

# PRÁCTICA 4 ORACLE – BLOQUEOS Y TRANSACCIONES

## APARTADO 1 – BLOQUEOS (SELECT)

1. Desde Oracle SQL Developer, crear la siguiente tabla de cuentas, insertar tuplas y confirmarlas:

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

Salida de Script:

```
table CUENTAS creado.  
1 filas insertadas.  
1 filas insertadas.  
confirmado.
```

2. Abrir dos sesiones T1 y T2 de SQLPlus con tu usuario y contraseña

```
sqlplus USU_GII_MATANOTARIO/dr1234567890@BDc
```

3. Deshabilitar la autoconfirmación en ambas sesiones con:

```
SET AUTOCOMMIT OFF
```

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

```
UPDATE cuentas  
SET saldo = saldo + 100  
WHERE numero = 123;
```

5. Desde T2 consultar el saldo de la cuenta 123. ¿Cuánto es su saldo?

```
SELECT saldo  
FROM cuentas  
WHERE numero = 123;
```

Su saldo es 500 euros.

6. Desde T1 confirmar los datos con:

```
COMMIT;
```

7. Desde T2 consultar el saldo de la cuenta 123. ¿Cuánto es su saldo?

```
SELECT saldo  
FROM cuentas  
WHERE numero = 123;
```

Su saldo es 500 euros.

## APARTADO 2 – BLOQUEOS (UPDATE)

### 1. Deshabilitar la autoconfirmación en T2 con:

```
SET AUTOCOMMIT OFF;
```

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

```
UPDATE cuentas  
SET saldo = saldo + 100  
WHERE numero = 123;
```

### 3. Desde T2 aumenta 200 euros el saldo de la cuenta 123. ¿Se puede? ¿Qué le pasa a T2?

```
UPDATE cuentas  
SET saldo = saldo + 200  
WHERE numero = 123;
```

No se puede. T2 se queda bloqueado esperando.

### 4. Desde T1 confirmar los datos con:

```
COMMIT;
```

Al confirmar los datos, T2 deja de estar bloqueado y se hace el cambio.

### 5. Desde T1 consultar el saldo de la cuenta 123. ¿Cuánto es su saldo?

```
SELECT saldo  
FROM cuentas  
WHERE numero = 123;
```

Su saldo es 600 euros.

### 6. Desde T2 confirmar los datos con:

```
COMMIT;
```

### 7. Desde T1 consultar el saldo de la cuenta 123. ¿Cuánto es su saldo?

```
SELECT saldo  
FROM cuentas  
WHERE numero = 123;
```

Su saldo es 800 euros.

## APARTADO 3 – BLOQUEOS (DEADLOCK)

### 1. Desde 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.

```
UPDATE cuentas  
SET saldo = saldo + 200  
WHERE numero = 456;
```

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

```
UPDATE cuentas  
SET saldo = saldo + 300  
WHERE numero = 456;
```

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

```
UPDATE cuentas  
SET saldo = saldo + 400  
WHERE numero = 123;
```

### ¿Qué ocurre?

Lo que ocurre es que para que no se queden bloqueados ambos terminales, Oracle se carga la petición del primer terminal indicándonoslo mediante el siguiente mensaje:

```
ERROR en línea 1:  
ORA-00060: detectado interbloqueo mientras se esperaba un recurso
```

## **APARTADO 4 – NIVELES DE AISLAMIENTO**

### 1. En T1:

```
ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;
```

### 2. En T1:

```
SELECT SUM(saldo) FROM cuentas;
```

El valor de SUM(saldo) es 1800 euros.

### 3. En T2:

```
UPDATE cuentas SET saldo = saldo + 100;  
COMMIT;
```

### 4. En T1:

```
SELECT SUM(saldo) FROM cuentas;
```

### ¿Qué ha pasado?

Lo que ha pasado es que en el nivel de aislamiento “serializable”, sólo se puede acceder en una consulta a los datos que se confirmaron en el inicio de la transacción y a los realizados por la propia transacción a través de inserciones, actualizaciones y eliminaciones.

### 5. En T1:

```
ALTER SESSION SET ISOLATION_LEVEL = READ COMMITTED;
```

### 6. En T1:

```
SELECT SUM(saldo) FROM cuentas;
```

El valor de SUM(saldo) es 2000 euros.

### 7. En T2:

```
UPDATE cuentas SET saldo = saldo + 100;  
COMMIT;
```

## 8. En T1:

```
SELECT SUM(saldo) FROM cuentas;
```

¿Qué ha pasado? Explicar si hay alguna diferencia según los niveles de aislamiento.

El valor de SUM(saldo) en este último caso es 2220 euros.

Lo que ocurre es que ahora estamos utilizando el nivel de aislamiento “read committed” Con este nivel de aislamiento, cada consulta sólo puede ver los datos confirmados antes de la consulta. En cambio, como he dicho en el punto 4, con el nivel de aislamiento “serializable” se puede acceder en una consulta a los datos que se confirmaron en el inicio de la transacción y a los realizados por la propia transacción a través de inserciones, actualizaciones y eliminaciones.

## **APARTADO 5 – TRANSACCIONES**

### 1. Crea las tablas y secuencias para los identificadores:

```
CREATE TABLE butacas(id number(8) primary key,  
                      evento varchar(30),  
                      fila    varchar(10),  
                      columna varchar(10));
```

```
CREATE TABLE reservas(id number(8) primary key,  
                      evento varchar(30),  
                      fila    varchar(10),  
                      columna varchar(10));
```

```
CREATE SEQUENCE Seq_Butacas INCREMENT BY 1 START WITH 1 NOMAXVALUE;
```

```
CREATE SEQUENCE Seq_Reservas INCREMENT BY 1 START WITH 1 NOMAXVALUE;
```

Salida de Script:

```
table BUTACAS creado.  
table RESERVAS creado.  
sequence SEQ_BUTACAS creado.  
sequence SEQ_RESERVAS creado.
```

### 2. Inserta algunos valores de prueba en butacas:

```
INSERT INTO butacas VALUES (Seq_Butacas.NEXTVAL, 'Circo', '1', '1');  
INSERT INTO butacas VALUES (Seq_Butacas.NEXTVAL, 'Circo', '1', '2');  
INSERT INTO butacas VALUES (Seq_Butacas.NEXTVAL, 'Circo', '1', '3');  
COMMIT;
```

Salida de Script:

```
1 filas insertadas.  
1 filas insertadas.  
1 filas insertadas.  
confirmado.
```

3. Prueba el script desde SQL Developer reservando la fila 1, columna 1 para 'Circo'. Antes hay que modificar las rutas que hacen referencia a los otros scripts a la carpeta en donde se hayan depositado.

```

--SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET serveroutput ON size 1000000;
set define on;
SET echo OFF;
SET verify OFF;
def v_evento='Circo';
def v_fila='1';
def v_columna='1';
variable v_error char(20)
/
declare
    v_existe varchar(20) default null;
begin
    select count(*) into v_existe from butacas where evento='&v_evento'
and fila='&v_fila' and columna='&v_columna';
    if v_existe<>'0' then
        select count(*) into v_existe from reservas where
evento='&v_evento' and fila='&v_fila' and columna='&v_columna';
        if v_existe='0' then
            dbms_output.put_line('INFO: Se intenta reservar.');
```

```

            :v_error:='false';
        else
            dbms_output.put_line('ERROR: La localidad ya est? reservada.');
```

```

            :v_error:='true';
        end if;
    else
        dbms_output.put_line('ERROR: No existe esa localidad.');
```

```

        :v_error:='true';
    end if;
end;
/
col SCRIPT_COL new_val SCRIPT
select
decode(:v_error,'false','"C:\hlocal\preguntar.sql"', '"C:\hlocal\nopre
guntar.sql"') as SCRIPT_COL from dual;
print :v_error
--prompt 'Valor script: '&SCRIPT
@ &SCRIPT
prompt &v_confirmar
/
begin
    if '&v_confirmar'='s' and :v_error='false' then
        insert into reservas values
(Seq_Reservas.NEXTVAL, '&v_evento', '&v_fila', '&v_columna');
```

```

        dbms_output.put_line('INFO: Localidad reservada.');
```

```

    else
        dbms_output.put_line('INFO: No se ha reservado la localidad.');
```

```

    end if;
end;
/
COMMIT;

```

### Salida de Script:

bloque anónimo terminado  
INFO: Se intenta reservar.

```

SCRIPT_COL
-----
"C:\hlocal\preguntar.sql"

V_ERROR

```

```

-----
false

'?Confirmar la reserva?'
s
bloque anónimo terminado
INFO: Localidad reservada.

confirmado.

```

**4. Intenta reservar de nuevo la misma fila desde la misma consola SQL Developer y comprueba que no sea posible.**

*(Volvemos a ejecutar el Script anterior)*

**Salida de Script:**

```

bloque anónimo terminado
ERROR: La localidad ya est? reservada.

```

```

SCRIPT_COL
-----
"C:\hlocal
o_preguntar.sql"

```

```

V_ERROR
-----

n
bloque anónimo terminado
INFO: No se ha reservado la localidad.

confirmado.

```

**5. Realiza una nueva reserva desde la misma consola para la fila 1, columna 4 para la 'Circo' y comprueba que no es posible porque no existe esa butaca.**

```

--SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET serveroutput ON size 1000000;
set define on;
SET echo OFF;
SET verify OFF;
def v_evento='Circo';
def v_fila='1';
def v_columna='4';
variable v_error char(20)
/
declare
  v_existe varchar(20) default null;
begin
  select count(*) into v_existe from butacas where evento='&v_evento'
and fila='&v_fila' and columna='&v_columna';
  if v_existe<>'0' then
    select count(*) into v_existe from reservas where
evento='&v_evento' and fila='&v_fila' and columna='&v_columna';
    if v_existe='0' then
      dbms_output.put_line('INFO: Se intenta reservar.');
```

```

      :v_error:='false';
    else

```

```

      dbms_output.put_line('ERROR: La localidad ya est? reservada.');
```

```

        :v_error:='true';
    end if;
else
    dbms_output.put_line('ERROR: No existe esa localidad.');
```

:v\_error:='true';
 end if;
end;
/
col SCRIPT\_COL new\_val SCRIPT
select
decode(:v\_error,'false','C:\hlocal\preguntar.sql','C:\hlocal\no\_pre
guntar.sql') as SCRIPT\_COL from dual;
print :v\_error
--prompt 'Valor script: '&SCRIPT
@ &SCRIPT
prompt &v\_confirmar
/
begin
 if '&v\_confirmar'='s' and :v\_error='false' then
 insert into reservas values
(Seq\_Reservas.NEXTVAL,'&v\_evento','&v\_fila','&v\_columna');
 dbms\_output.put\_line('INFO: Localidad reservada.');

else
 dbms\_output.put\_line('INFO: No se ha reservado la localidad.');

end if;
 end;
/

#### Salida de Script:

bloque anónimo terminado  
 ERROR: No existe esa localidad.

```

SCRIPT_COL
-----
"C:\hlocal
o_preguntar.sql"

```

```

V_ERROR
-----

```

n  
 bloque anónimo terminado  
 INFO: No se ha reservado la localidad.

#### 6. Realiza una nueva reserva desde la misma consola para la fila 1, columna 2 para la 'Circo' pero sin realizar aún la confirmación.

```

--SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET serveroutput ON size 1000000;
set define on;
SET echo OFF;
SET verify OFF;
def v_evento='Circo';
def v_fila='1';
def v_columna='2';
variable v_error char(20)
/
declare
    v_existe varchar(20) default null;

```

```

begin
    select count(*) into v_existe from butacas where evento='&v_evento'
    and fila='&v_fila' and columna='&v_columna';
    if v_existe<>'0' then
        select count(*) into v_existe from reservas where
        evento='&v_evento' and fila='&v_fila' and columna='&v_columna';
        if v_existe='0' then
            dbms_output.put_line('INFO: Se intenta reservar.');
```

```

            :v_error:='false';
        else
            dbms_output.put_line('ERROR: La localidad ya est? reservada.');
```

```

            :v_error:='true';
        end if;
    else
        dbms_output.put_line('ERROR: No existe esa localidad.');
```

```

        :v_error:='true';
    end if;
end;
/
col SCRIPT_COL new_val SCRIPT
select
decode(:v_error,'false','C:\hlocal\preguntar.sql','C:\hlocal\nopre
guntar.sql') as SCRIPT_COL from dual;
print :v_error
--prompt 'Valor script: '&SCRIPT
@ &SCRIPT
prompt &v_confirmar
/
begin
    if '&v_confirmar'='s' and :v_error='false' then
        insert into reservas values
        (Seq_Reservas.NEXTVAL,'&v_evento','&v_fila','&v_columna');
```

```

        dbms_output.put_line('INFO: Localidad reservada.');
```

```

    else
        dbms_output.put_line('INFO: No se ha reservado la localidad.');
```

```

    end if;
end;
/

```

### Salida de Script:

```

bloque anónimo terminado
INFO: Se intenta reservar.

```

```

SCRIPT_COL
-----
"C:\hlocal\preguntar.sql"

V_ERROR
-----
false

```

```

'?Confirmar la reserva?'
s
bloque anónimo terminado
INFO: Localidad reservada.

```

### 7. Abre una nueva instancia de SQL Developer y realiza la misma reserva anterior desde esta instancia. Confirma la reserva.

*(Abrimos una nueva instancia de SQL Developer)*



```

--SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET serveroutput ON size 1000000;
set define on;
SET echo OFF;
SET verify OFF;
def v_evento='Circo';
def v_fila='1';
def v_columna='2';
variable v_error char(20)
/
declare
    v_existe varchar(20) default null;
begin
    select count(*) into v_existe from butacas where evento='&v_evento'
and fila='&v_fila' and columna='&v_columna';
    if v_existe<>'0' then
        select count(*) into v_existe from reservas where
evento='&v_evento' and fila='&v_fila' and columna='&v_columna';
        if v_existe='0' then
            dbms_output.put_line('INFO: Se intenta reservar.');
```

```

            :v_error:='false';
        else
            dbms_output.put_line('ERROR: La localidad ya est? reservada.');
```

```

            :v_error:='true';
        end if;
    else
        dbms_output.put_line('ERROR: No existe esa localidad.');
```

```

        :v_error:='true';
    end if;
end;
/
col SCRIPT_COL new_val SCRIPT
select
decode(:v_error,'false','"C:\hlocal\preguntar.sql"', '"C:\hlocal\nopre
guntar.sql"') as SCRIPT_COL from dual;
print :v_error
--prompt 'Valor script: '&SCRIPT
@ &SCRIPT
prompt &v_confirmar
/
begin
    if '&v_confirmar'='s' and :v_error='false' then
        insert into reservas values
(Seq_Reservas.NEXTVAL, '&v_evento', '&v_fila', '&v_columna');
```

```

        dbms_output.put_line('INFO: Localidad reservada.');
```

```

    else
        dbms_output.put_line('INFO: No se ha reservado la localidad.');
```

```

    end if;
end;
/
COMMIT;

```

### Salida de Script:

bloque anónimo terminado  
INFO: Se intenta reservar.

```

SCRIPT_COL
-----
"C:\hlocal\preguntar.sql"

V_ERROR

```

```

-----
false

'?Confirmar la reserva?'
s
bloque anónimo terminado
INFO: Localidad reservada.

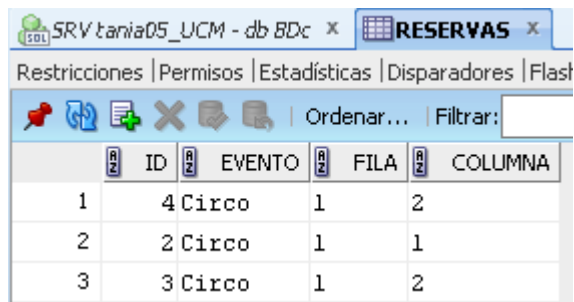
confirmado.

```

### 8. Confirma la reserva del punto 6. ¿Qué sucede?

```
COMMIT;
```

Lo que sucede es que se realiza una reserva en la misma butaca 2 veces tal y como podemos ver en la siguiente imagen:



	ID	EVENTO	FILA	COLUMNA
1	4	Circo	1	2
2	2	Circo	1	1
3	3	Circo	1	2

### 9. Modifica el script para resolver el punto anterior.

He arreglado el script añadiendo al final las siguientes líneas de código:

```

/
declare
  v_existe_2 number;
begin
  select count(*) into v_existe_2 from reservas where
    evento='&v_evento' and fila='&v_fila' and columna='&v_columna';
  if v_existe_2 > 1 then
    ROLLBACK;
  end if;
end;
/

```