

# Ejercicios de transacciones y backup (1)

28 de octubre de 2019

## 1. Uso de transacciones para concurrencia.

En una entidad bancaria se utiliza Oracle para el sistema de gestión de cuentas bancarias. Se ha informado al departamento de informática de un problema que ocurre de forma intermitente: las operaciones de traspasos de fondos entre cuentas bancarias generan algunas veces un mensaje de error indicando que el saldo de la cuenta no coincide con la suma de los importes de los movimientos realizados. Sin embargo, cuando se realizan las comprobaciones de balance al final de cada día (cuando no se realizan traspasos), no se detecta ningún error. Las tablas y el procedimiento almacenado que realiza los traspasos está contenido en el fichero `08Ej-transacciones-backup1.sql`. (Este código es una simplificación de una operación de traspaso real; no se incluyen comprobaciones que se deberían realizar –por ejemplo, que haya saldo suficiente en la cuenta de origen).

Nos han informado de que este error se produce especialmente en aquellas cuentas que tienen gran número de traspasos cada día resultado de distintas operaciones. Los programadores han estudiado el código y no han visto ningún problema en la transacción incluida en el procedimiento, y por otra parte en las pruebas unitarias no se ha conseguido reproducir ningún error, por lo que se sospecha de que pueden estar produciéndose interferencias entre operaciones de traspaso que se ejecutan concurrentemente.

**NOTA:** Para este ejercicio no ejecutes el procedimiento PL/SQL directamente para intentar reproducir el error: solo sucede cuando se ejecutan concurrentemente varias operaciones de traspaso, y es realmente difícil reproducirlo con ejecuciones manuales. Es mejor razonar en el código fuente cómo se pueden intercalar las sentencias para que se produzca un resultado incorrecto y, si es necesario, ejecutar las sentencias una por una utilizando dos sesiones distintas de SQL\*Plus o SQLDeveloper para ver qué es lo que está ocurriendo en un ejemplo sencillo.

**Ejercicio 1.** *Contesta a las siguientes preguntas:*

1. *¿Qué está ocurriendo?*
2. *¿Cómo se podría corregir el problema de forma sencilla?*
3. *¿Se podría permitir un nivel de concurrencia más alto en el apartado (b)?*

## 2. Bloqueos y contención por bloqueo.

### 2.1. Tipos de bloqueo.

En Oracle hay dos tipos de bloqueo: bloqueo compartido y bloqueo exclusivo. Los bloqueos exclusivos se utilizan para modificar los datos de una fila con sentencias DML (se bloquea la fila), o para modificar la estructura de una tabla con sentencias DDL (se bloquea toda la tabla). Cuando una sesión tiene el bloqueo exclusivo sobre una fila (o tabla), ninguna otra sesión puede tomar el bloqueo exclusivo sobre la misma fila (o cualquier fila de la tabla en el segundo caso). Vamos a ver una situación en la que una sentencia SELECT puede bloquear determinadas operaciones.

**Ejercicio 2.** *Crea una tabla con dos columnas, C1 y C2, de tipo integer e inserta en ella 2 filas:*

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

Vamos a bloquear la tabla con un bloqueo compartido producido por una consulta SQL de modificación de datos y vamos a comprobar que efectivamente está bloqueada, intentando modificar la estructura de la tabla mientras se ejecuta la consulta SQL:

**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 mitabla drop column C2;
```

*Anota en la memoria del ejercicio lo que ocurre en la segunda sesión.*

¿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.

**Ejercicio 4.** *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.*

### 2.2. Contención por bloqueo.

**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.*

### 3. Interbloqueos (*deadlocks*).

#### 3.1. Interbloqueo sencillo entre dos transacciones.

**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.*

**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 trasposos ejecutando ese código? ¿Cómo se podría resolver?*

#### 3.2. Interbloqueo entre tres transacciones.

En clase hemos visto una forma sencilla de provocar un interbloqueo (deadlock) entre dos transacciones. Los interbloqueos pueden ser mucho más complejos en los que intervienen múltiples transacciones. Para resolver problemas complejos de interbloqueo, es fundamental obtener información sobre las sesiones que intervienen en el bloqueo y las consultas asociadas. Vamos a verlo en el siguiente ejercicio.

**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.*

#### 3.3. Consulta del fichero `alert_orcl.log`.

Para estudiar el interbloqueo vamos a consultar el log de alertas. Para ello, consulta la tabla `v$diag_info` para saber el directorio donde se guardan las trazas (busca el directorio de 'Diag Trace').

**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) y (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.*

### **3.4. Consulta desde Enterprise Manager.**

El log de alertas también lo puedes consultar desde Enterprise Manager: en la pestaña de inicio (Home) selecciona el enlace Alert log contents. Carga las últimas 50 entradas del log para mostrar la información del interbloqueo (el interbloqueo debería aparecer la primera).