Prácticas de Administración de Bases de Datos

Grado en Ingeniería Informática

PRÁCTICA 6

Gestión y control de la concurrencia
(Transacciones)

Ejercicios

1. Inicia una sesión y ejecuta las siguientes sentencias. Observa el resultado.

```
CREATE TABLE T (id INT PRIMARY KEY, s VARCHAR(30), si NUMBER(5));
INSERT INTO T (id, s) VALUES (1, 'first');
COMMIT;
INSERT INTO T (id, s) VALUES (2, 'second');
SELECT * FROM T;
ROLLBACK;
SELECT * FROM T;
```

SQL> CREATE TABLE T (id INT PRIMARY KEY, s VARCHAR(30), si NUMBER(5));

Tabla creada.

SQL> INSERT INTO T (id, s) VALUES (1, 'first');

1 fila creada.

SQL> COMMIT;

Confirmación terminada.

SQL> INSERT INTO T (id, s) VALUES (2, 'second');

1 fila creada.

SQL> SELECT * FROM T;

```
        ID S                              SI
---------- ------------------------------ ----------
         1 first
         2 second
```

SQL> ROLLBACK;

Rollback terminado.

SQL> SELECT * FROM T;

```
        ID S                              SI
---------- ------------------------------ ----------
```

1 first

Añade las siguientes sentencias. ¿Cuál ha sido el efecto de ROLLBACK? ¿Ocurrió algún COMMIT implícito?

```
INSERT INTO T (id, s) VALUES (2, 'Will this be committed?');
CREATE TABLE T2 (id INT);
INSERT INTO T2 (id) VALUES (1);
SELECT * FROM T2;
ROLLBACK;
```

SQL> INSERT INTO T (id, s) VALUES (2, 'Will this be committed?');

1 fila creada.

SQL> CREATE TABLE T2 (id INT);

Tabla creada.

SQL> INSERT INTO T2 (id) VALUES (1);

1 fila creada.

SQL> SELECT * FROM T2;

```
        ID
----------
         1
```

SQL> ROLLBACK;

Rollback terminado.

SQL> select * from t2;

ninguna fila seleccionada

SQL> select * from t;

```
        ID S                            SI
---------- ------------------------------ ----------
         1 first
         2 Will this be committed?
```

La creación de la tabla t2, provoca la confirmación implícita del insert del valor 2 en t

Añade las siguientes sentencias. ¿Qué puede observarse?

```
DELETE FROM T WHERE id > 1 ;
COMMIT;

INSERT INTO T (id, s) VALUES (2, 'The test starts by this');
SELECT (1/0) FROM DUAL;
UPDATE T SET s = 'foo' WHERE id = 9999;
DELETE FROM T WHERE id = 7777;
COMMIT;

SELECT * FROM T;
```

SQL> DELETE FROM T WHERE id > 1 ;

1 fila suprimida.

SQL> COMMIT;

Confirmación terminada.

SQL> INSERT INTO T (id, s) VALUES (2, 'The test starts by this');

1 fila creada.

SQL> SELECT (1/0) FROM DUAL;
SELECT (1/0) FROM DUAL
       *
ERROR en línea 1:
ORA-01476: el divisor es igual a cero

SQL> UPDATE T SET s = 'foo' WHERE id = 9999;

0 filas actualizadas.

SQL> DELETE FROM T WHERE id = 7777;

0 filas suprimidas.

SQL> COMMIT;

Confirmación terminada.

SQL> SELECT * FROM T;

```
        ID S                             SI
---------- ------------------------------ ----------
         1 first
         2 The test starts by this
```

Ahora añade las siguientes sentencias. ¿Qué se observa?

```
UPDATE T SET s = 'Otro test' WHERE id = 2;
INSERT INTO T (id, s)
   VALUES (2, 'Soy un duplicado');
INSERT INTO T (id, s)
   VALUES (3, 'Intentando insertar una cadena muy larga');
INSERT INTO T (id, s, si)
   VALUES (4, 'Se puede un 123456?', 123456);
COMMIT;
SELECT * FROM T;
```

```
SQL> UPDATE T SET s = 'Otro test' WHERE id = 2;

1 fila actualizada.

SQL> INSERT INTO T (id, s) VALUES (2, 'Soy un duplicado');
INSERT INTO T (id, s)
*
ERROR en línea 1:
ORA-00001: restricción ·nica (SYSTEM.SYS_C0011429) violada

SQL> INSERT INTO T (id, s) VALUES (3, 'Intentando insertar una cadena muy larga');
  VALUES (3, 'Intentando insertar una cadena muy larga')
         *
ERROR en línea 2:
ORA-12899: el valor es demasiado grande para la columna "SYSTEM"."T"."S" (real:
40, máximo: 30)

SQL> INSERT INTO T (id, s, si) VALUES (4, 'Se puede un 123456?', 123456);
  VALUES (4, 'Se puede un 123456?', 123456)
                         *
ERROR en línea 2:
ORA-01438: valor mayor que el que permite la precisi¾n especificada para esta
columna

SQL> COMMIT;

Confirmación terminada.

SQL> SELECT * FROM T;
```

```
        ID S                            SI
---------- ------------------------------ ----------
         1 first
         2 Otro test
```

2. Ejecuta las siguientes sentencias. Entendiendo que se trata de transferencias de una cuenta a otra.. ¿hay algún problema en lo que se expresa a continuación? ¿Cómo podría solucionarse?

```
DROP TABLE Accounts;
CREATE TABLE Accounts (
  acctID  INTEGER NOT NULL PRIMARY KEY,
  balance INTEGER NOT NULL
          CONSTRAINT unloanable_account  CHECK (balance >= 0)
);

INSERT INTO Accounts (acctID, balance) VALUES (101, 1000);
INSERT INTO Accounts (acctID, balance) VALUES (202, 2000);
COMMIT;
SELECT * FROM Accounts;

UPDATE Accounts SET balance = balance - 100 WHERE acctID = 101;
UPDATE Accounts SET balance = balance + 100 WHERE acctID = 202;
COMMIT;
SELECT * FROM Accounts;

UPDATE Accounts SET balance = balance - 2000 WHERE acctID = 101;
UPDATE Accounts SET balance = balance + 2000 WHERE acctID = 202;
COMMIT;
SELECT * FROM Accounts;
```

```
SQL> DROP TABLE Accounts;
DROP TABLE Accounts
           *
ERROR en lÝnea 1:
ORA-00942: la tabla o vista no existe

SQL> CREATE TABLE Accounts (
     acctID  INTEGER NOT NULL PRIMARY KEY,
     balance INTEGER NOT NULL
             CONSTRAINT unloanable_account  CHECK (balance >= 0));

Tabla creada.

SQL> INSERT INTO Accounts (acctID, balance) VALUES (101, 1000);
1 fila creada.

SQL> INSERT INTO Accounts (acctID, balance) VALUES (202, 2000);
1 fila creada.
```

```
SQL> COMMIT;
Confirmación terminada.

SQL> SELECT * FROM Accounts;
   ACCTID    BALANCE
---------- ----------
       101       1000
       202       2000

SQL> UPDATE Accounts SET balance = balance - 100 WHERE acctID = 101;
1 fila actualizada.

SQL> UPDATE Accounts SET balance = balance + 100 WHERE acctID = 202;
1 fila actualizada.

SQL> COMMIT;
Confirmación terminada.

SQL> SELECT * FROM Accounts;
   ACCTID    BALANCE
---------- ----------
       101        900
       202       2100

SQL> UPDATE Accounts SET balance = balance - 2000 WHERE acctID = 101;
UPDATE Accounts SET balance = balance - 2000 WHERE acctID = 101
*
ERROR en lÝnea 1:
ORA-02290: restricci¾n de control (SYSTEM.UNLOANABLE_ACCOUNT) violada

SQL> UPDATE Accounts SET balance = balance + 2000 WHERE acctID = 202;
1 fila actualizada.

SQL> COMMIT;
Confirmación terminada.

SQL> SELECT * FROM Accounts;
   ACCTID    BALANCE
---------- ----------
       101        900
       202       4100
```

Se ha realizado "ingreso" en la cuenta 202 pero no la "resta" en 101. Esto podría corregirse del siguiente modo

```
BEGIN
  UPDATE Accounts SET balance = balance - 2000 WHERE acctID = 101;
```

```
  UPDATE Accounts SET balance = balance + 2000 WHERE acctID = 202;
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN ROLLBACK;
END;
```

3. Abre dos sesiones en dos ventanas SQL*Plus, correspondientes a los clientes A y B. Ejecuta las correspondientes sentencias para cada cliente. ¿Observas alguna anomalía? ¿Alguna sugerencia?

```
-- En cualquier cliente:
DROP TABLE Accounts;
CREATE TABLE Accounts (
  acctID  INTEGER NOT NULL PRIMARY KEY,
  balance INTEGER NOT NULL
          CONSTRAINT unloanable_account  CHECK (balance >= 0)
);

INSERT INTO Accounts (acctID, balance) VALUES (101, 1000);
INSERT INTO Accounts (acctID, balance) VALUES (202, 2000);

COMMIT;
SELECT * FROM Accounts;

-- Cliente A comienza:
SELECT balance FROM Accounts WHERE acctID = 101;
-- se simula la lectura de 1000 en una variable

-- Cliente B comienza:
SELECT balance FROM Accounts WHERE acctID = 101;
-- se simula la lectura de 1000 en una variable

-- Cliente A continúa:
UPDATE Accounts SET balance = (1000 - 200) WHERE acctID = 101;

-- Cliente B continúa:
UPDATE Accounts SET balance = (1000 - 500) WHERE acctID = 101;

-- Cliente A continúa:
SELECT acctID, balance FROM Accounts WHERE acctID = 101;
COMMIT;

-- Cliente B continúa:
SELECT acctID, balance FROM Accounts WHERE acctID = 101;
COMMIT;
```

```
SQL> DROP TABLE Accounts;

SQL> CREATE TABLE Accounts (
     acctID  INTEGER NOT NULL PRIMARY KEY,
     balance INTEGER NOT NULL
             CONSTRAINT unloanable_account  CHECK (balance >= 0));
```

```
Tabla creada.

SQL> INSERT INTO Accounts (acctID, balance) VALUES (101, 1000);
1 fila creada.

SQL> INSERT INTO Accounts (acctID, balance) VALUES (202, 2000);
1 fila creada.

SQL> COMMIT;

Confirmación terminada.
SQL> SELECT * FROM Accounts;

   ACCTID    BALANCE
---------- ----------
      101       1000
      202       2000

SQL> SELECT balance FROM Accounts WHERE acctID = 101;

  BALANCE
----------
     1000

SQL> SELECT balance FROM Accounts WHERE acctID = 101;

  BALANCE
----------
     1000

SQL> -- Cliente A continúa:
SQL> UPDATE Accounts SET balance = (1000 - 200) WHERE acctID =101;


1 fila actualizada.

SQL>--Cliente B continúa:
SQL> UPDATE Accounts SET balance = (1000 - 500) WHERE acctID = 101;

*** Esto se queda bloqueado****

SQL> -- Cliente A continúa:
SQL> SELECT acctID, balance FROM Accounts WHERE acctID = 101;

   ACCTID    BALANCE
---------- ----------
      101        800
```

SQL> COMMIT;

Confirmación terminada.

***Ahora A desbloquea y la operación de B se puede hacer, pero B hemos simulado que tiene en memoria 1000 y restarle 500 deja el valor como 500 (y si ya A había quitado 200 no es muy correcto) ***

-- Cliente B continúa:
SQL> SELECT acctID, balance FROM Accounts WHERE acctID = 101;

```
   ACCTID    BALANCE
---------- ----------
       101        500
```

SQL> COMMIT;

Confirmación terminada.

***El cliente B utiliza un valor "antiguo" de A (1000, al haber guardado su "valor"), que podría solucionarse mediante la instrucción UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101; (de esta forma el valor del balance queda como 300) ***

Ahora ejecuta las siguientes sentencias. ¿Qué ocurre?

```
-- En cualquier cliente:
DELETE FROM Accounts;
INSERT INTO Accounts (acctID,balance) VALUES (101,1000);
INSERT INTO Accounts (acctID,balance) VALUES (202,2000);
COMMIT;

-- El cliente A transfiere 100 euros de la cuenta 101 a la 202
-- El cliente B transfiere 200 euros de la cuenta 202 a la 101

-- Cliente A:
UPDATE Accounts SET balance = balance - 100 WHERE acctID = 101;

-- Cliente B:
UPDATE Accounts SET balance = balance - 200 WHERE acctID = 202;

-- Cliente A:
UPDATE Accounts SET balance = balance + 100 WHERE acctID = 202;

-- Cliente B:
UPDATE Accounts SET balance = balance + 200 WHERE acctID = 101;
```

SQL> DELETE FROM Accounts;
2 filas suprimidas.

SQL> INSERT INTO Accounts (acctID,balance) VALUES (101,1000);

1 fila creada.

SQL> INSERT INTO Accounts (acctID,balance) VALUES (202,2000);
1 fila creada.

SQL> COMMIT;
Confirmación terminada.

SQL> -- Cliente A:
SQL> UPDATE Accounts SET balance = balance - 100 WHERE acctID = 101;
1 fila actualizada.

SQL> -- Cliente B:
SQL> UPDATE Accounts SET balance = balance - 200 WHERE acctID = 202;
1 fila actualizada.

SQL> -- Cliente A:
SQL> UPDATE Accounts SET balance = balance + 100 WHERE acctID = 202;
**** Se queda A Bloqueado esperando a B*****

SQL> -- Cliente B:
SQL>UPDATE Accounts SET balance = balance + 200 WHERE acctID = 101
**** Se queda B Bloqueado esperando a A*****

El sistema genera en A
UPDATE Accounts SET balance = balance + 100 WHERE acctID = 202
*
ERROR en línea 1:
ORA-00060: detectado interbloqueo mientras se esperaba un recurso

Ahora ejecuta las siguientes sentencias. ¿Observas alguna anomalía?

```
DELETE FROM Accounts;
INSERT INTO Accounts (acctID,balance) VALUES (101,1000);
INSERT INTO Accounts (acctID,balance) VALUES (202,2000);
COMMIT;

-- Cliente A:
SELECT * FROM Accounts WHERE balance > 500;

-- Cliente B:
UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101;
UPDATE Accounts SET balance = balance + 500 WHERE acctID = 202;
COMMIT;

-- Cliente A:
SELECT * FROM Accounts WHERE balance > 500;
COMMIT;
```

```
SQL> DELETE FROM Accounts;
2 filas suprimidas.

SQL> INSERT INTO Accounts (acctID,balance) VALUES (101,1000);
1 fila creada.

SQL> INSERT INTO Accounts (acctID,balance) VALUES (202,2000);
1 fila creada.

SQL> COMMIT;
Confirmación terminada.

SQL> -- Cliente A:
SQL> SELECT * FROM Accounts WHERE balance > 500;

   ACCTID    BALANCE
---------- ----------
      101       1000
      202       2000

SQL>-- Cliente B:
SQL>UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101;
1 fila actualizada.

SQL>UPDATE Accounts SET balance = balance + 500 WHERE acctID = 202;
1 fila actualizada.

SQL>COMMIT;
Confirmación terminada.

SQL> -- Cliente A:
SQL> SELECT * FROM Accounts WHERE balance > 500;

   ACCTID    BALANCE
---------- ----------
      101       2500
```

****Lectura no repetible.****

4. Ejecuta las siguientes sentencias y analiza su resultado.

```
DELETE FROM Accounts;
INSERT INTO Accounts (acctID,balance) VALUES (101,1000);
INSERT INTO Accounts (acctID,balance) VALUES (202,2000);
COMMIT;

SELECT * FROM Accounts;
```

```
SQL> DELETE FROM Accounts;
2 filas suprimidas.

SQL> INSERT INTO Accounts (acctID,balance) VALUES (101,1000);
1 fila creada.

SQL> INSERT INTO Accounts (acctID,balance) VALUES (202,2000);
1 fila creada.

SQL> COMMIT;

Confirmación terminada.
SQL> SELECT * FROM Accounts;

   ACCTID    BALANCE
---------- ----------
       101       1000
       202       2000
```

A continuación se ejecutan las siguientes instrucciones, Obtenga conclusiones..

```
SELECT NAME, STATUS FROM V$TRANSACTION;

SET TRANSACTION NAME 'ingresos_1';
UPDATE Accounts SET balance = balance + 250 WHERE acctID = 101;
SAVEPOINT tras_ingreso_250;

SELECT NAME, STATUS FROM V$TRANSACTION;

UPDATE Accounts SET balance = balance + 100 WHERE acctID = 202;
SAVEPOINT tras_ingreso_100;

ROLLBACK TO SAVEPOINT tras_ingreso_250;
SELECT * FROM Accounts;
SELECT NAME, STATUS FROM V$TRANSACTION;

UPDATE Accounts SET balance = balance + 300 WHERE acctID = 202;
ROLLBACK;
SELECT * FROM Accounts;
SELECT NAME, STATUS FROM V$TRANSACTION;

SET TRANSACTION NAME 'ingresos_2';
SELECT NAME, STATUS FROM V$TRANSACTION;

UPDATE Accounts SET balance = balance + 150 WHERE acctID = 101;
UPDATE Accounts SET balance = balance + 200 WHERE acctID = 202;

SELECT NAME, STATUS FROM V$TRANSACTION;

COMMIT;

SELECT * FROM Accounts;
SELECT NAME, STATUS FROM V$TRANSACTION;
```

```
SQL> SELECT NAME, STATUS FROM V$TRANSACTION;
ninguna fila seleccionada

SQL> SET TRANSACTION NAME 'ingresos_1';
Transacción definida.

SQL> UPDATE Accounts SET balance = balance + 250 WHERE acctID = 101;
1 fila actualizada.

SQL> SAVEPOINT tras_ingreso_250;
Punto de grabación creado.

SQL> SELECT NAME, STATUS FROM V$TRANSACTION;
NAME                 STATUS
-------------------- ----------------
ingresos_1           ACTIVE

SQL> UPDATE Accounts SET balance = balance + 100 WHERE acctID = 202;
1 fila actualizada.

SQL> SAVEPOINT tras_ingreso_100;
Punto de grabación creado.

SQL> ROLLBACK TO SAVEPOINT tras_ingreso_250;
Rollback terminado.

SQL> SELECT * FROM Accounts;
   ACCTID    BALANCE
---------- ----------
      101       1250
      202       2000

SQL> SELECT NAME, STATUS FROM V$TRANSACTION;
NAME                         STATUS
---------------------------  ----------------
ingresos_1                   ACTIVE

SQL> UPDATE Accounts SET balance = balance + 300 WHERE acctID = 202;
1 fila actualizada.

SQL> ROLLBACK;
Rollback terminado.

SQL> SELECT * FROM Accounts;
```

```
   ACCTID    BALANCE
---------- ----------
       101       1000
       202       2000

SQL> SELECT NAME, STATUS FROM V$TRANSACTION;
ninguna fila seleccionada

SQL> SET TRANSACTION NAME 'ingresos_2';
Transacción definida.

SQL> SELECT NAME, STATUS FROM V$TRANSACTION;
ninguna fila seleccionada

SQL> UPDATE Accounts SET balance = balance + 150 WHERE acctID = 101;
1 fila actualizada.

SQL> UPDATE Accounts SET balance = balance + 200 WHERE acctID = 202;
1 fila actualizada.

SQL> SELECT NAME, STATUS FROM V$TRANSACTION;
NAME                           STATUS
------------------------------ ----------------
ingresos_2                     ACTIVE

SQL> COMMIT;
Confirmación terminada.

SQL> SELECT * FROM Accounts;
   ACCTID    BALANCE
---------- ----------
       101       1150
       202       2200

SQL> SELECT NAME, STATUS FROM V$TRANSACTION;
ninguna fila seleccionada
```

A continuación se ejecutan las siguientes instrucciones, ¿se produce algún problema?. Obtenga conclusiones..

```
SET TRANSACTION NAME 'uno';
UPDATE Accounts SET balance = balance + 250 WHERE acctID = 101;
SET TRANSACTION NAME 'dos';
```