# AMPLIACION de BASES DE DATOS

( Profesor : Héctor Gómez Gauchía)

## Práctica 3 Apartado 1: Transacciones, nivel aislamiento, bloqueos implícitos y explícitos

***- 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**
- 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

### APARTADO 0

**→ 1.- Recuerda durante toda la práctica usar en el editor:**
`set serveroutput on size 10000` y el `set autocommit off !!!!`

**→ 2.- Ejecuta la BDejemplo.sql que está en esta práctica para empezar con la BD correcta.**

**→ 3.- Para entender los conceptos de la práctica se recomienda hacer ANTES el "APARTADO ANEXO"**

**→ 4.- Como las anteriores, esta práctica se ejecuta en el servidor de la Facultad:**
**→ IMPORTANTE: ver Nota al final de este documento**

### 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) ← como el que se vió en clase

Para ver los efectos de los mecanismos de Oracle, necesitamos simular que las transacciones T***i*** (i es 1, 2, etc.) trabajen durante un periodo de tiempo. Controlando este tiempo, podemos pararlas y arrancarlas en los momentos precisos para provocar concurrencia a las tablas y situaciones concretas. Para ello vamos a hacer lo siguiente:

→ Crear en la Hoja de Trabajo, una secuencia *sec_T1* (busca el tipo de secuencia "cycle sequence Oracle" en internet, debe tener la opción NOCACHE). Va a tener un valor mínimo de 0 y máximo de 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 proc. *trabajando_T1* continúa o se detiene (ver más adelante). En realidad queremos crear una secuencia para cada transacción (T) a simular: sec_T1, sec_T2, . . . sec_T***i*** . *Prueba que funciona como esperas*.

→ Crear un procedimiento *trabajando_T1 (X)*, donde *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

nuevo: *trabajando_T1 (X) , trabajando_T2 (X), . . . trabajando_Ti (X)* El pseudocódigo de cada procedimiento es este:

*Hay un bucle* infinito con estos pasos:

- Llama al proc.: *ABDMIUTIL.dormir (núm.segundos)*, que mantiene parada la T durante esos segundos (no necesitas ver el contenido, es un proc. que está en el usuario *ABDMIUTIL*)
- Comprueba si debe terminar el bucle usando *sec_T1*: si ve que no, vuelve al principio del bucle y repite el ciclo. **NO uses CURRVAL, no funciona. Usa solo NEXTVAL.**
- La forma de indicarle que termine el bucle es usando la secuencia *sec_T1*. Así, cuando modifiquemos *sec_T1* desde otra sesión del sqldeveloper (es una T nueva), lo que sucede es que la T de *trabajando_T1* terminará el bucle y el procedimiento.

  Para probar esto: abre tú otro SQLDeveloper nuevo y conectaté con tu usuario: cambia tú el valor de *sec_T1* y comprueba que el proc *trabajando_T1* termina.
- Para facilitar el seguimiento: lo último que hace el procedimiento es mostrar el mensaje "he terminado de trabajar" junto con el número de la transacción donde estaba (ver *que-num-trans.sql*).

**b).- Probar el procedimento *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*
    - Además, imprime esa filas.
  - Parar la T: llamando a *trabajando_T1 (5)*. (se dormirá hasta que se le ordene que continúe)
  - Continuar la T con otros INSERTs de otras tres COMPRAS diferentes.
    - Además, imprime esa filas.
  - Después, parar la T de nuevo, poniendo una 2ª llamada a *trabajando_T1 (5)*
    (se vuelve a dormir hasta que se le ordene que continúe).

- Ejecutar *probarMiT1*: ¿Se para? (debería)

- Desde otra transacción (abre otro sqlDeveloper) queremos *pedir que continúe*: hacer una modificación de la secuencia sec_T1 para provocar que *probarMiT1* continúe hasta la 2ª llamada. ¿Se para? (debería)

- Si volvemos a hacer otra modificación a la secuencia: ¿ el *trabajando_T1* continúa y termina?.

- Comprobar viendo el contenido de COMPRAS que ha insertado las filas esperadas y coincide con lo impreso.

**c).- Probar** con dos Ts: ***trabajando_T1 y trabajando_T2***: → *Así probaremos los otros Apartados de la Prác.*

- Ahora simular dos Ts concurrentes:
  - Necesito hacer un nuevo procedimiento *probarMiT2*, el mismo contenido que el *probarMiT1* salvo:
    - que llame a un nuevo procedimiento *trabajando_T2 (5)* , que hay que crear igual que el *trabajando_T1*
      - DEBE tener una nueva secuencia *sec_T2*.
    - Cuyas compras a insertar deben tener distinta PK.

- Este procedimiento *probarMiT2* se ejecutará en una copia nueva del sqlDeveloper. (será la T2)
En otra copia del sqlDeveloper ejecutamos *probarMiT1* . (será la T1)

- Las ordenes de continuar se las damos desde una tercera copia del sqlDeveloper (será una T3) haciendo tu estas operaciones desde una Hoja de Trabajo, una por una:
  - (modificando a mano la secuencia de cada T) Alterna las "ordenes de continuar" de la T1 y de la T2 hasta que terminen.

- Consulta qué filas de la tabla ve la T1 y la T2 antes de confirmar.
- Haz un *commit* a mano en la T1.
  Comprueba en qué número de transacción está la sesión de T1: → 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 número de transacción está la sesión de T2
- Comprueba ahora si en la tabla están las filas esperadas.

*Ahora estás preparado para ver la diferencia con otro nivel de aislamiento (en el siguiente apartado)*
(por defecto, se ha ejecutado todo en Nivel de Aislamiento *Read Committed*)

**d).- Repetir el mismo experimento como el c), poniendo el nivel de Aislamiento Secuenciable en los dos procedimentos** *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 y entender lo que pasa:
  - (ya visto en clase de Teoría) Situaciones de multiprogramación y bloqueos implícitos: *Transacciones-Posibles-Situaciones.pdf*
    - Comprueba los valores de las tablas en cada paso: Porqué tienen esos valores?
  - *probarNivelesAislamientoTrans.pdf*
  - Varias pruebas con distintos niveles de aislamiento *probarRollback.docx* (escribe el resultado en cada línea según la ejecutas, para entender porqué): te ayuda:
  - Comprobar en qué número de transacción estoy y cómo se usa el proc dormir ( ): *que-num-trans.sql*

---

**NOTA:**

**→ La práctica debe estar ejecutada en el servidor de Oracle de la Facultad (tiene algunos elementos que necesitas.**

**→ Aunque no lo necesitas, si has instalado el servidor Oracle en tu portátil (no es el SQLdeveloper), y quieres ejecutar esta práctica con ese servidor (además del de la Facultad) necesitas crear el proc. *dormir*: ver archivo *crear_sinonimos.pdf en vez de usar el del usuario ABDMIUTIL en que hay creado el dormir y un sinónimo.***