

INFORME PRACTICA 4

ASIGNATURA: BASES DE DATOS

AUTORES: RAYNER TAN LUC
LUIS JARAMILLO

Hay que tener en cuenta que al ejecutarse las instrucciones DDL, el gestor realiza de forma implícita un commit antes y después de cada sentencia DDL

```
CREATE TABLE cuentas (  
  numero number primary key,  
  saldo number not null  
);  
  
INSERT INTO cuentas VALUES (123, 400);  
INSERT INTO cuentas VALUES (456, 300);  
COMMIT;
```

```
1 CREATE TABLE cuentas {  
2   numero number primary key,  
3   saldo number not null  
4 };  
5  
6 INSERT INTO cuentas VALUES (123, 400);  
7 INSERT INTO cuentas VALUES (456, 300);  
8 COMMIT;  
9
```

BLOQUEOS

3. Desde la T1 ver el saldo de la cuenta 123. ¿Qué se ve?

Se ve el saldo inicial de la cuenta '123' en la transacción "T1"

```
select saldo from cuentas where numero = '123';
```

SALDO	
1	400

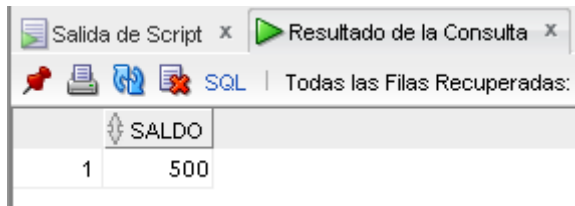
4. Desde T1 aumenta 100 euros el saldo de la cuenta 123.

Se actualiza en T1 añadiéndole 100 al saldo de la cuenta '123'

```
12 update cuentas  
13 set saldo =saldo +100  
14 where numero = '123';
```

5. Desde la T1 ver el saldo de la cuenta 123. ¿Qué se ve?

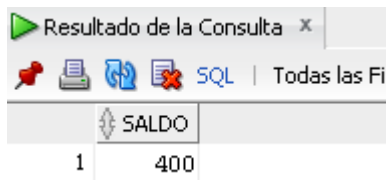
Se ve la actualización del saldo en la transacción T1



ID	SALDO
1	500

6. Desde la T2 ver el saldo de la cuenta 123. ¿Qué se ve?

En T2 no se ve la actualización de la otra transacción T1 porque no se ha confirmado



ID	SALDO
1	400

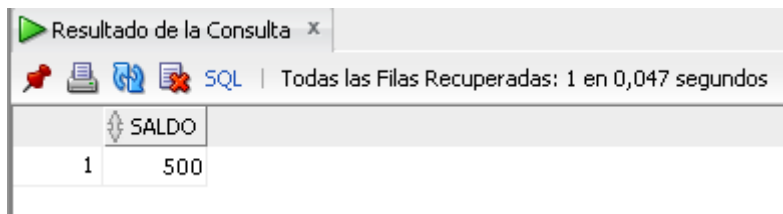
7. Desde la T1: COMMIT;

Una vez la transacción T1 es confirmado, los cambios provocados se hacen permanentes

```
8 | COMMIT ;
```

8. Desde la T2 ver el saldo de la cuenta 123. ¿Qué se ve?

Se observa que en T2 se ha actualizado el valor del saldo de la cuenta '123' porque en T1 se ha ejecutado explícitamente la sentencia COMMIT



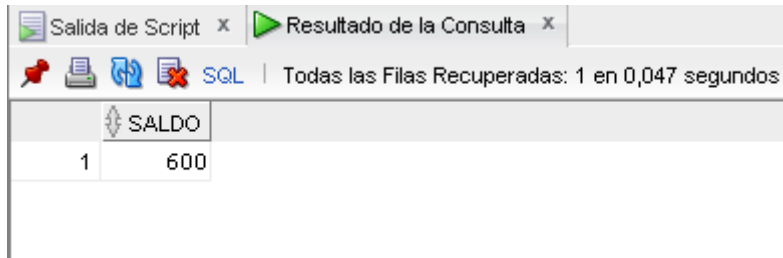
ID	SALDO
1	500

¿Qué ha pasado?

T2 no ve los cambios de T1 hasta que se confirme mediante la sentencia "Commit ", la transacción se confirma, y los cambios se hacen permanentes y visibles al resto de los usuarios.

2. Bloqueos (update vs update)

1. Desde T1 aumenta 100 euros el saldo de la cuenta 123



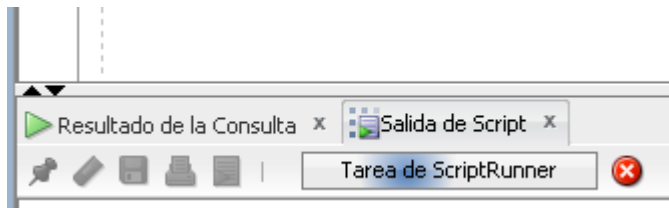
Salida de Script x Resultado de la Consulta x

Todas las Filas Recuperadas: 1 en 0,047 segundos

	SALDO
1	600

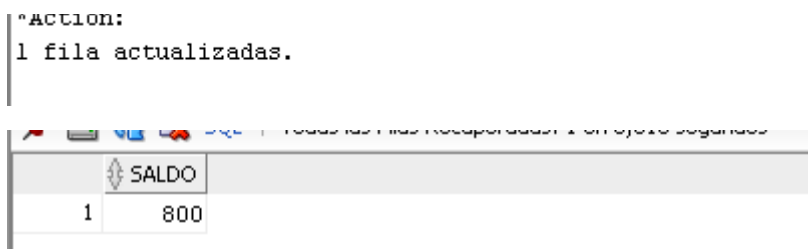
2. Desde la T2 aumenta 200 euros el saldo de la cuenta 123. ¿Se puede?. ¿qué le pasa a la T2?

No se puede, porque T1 bloquea ese elemento de dato, es decir cuando una transacción quiere acceder a un elemento de dato primero bloque el elemento que quiere acceder luego solicita al gestor de control de bloqueo que decide si es adecuado, T2 espera a T1 para asegurar la secuencialidad basado en ejecutar transacciones concurrentemente



3. T1: COMMIT; ¿qué le pasa a la T2?

Una vez confirmada T1 se produce la actualización en T2



^ACTION:
1 fila actualizadas.

	SALDO
1	800

3. Bloqueos (Deadlock)

1. En T1 aumenta 100 euros el saldo de la cuenta 123

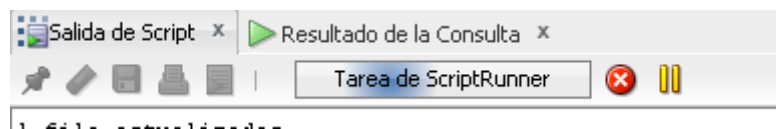
```
update cuentas
set saldo =saldo+100
where numero='123';
```

2. Desde T2 aumenta 200 euros el saldo de la cuenta 456

```
1 update cuentas
2 set saldo =saldo + 200
3 where numero='456';
4
```

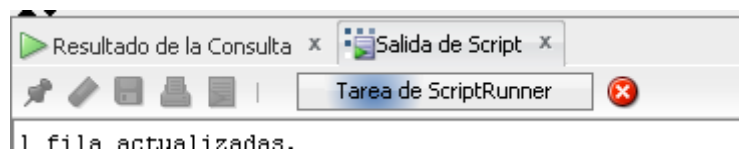
3. Desde T1 aumenta 300 euros el saldo de la cuenta '456'

```
7 update cuentas
8 set saldo =saldo+300
9 where numero='456';
10
11
```



4. Desde T2 aumenta 400 euros el saldo de la cuenta 123

```
1 update cuentas
2 set saldo =saldo + 400
3 where numero='123';
4
```



¿Qué ha pasado?

Se ha producido un DeadLock (Bloqueo mutuo o interbloqueo), que es un bloqueo permanente de 2 procesos que intentan utilizar los recursos del otro proceso produciéndose una espera de liberación del recurso, no existe una solución general.

En este caso T1 toma el lock del Recurso 1 (R1), T2 toma el lock del recurso 2 (R2), luego T2 intenta tomar el bloqueo de R1 y queda a la espera de que finalice T1, luego T1 a su vez intenta utilizar R2, se produce un interbloqueo, donde las dos transacciones esperan la liberación del recurso que utiliza la otra transacción

4. Nivel de aislamiento

1 En T1: ALTER SESSION SET ISOLATION_LEVEL SERIALIZABLE;

```
SQL> ALTER SESSION SET ISOLATION_LEVEL SERIALIZABLE
```

2 En T1 : SELECT SUM(saldo) FROM cuentas

```
SQL> SELECT SUM(SALDO) FROM CUENTAS
```

3 En T2 : UPDATE cuentas SET saldo=saldo +100; COMMIT;

```
SQL> UPDATE CUENTAS SET SALDO =SALDO+100; COMMIT
```

4 En T1 : SELECT SUM(saldo) FROM cuentas

```
SQL> SELECT SUM(SALDO) FROM CUENTAS
```

¿Qué ha pasado?

Los bloqueos de lectura depende del nivel de aislamiento. El tipo de aislamiento "Serializable" protege la lectura de cualquier modificación al final de la transacción evitando que aparezcan las lecturas fantasmas

En este caso T1 realiza una segunda lectura de datos que han sido modificados por otra transacción T2 ya confirmada

5 En T1: ALTER SESSION SET ISOLATION_LEVEL READ COMMITTED

```
SQL> ALTER SESSION SET ISOLATION_LEVEL READ COMMITTED
```

6 En T1 : SELECT SUM(saldo) FROM cuentas

```
SQL> SELECT SUM(SALDO) FROM CUENTAS
```

7 En T2 : UPDATE cuentas SET saldo=saldo +100; COMMIT;

```
SQL> UPDATE CUENTAS SET SALDO=SALDO+100; COMMIT
```

8 En T1 : SELECT SUM(saldo) FROM cuentas

```
SQL> SELECT SUM(SALDO) FROM CUENTAS
```

¿Qué ha pasado?

Los niveles de aislamiento READ COMMITTED, los bloqueos de lectura se liberan en cuanto se termina una lectura.

Ocurre que la segunda lectura es distinta de la primera lectura en T1, y eso es debido que al no mantener los bloqueos de lectura de las filas leídas pueden ser modificadas, en este caso el saldo ha sido modificado de la tabla Cuentas en T2, ocurriendo que la segunda lectura en T1 saldo haya cambiado de valor siendo distinta a la primera lectura en T1