el usuario *ABDMIUTIL*)
(**ONLINE**: si usas tu portátil necesitas crear el proc. *dormir*: ver archivo *crear_sinonimos.pdf*).
 - Comprueba si debe terminar de trabajar usando *sec_Ti*: si ve que no, vuelve al principio del bucle y repite el ciclo.

- La forma de indicarle que termine es usando la secuencia *sec_Ti*. Cuando modifiquemos *sec_Ti* desde otra T.: la T. de *sec_Ti* terminará el bucle y el procedimiento:
 - para probar si termina: abre tú otro SQLDeveloper nuevo y conectaté con tu usuario: cambia tú el valor de *sec_Ti* y comprueba que se para el proc.
- Para facilitar el seguimiento: lo último que hace el procedimiento es dar el mensaje “he terminado de trabajar” junto con el número de la transacción donde estaba (ver *que-num-trans.sql*).

b).- Probar el procedimiento *trabajando_T1* del siguiente modo:





- En el editor: crear una secuencia cíclica *sec_T1*.
- Hacer un procedimiento nuevo *probarMiT1*, que incluye estos pasos:
 - Empezar una T con INSERTs de tres COMPRAS: formato igual que en *BDejemplo.sql*
 - Parar la T: llamando a *trabajando_T1* (5). (se dormirá hasta que le demos la orden)
 - Continuar la T con otros INSERTs de otras tres COMPRAS
 - Después, parar la T de nuevo, poniendo una 2ª llamada a *trabajando_T1* (5) (se vuelve a dormir hasta otra orden).
- Ejecutar *probarMiT1*: ¿Se para?
- Desde otra transacción (abre otro sqlDeveloper) queremos *ordenar que continúe*: hacer una modificación de la secuencia *sec_T1* para provocar que *probarMiT1* continúe hasta la 2ª llamada. ¿Se para?
- Si volvemos a hacer otra modificación a la secuencia: ¿el *trabajando_T1* continúa y termina?
- Comprueba tú que ha insertado las filas esperadas.

c).- Probar el procedimiento *trabajando_T1* con dos Ts: → Así probaremos los otros Apartados de la Prác.

- Ahora simula dos Ts concurrentes:
 - falta hacer un nuevo procedimiento *probarMiT2*, el mismo contenido que el *probarMiT1* salvo:
 - que llame a un nuevo procedimiento *trabajando_T2* (5)
 - que tenga una nueva secuencia *sec_T2*.

Este procedimiento *probarMiT2* se ejecuta en otra copia nueva del sqlDeveloper.
En otra copia del sqlDeveloper ejecutamos *probarMiT1*

Las ordenes de continuar se las damos desde una tercera copia del sqlDeveloper(será una T3) haciendo estas operaciones a mano:

- Alterna las ordenes de continuar de la T1 y de la T2 hasta que terminen ambas.
- Consulta qué filas de la tabla ve cada T antes de confirmar.
- Haz un *commit* a mano en la T1.
Comprueba en qué transacción está esa T: → ver *que-num-trans.sql*
- Comprueba ahora qué filas ve la T2 en la tabla.  
- Haz un *commit* a mano en la T2.
Comprueba en qué transacción está esa T
- Comprueba ahora si en la tabla están las filas esperadas.  

Estás preparado para ver la diferencia con otro nivel de aislamiento (en el siguiente apartado)

d).- Repetir el mismo experimento que c), poniendo el nivel de Aislamiento Secuenciable en los dos procedimientos *probarMiT1* y *probarMiT2*. Debe haber diferencias, indica cuáles has encontrado.

APARTADO ANEXO.- Para entender los conceptos de la práctica se recomienda hacer los ejemplos dados en clase siguiendo estos pasos:

Para simular dos transacciones concurrentes:

- Abre dos SQLDeveloper, cada uno será una sesión con una o varias transacciones (siempre usa el `SET AUTOCOMMIT OFF`).
- Prueba los siguientes archivos, copiando cada instrucción, *una a una*, y comprobando el efecto del contenido que queda en las tablas:
 - Situaciones de multiprogramación y bloqueos implícitos: *Transacciones-Posibles-Situaciones.pdf*
 - Comprueba los valores de las tablas en cada paso: Porqué tienen esos valores?
 - Varias pruebas con distintos niveles de aislamiento *probarRollback.docx* (escribe el resultado en cada línea según la ejecutas, para entender porqué):
 - Comprobar en qué número de transacción estoy y cómo se usa el proc dormir (): *que-num-trans.sql*