Universidad Complutense de Madrid
Facultad de Informática
Máster en Ingeniería Informática - Administración de Bases de Datos

Práctica 5

20 de octubre de 2020

Daniel Bastarrica Lacalle
Jose Javier Cortés Tejada

Índice

1. Uso de transacciones para concurrencia.

1.1. Ejercicio 1

Contesta a las siguientes preguntas:

1. **¿Qué está ocurriendo?**: las sentencias *SELECT* del procedimiento están siendo bloqueadas por transacciones concurrentes que modifican la información de las tablas. Concretamente el *SELECT* que realiza la suma ve una información diferente a la que ve el *SELECT* que consulta el saldo.

2. **¿Cómo se podría corregir el problema de forma sencilla?**: la forma más simple de eliminar estos conflictos es mover las comprobaciones antes del *COMMIT*. El problema de esta solución es que retrasa la realización del *COMMIT*, y por tanto alarga el tiempo que permanecen bloqueadas las filas de las tablas que intervengan en la transacción.

3. **¿Se podría permitir un nivel de concurrencia más alto en el apartado (2)?**: para evitar las inconsistencias en las transacciones es necesario marcar el procedimiento que comprueba los saldos con la sentencia *SET TRANSACTION READ ONLY*. Con esta solución se crea una transacción al comenzar el procedimiento de forma que las modificaciones de los datos no se ven reflejadas durante dicho procedimiento.

2. Bloqueos y contención por bloqueo

2.1. Ejercicio 3

Utiliza dos sesiones simultáneas de SQL*Plus o SQLDeveloper. En la primera sesión ejecuta una consulta de modificación de datos. Por ejemplo:

```
update mitabla set c2 = 1 where c1 = 10;
```

Antes de confirmarla con *commit*, ejecuta en la segunda sesión una sentencia DDL para eliminar la columna C2 de la tabla:

```
alter table drop column C2;
```

Anota en la memoria del ejercicio lo que ocurre en la segunda sesión:

```
Error que empieza en la línea: 1 del comando :
alter table mitabla drop column C2
Informe de error -
ORA-00054: recurso ocupado y obtenido con NOWAIT especificado o timeout vencido
00054. 00000 -  "resource busy and acquire with NOWAIT specified or timeout expired"
*Cause:    Interested resource is busy.
*Action:   Retry if necessary or increase timeout.
```

2.2. Ejercicio 4

¿Qué ocurre si en lugar de una sentencia DDL ejecutas una sentencia DML?: utiliza en la segunda sesión una sentencia *UPDATE* para modificar otra fila diferente a la modificada en la sentencia anterior y vuelve a repetir el ejercicio. Ejecuta en la segunda sesión:

```
update mitabla set c2 = 1 where c1 = 11;
```

Anota en la memoria del ejercicio lo que ocurre en este caso.

En este caso no hay problemas de bloqueos dado que la sentencias bloquean diferentes filas de la tabla, al contrario que en el caso anterior, donde la segunda sentencia intentaba bloquear la tabla entera mientras la primera bloqueaba una fila.

2.3. Ejercicio 5

Prepara dos consultas SELECT ... FOR UPDATE en sesiones distintas de tal forma que la primera bloquee a la segunda. Consulta en *Enterprise Manager* el bloqueo que se ha producido: entra en la pestaña *Performance* y ahí en el enlace *Instance Locks*. En el bloqueo que se muestra en la pantalla, utiliza el botón correspondiente para cancelar la sesión que está bloqueando y anota en la memoria del ejercicio el mensaje que aparece en la sesión cancelada.

```
Se ha restablecido la conexión a la base de datos.
Se ha perdido cualquier estado de sesión o transacción pendiente.
```

3. Interbloqueos (deadlocks)

3.1. Ejercicio 6

Considera la tabla y las filas que has creado para los ejercicios del apartado 2.1 y abre dos sesiones de SQL*Plus o SQL Developer. Indica la secuencia de sentencias SQL que operen sobre esa tabla y lleven a un interbloqueo entre ambas sesiones. Muestra el comportamiento de las sesiones ante el interbloqueo.

Se ejecutan las siguientes instrucciones para crear y llenar las tablas:

```
1 create table mitabla (c1 integer, c2 integer);
2 insert into mitabla values (10,0);
3 insert into mitabla values (11,0);
4 commit;
```

A continuación se ejecutan las siguientes instrucciones, en el orden indicado, para provocar un *deadlock*:

```
-- sesión 1
update mitabla set c2 = 1 where c1 = 10;

-- sesión 2
update mitabla set c2 = 1 where c1 = 11;

-- sesión 1
update mitabla set c2 = 1 where c1 = 11;

-- sesión 2
update mitabla set c2 = 1 where c1 = 10;
```

Tras haber ejecutado la última consulta en la sesión 2 hemos obtenido el siguiente mensaje por consola de la sesión 1. En dicha salida se observa que Oracle ha abortado la consulta que se estaba ejecutando, conviertiendo el *deadlock* en un bloqueo normal. En este caso se aborta una de las operaciones mientras que la otra se queda esperando al *COMMIT* (o ROLLBACK) de la sesión cuya consulta ha sido abortada.

```
-- sesión 1

Error que empieza en la línea: 1 del comando :
update mitabla set c2 = 1 where c1 = 11
Informe de error -
ORA-00060: detectado interbloqueo mientras se esperaba un recurso
```

3.2. Ejercicio 7

En el caso de estudio visto en el apartado 1 (*08Ej-transacciones-backup1.sql*) el administrador de la BD detecta que en muy raras ocasiones se producen situaciones de interbloqueo. ¿Podrías determinar en qué caso se puede producir un interbloqueo entre dos traspasos ejecutando ese código? ¿Cómo se podría resolver?

Se podría producir un *deadlock* si se realizasen dos transacciones de forma concurrente donde se hace un traspaso de dinero entre dos cuentas de forma bidireccional, es decir, se hace una transferencia de A a B y otra de B a A.

Se podría resolver añadiendo al principio del procedimiento una sentencia ***SELECT ... FOR UPDATE*** donde la condición ***WHERE*** incluyese a las dos cuentas como se muestra a continuación. Con esto conseguimos una operación atómica que bloquea las dos filas de forma simultánea.

```
select *
from cuentas
where codCuenta = p_ctaOrigen or codCuenta = p_ctaDestino for update;
```

3.3. Ejercicio 8

Utiliza una tabla como la que se ha generado en el apartado 2 para provocar un interbloqueo entre tres transacciones. Para ello, abre tres sesiones con SQL*Plus y ejecuta sentencias *SELECT ... FOR UPDATE* para generar el interbloqueo. Una vez realizado el interbloqueo, Oracle automáticamente cancela una de las transacciones. ¿En qué estado quedan las otras dos transacciones? Explica el motivo por el que quedan en ese estado.

Hemos ejecutado las siguientes instrucciones en los terminales según se indica. Dichas instrucciones provocan un interbloqueo triple que Oracle soluciona abortando una de las consultas, en este caso la de la sesión 2. Las otras dos sesiones se han quedado bloqueadas según se explica a continuación:

- la sesión 1 se ha quedado esperando a que termine la transacción de la sesión 2.
- la sesión 3 se ha quedado esperando a que termine la transacción de la sesión 1, la cual estaba siendo bloqueada a su vez por la sesión 2.

El bloqueo se resuelve terminando la transacción de la sesión 2. Con esto la sesión 1 queda liberada para que acabe su transacción. Una vez se termine esta transacción la sesión 3 quedará liberada.

```
-- sesión 1
select * from mitabla where c1 = 10 for update;

-- sesion 2
select * from mitabla where c1 = 11 for update;

-- sesion 3
select * from mitabla where c1 = 12 for update;

-- sesión 1
select * from mitabla where c1 = 11 for update;

-- sesion 2
select * from mitabla where c1 = 12 for update;

-- sesion 3
select * from mitabla where c1 = 10 for update;
```

3.4. Ejercicio 9

Consulta el contenido del fichero alert_orcl.log que se encuentra en ese directorio y abre el fichero de traza que se indica en el mensaje de interbloqueo (está hacia el final del fichero). Ese fichero de traza contiene gran cantidad de información, incluyendo el volcado de la memoria del proceso cancelado. Esta información puede ser de gran utilidad para determinar las causas y plantear la solución de problemas de interbloqueos complejos en aplicaciones. Recorre este fichero para ver el tipo de información que se proporciona. En particular, puedes identificar la siguiente información:

1. **los procesos que intervienen en el interbloqueo (*deadlock graph*).**
2. **las sesiones y consultas SQL que se estaban ejecutando en cada una de ellas cuando se produjo el interbloqueo.**

Observa que en el grafo de interbloqueo deben aparecer las tres sesiones implicadas en el interbloqueo. Incluye en la memoria de este ejercicio esta información del fichero de traza.

```
*** 2019-11-04 17:47:20.656
DEADLOCK DETECTED ( ORA-00060 )

[Transaction Deadlock]

The following deadlock is not an ORACLE error. It is a
deadlock due to user error in the design of an application
or from issuing incorrect ad-hoc SQL. The following
information may aid in determining the deadlock:

Deadlock graph:
                       ---------Blocker(s)--------  ---------Waiter(s)---------
Resource Name          process session holds waits  process session holds waits
TX-0007001c-0000023d        25      23     X             32      28           X
TX-00040001-0000022e        32      28     X             40      25           X
TX-00090019-000002be        40      25     X             25      23           X

session 23: DID 0001-0019-0000003D        session 28: DID 0001-0020-0000000C
session 28: DID 0001-0020-0000000C        session 25: DID 0001-0028-00000036
session 25: DID 0001-0028-00000036        session 23: DID 0001-0019-0000003D

Rows waited on:
  Session 23: obj - rowid = 00011EF9 - AAAR75AAEAAAAGLAAC
  (dictionary objn - 73465, file - 4, block - 395, slot - 2)
  Session 28: obj - rowid = 00011EF9 - AAAR75AAEAAAAGLAAB
  (dictionary objn - 73465, file - 4, block - 395, slot - 1)
  Session 25: obj - rowid = 00011EF9 - AAAR75AAEAAAAGLAAA
  (dictionary objn - 73465, file - 4, block - 395, slot - 0)
```

```
29
30  ----- Information for the OTHER waiting sessions -----
31  Session 28:
32    sid: 28 ser: 42 audsid: 81035 user: 5/SYSTEM flags: 0x45
33    pid: 32 O/S info: user: oracle, term: UNKNOWN, ospid: 4053
34      image: oracle@ubuntu32vb
35    client details:
36      O/S info: user: usuario_local, term: unknown, ospid: 9652
37      machine: PTO0203 program: SQL Developer
38      application name: SQL Developer, hash value=1012150930
39    current SQL:
40    select * from mitabla where c1 = 11 for update
41
42  Session 25:
43    sid: 25 ser: 160 audsid: 4294967295 user: 5/SYSTEM flags: 0x41
44    pid: 40 O/S info: user: oracle, term: UNKNOWN, ospid: 5659
45      image: oracle@ubuntu32vb
46    client details:
47      O/S info: user: usuario_local, term: unknown, ospid: 9652
48      machine: PTO0203 program: SQL Developer
49      application name: SQL Developer, hash value=1012150930
50    current SQL:
51    select * from mitabla where c1 = 10 for update
52
53  ----- End of information for the OTHER waiting sessions -----
54
55  Information for THIS session:
56
57  ----- Current SQL Statement for this session (sql_id=gywu0r217h1bn) -----
58  select * from mitabla where c1 = 12 for update
59  ================================================
```