BASES DE DATOS Fecha Entrega 12-05-2016

Profesor: Héctor Gómez Gauchía

Práctica: Transacciones, niveles aislamiento, bloqueos implícitos y explícitos

→ ***Entrega en cada apartado: el código, las pruebas con experimentos, respuestas a las preguntas, una breve explicación de lo que haces y tus conclusiones.***

Antes de hacer la práctica se recomienda hacer los ejemplos dados en clase:
Para simular dos transacciones concurrentes abre dos ejecuciones del SQLDeveloper, cada una será una sesión con una o varias transacciones (siempre usa el `SET AUTOCOMMIT OFF`). Prueba estos archivos, copiando una a una cada instrucción y comprobando el efecto del contenido que queda en las tablas:
- Situaciones de multiprogramación y bloqueos implícitos: *Tema-3-Transacciones-Posibles-Situaciones.pdf*
- Varias pruebas con distintos niveles de aislamiento: *probarRollback.docx*
- Comprobar en qué número de transacción estoy y cómo se usa el proc dormir ( ): *que-num-trans.sql*

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

APARTADO 1.-

a).- Hacer pruebas repetibles y comprobables

Para ver los efectos de los mecanismos de Oracle necesitamos simular que las transacciones trabajen durante un periodo de tiempo controlado, pudiendo pararlas y arrancarlas en los momentos precisos para provocar concurrencia a las tablas y situaciones concretas.
Por ello vamos a hacer un procedimiento simulador *trabajando_trans_i (X núm.de segundos)*. Para cada T a simular hacemos un procedimiento y una secuencia *distintos,* por eso la "*i*" se sustituye por 1, 2 o el número indicado, **no** uses plsql dinámico, basta con copiar código y cambiar valores y nombre. El pseudocódigo es este:

- *Dentro de un bucle*:
  - Llama al procedimiento *hector.dormir (núm.segundos)* para detener el proceso durante esos segundos:
    Lo único que hace es llamar a dbms_lock.sleep (lapsus in number) que necesita un permiso especial.
  - Comprueba si ya debe salir: si ve que no, vuelve al principio del bucle y repite el ciclo.
  - La forma de indicarle que salga es usando una secuencia *sec_trans_i*, que crearemos fuera, cuyo valor mínimo es 0 y máximo es 1. Cuando la modifiquemos desde otra T., termina el bucle.
  - Para hacer el seguimiento, lo último que hace el procedimiento será dar el mensaje "he terminado de trabajar" junto con el número de la transacción donde estaba.

b).- Probar el procedimento *trabajando_trans_1* del siguiente modo*:*
- Hacer un procedimiento *probarMiTrans1*, que incluye
  - Empezar una T con INSERTs de tres pedidos: formato igual que en *carga_datos_restaurantes.sql*
    (no importa que tengan valores iguales porque la clave se genera de modo secuencial)
  - Parar la T, llamando a *trabajando_trans_1 (5).* (se dormirá hasta que le demos la orden)
  - Continuar la T con otros INSERTs de tres pedidos (no importa que tengan valores iguales)
  - Después para la T de nuevo, poniendo una 2ª llamada a *trabajando_trans_1 (5)* (se vuelve a dormir hasta otra orden).

- Ejecutar *probarMiTrans1*, y, desde otra transacción (abre otro sqlDeveloper) hacer una modificación de la secuencia para provocar que *probarMiTrans1* continúe hasta la 2ª llamada. Si volvemos a hacer otra modificación a la secuencia, continuará y terminará.

- Ha insertado lo esperado?

c).- Probar el procedimento *trabajando_trans_1* con dos Ts: → *Así probaremos los otros Apartados de la Prác.*

- Ahora simula dos Ts concurrentes: falta hacer otro procedimiento *probarMiTrans2* (el mismo contenido que el *1*) que llame a un nuevo *trabajando_trans_2 (5)* que tenga una nueva secuencia *sec_trans_2*. Este procedimiento se ejecuta en otra copia nueva del sqlDeveloper.

Las ordenes de continuar se las damos desde una tercera copia del sqlDeveloper:
- Alterna las ordenes de continuar de la Trans_1 y de la Trans_2 hasta que terminen ambas.
- Consulta qué filas de la tabla de Pedidos ve cada Ts antes de confirmar.
- Haz un *commit* a mano en la 1ª T.
- Comprueba ahora qué ve la 2ª de la tabla de pedidos.
- Haz un *commit* a mano en la 2ª T.
- Comprueba ahora si en la tabla de pedidos están las filas esperadas.

d).- Repetir el mismo experimento que c), poniendo el nivel de Aislamiento Secuenciable en los dos procedimentos *probarMiTrans1 y probarMiTrans2*. Debe haber diferencias, indica cuáles has encontrado.

APARTADO 2.-

a) Modifica probarMiTrans1 para incluir:
- Después de los tres primeros inserts un SAVEPOINT voy_Por_La_Mitad
- Después de los tres últimos inserts un ROLLBACK TO SAVEPOINT voy_Por_La_Mitad

b) Prueba el procedimiento:
- ejecutándolo, avanzando hasta que termine
- Consulta ahora qué núm. Trans. está activa : ¿es la misma que al terminar el proc?
- Haz commit ahora
- Consulta los pedidos de la tabla: ¿están solo los tres primeros?

APARTADO 3.-

a) Queremos hacer una T con un procedimiento para actualizar solo los platos que son de categoría pizza, la cambiamos por pizzaXX. A continuación queremos imprimir los datos de los platos cambiados. Antes de empezar la actualización queremos bloquear solo esos platos para que otras Ts puedan leer pero no modificarlos. *Justifica tu solución.*

b) Prueba el procedimiento en dos Ts concurrentes: en una está el procedimiento y dentro de él lo paramos después de la instrucción adecuada. Y en la otra está un procedimiento que intenta leer un plato de categoría pizza, y lo paramos también. Ejecútalos activando la T. correspondiente para comprobar si el comportamiento es como se desea.

APARTADO 4.-

a) Queremos hacer una procedimiento con varias consultas, escoge *el tipo de transacción* adecuada: queremos permitir que otras Ts puedan consultar el contenido mientras el proc está ejecutándose. Las consultas son: una con los platos del restaurante telericatorta y otra con sus horarios. Imprime sus contenidos. *Justifica tu solución.*

b) Prueba el procedimiento en dos Ts concurrentes: en una está el procedimiento y dentro de él lo paramos entre instrucción e instrucción. Y en la otra está un procedimiento que intenta leer un plato, lo paramos y luego intenta leer los horarios.

APARTADO 5.-

a) Queremos hace un procedimiento en el que suba el precio un 15% a los platos de menos de 10 euros y un 20% al resto. Después, queremos sumar todos los precios nuevos e imprimir la suma. Mientras se hace el proceso no queremos que ninguna T pueda modificar ni leer ningún plato aunque sí pueda acceder a otras tablas. *Justifica tu solución.*

b) (para las dos Read commited) Prueba el procedimiento en dos transacciones concurrentes: en una está el procedimiento y dentro de él lo paramos entre instrucción e instrucción. Y en la otra está un procedimiento que intenta leer un plato, lo paramos y luego intenta actualizar un plato y lo paramos de nuevo. Captura las excepciones adecuadas para que el experimento llegue hasta el final.

c) Repite el experimento para las dos Serializable. *Describe las diferencias de comportamiento*.

APARTADO 6.-

a) Queremos hacer un trigger que se ejecute en una T independiente. Se activará cada vez que se actualice el precio de un plato. El trigger insertará una fila en una tabla nueva *LogPrecios* en la que queremos registrar cada una de las modificaciones del precio del plato. Sus atributos: Nombre plato, precio antigüo, precio nuevo, fecha actualización. La clave primaria serán todos los atributos. *Justifica tu solución.*

b) Probar el trigger, ejecutando el apartado anterior de subir los precios a los platos. Añade lo necesario para que se produzcan las cuatro situaciones posibles: T. principal (el proc de actualización) y T. autónoma (el trigger) terminen bien, una de ellas termine con rollback o las dos terminen con rollback. Comprueba que el resultado es correcto o si sobra o falta alguna fila. Deben quedar registrados los cambios de precio que se han deshecho con un rollback en el proc?

c) Prueba lo mismo desde dos sesiones diferentes y poniendo en las dos trans (trigger y proc) que sean secuenciables. Describe lo que pasa y qué datos quedan en cada situación. Qué pasa? Son correctos?

d) Repasa tu práctica de triggers y analiza si los resultados serían correctos en las cuatro situaciones del apartado **b)**.

APARTADO ULTIMO.- Introduce contenidos en las wikis: (no entregar nada)

a).- Posibles preguntas de examen sobre el tema de plsql y del tema de transacciones. Mejor si tienes alguna respuesta, y la incluyes.

APARTADO (OPCIONAL) → Informe sobre el paquete DBMS_LOCK, que es el sistema interno de Oracle

→ Debes pedir al profesor que le de un privilegio a tu cuenta.

Sirven para diversos recursos.

a) Busca en internet información y escribe un resumen breve:
- para qué se usan
- cómo se usan con ejemplos.
- Qué diferencia hay en relación con el uso de los niveles de aislamiento y los locks.

b) Describe y prueba algunos de los ejemplos.