CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

Tratamiento de datos IV

Transacciones y bloqueos

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

Atomicidad, Consistencia, aIslamiento y Durabilidad

A.C.I.D.

A.C.I.D.

- Atomicidad:
  - Las transacciones son completas: o se ejecutan todas las acciones o ninguna.
- Consistencia: (Integridad).
  - Cualquier transacción llevará a la base de datos desde un estado válido a otro también válido, cumpliendo las restricciones de integridad.
- Aislamiento:
  - La realización de dos transacciones sobre la misma información son independientes y no generan ningún tipo de error.
- Durabilidad: (Persistencia).
  - Una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Transacciones

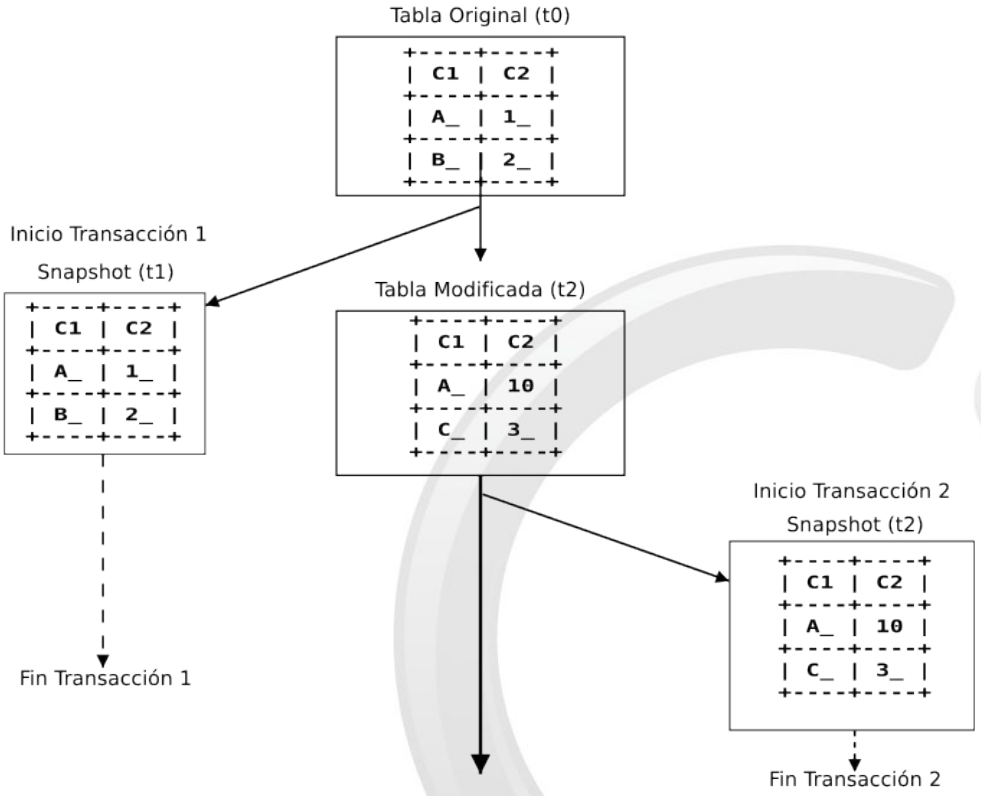
```
BEGIN;

UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';

UPDATE branches SET balance = balance - 100.00
    WHERE name = (
        SELECT branch_name FROM accounts
        WHERE name = 'Alice'
    );

UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Bob';

UPDATE branches SET balance = balance + 100.00
    WHERE name = (
        SELECT branch_name FROM accounts
        WHERE name = 'Bob'
    );

COMMIT;
```

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

MVCC

Control de Concurrencia Multiversión (PostgreSQL):

cada sentencia SQL ve una instantánea de los datos tal como estaba en algún momento anterior.



CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

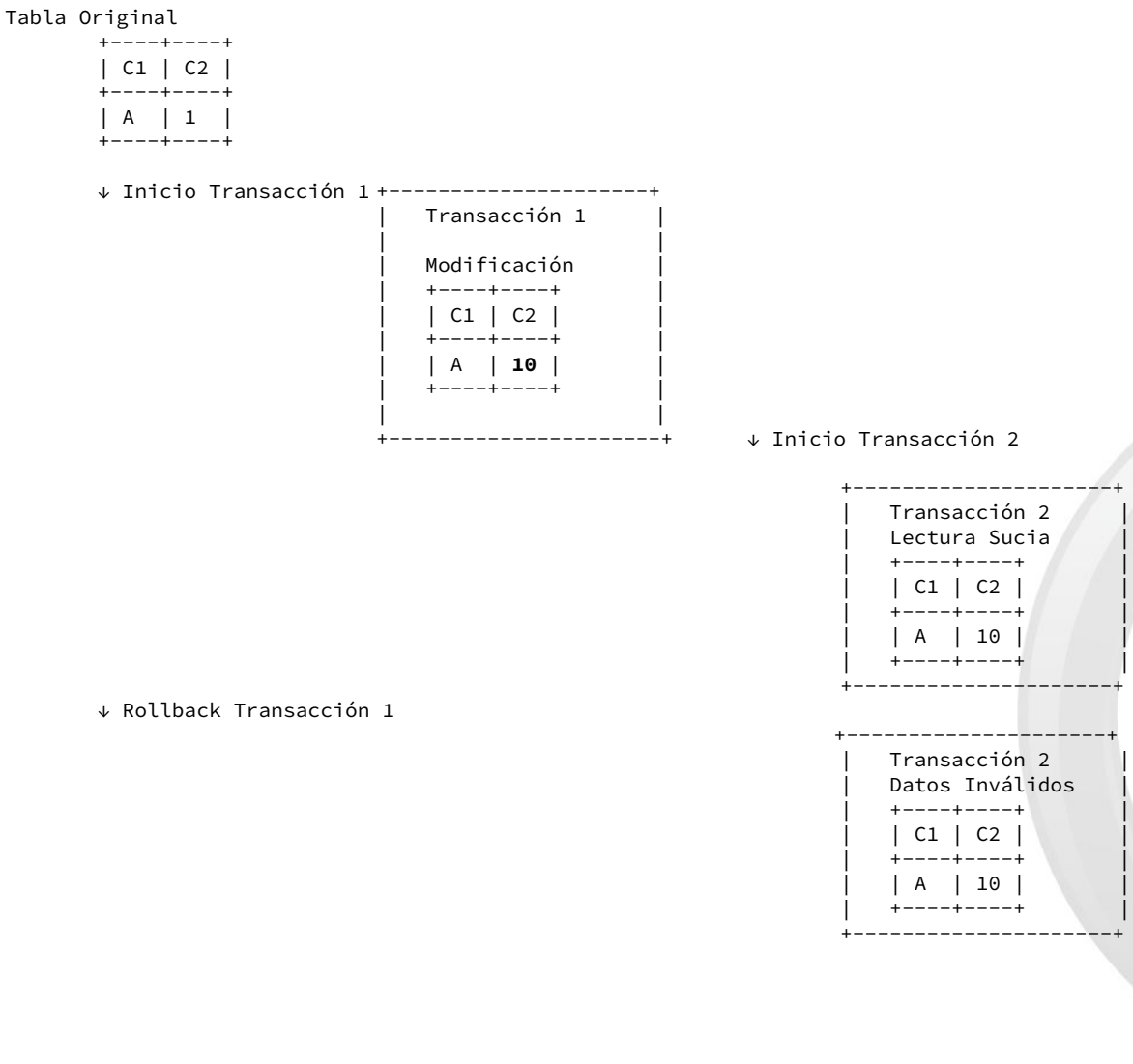
Centro Integrado de Formación Profesional

Problemas en transacciones

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

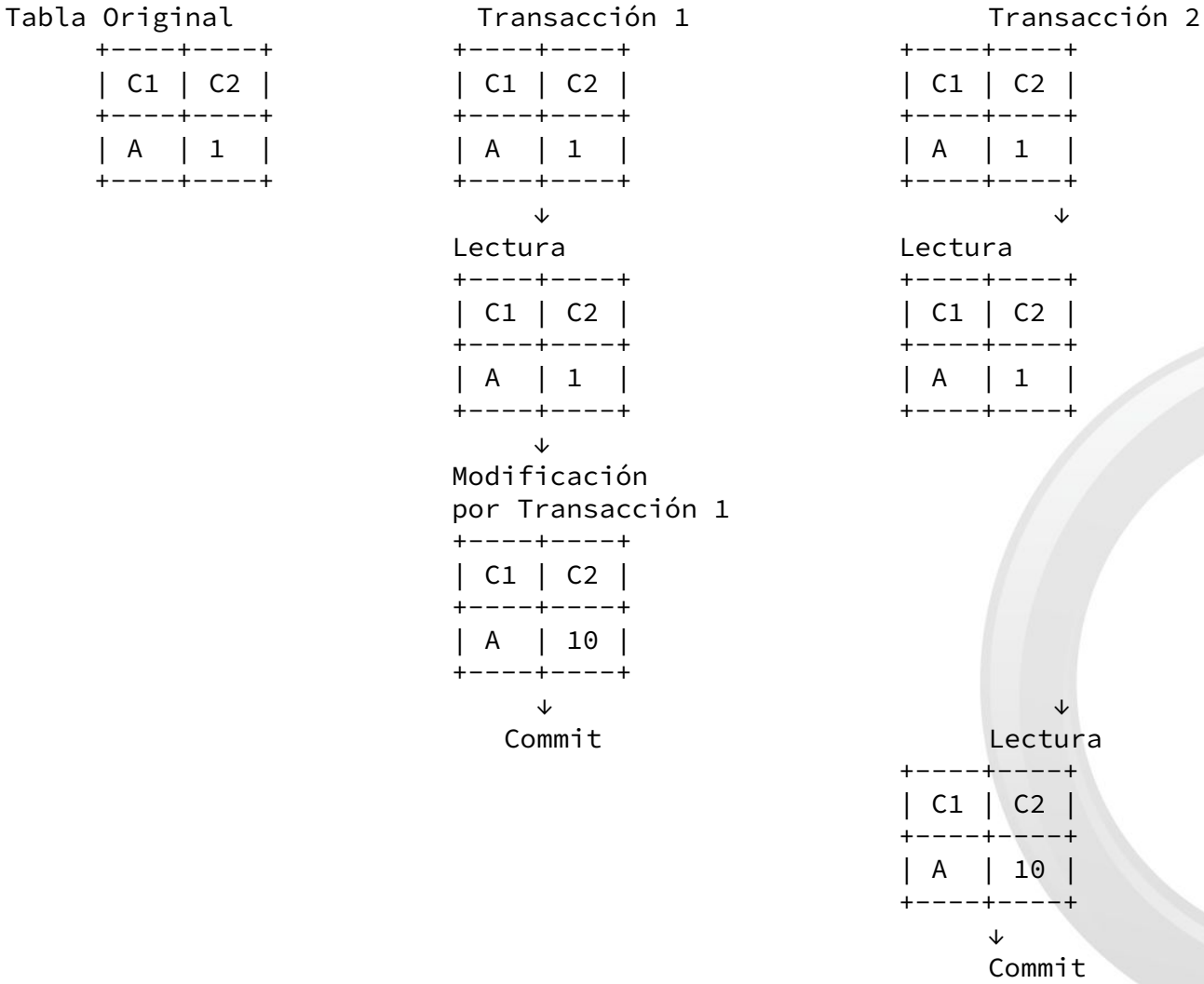
Centro Integrado de Formación Profesional

Lectura sucia



CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Lectura No Repetible



CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

Lectura Fantasma

Tabla Original

C1	C2
A	1
B	2

Transacción 1

C1	C2
A	1
B	2

↓

Inserción
por Transacción 1

C1	C2
C	3

↓

Commit

Transacción 2

C1	C2
A	1
B	2

↓

Lectura

C1	C2
A	1
B	2

↓

Lectura

C1	C2
A	1
B	2
C	3

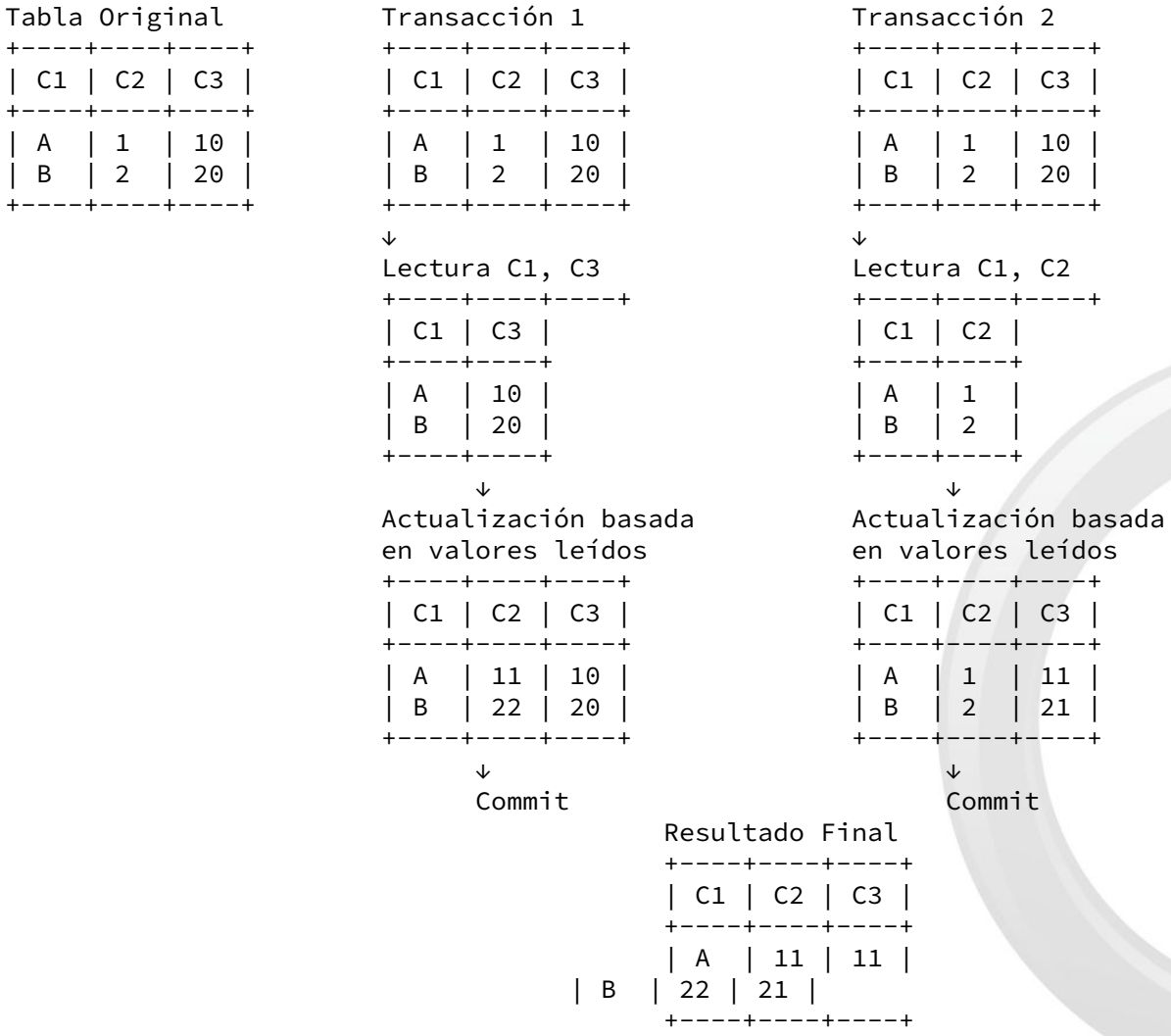
↓

Commit

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

Anomalías de serialización



CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Tablas para la práctica

- Para esta sesión, utilizaremos el siguiente script de base de datos, el cual podemos cargar en uno de los esquemas que tengamos en nuestro gestor:

```
CREATE TABLE TAVIONES(TIPO VARCHAR(15) PRIMARY KEY, CAPACIDAD INT);

CREATE TABLE TVUELOS(
    ID VARCHAR(10) PRIMARY KEY, TIPOAVION VARCHAR(15) REFERENCES TAVIONES(TIPO), PASAJEROS INT, DESTINO VARCHAR(20) );

CREATE TABLE TPASAJEROS(NOMBRE VARCHAR(20), VUELO VARCHAR(10) REFERENCES TVUELOS, PRIMARY KEY(NOMBRE,VUELO));

INSERT INTO TAVIONES VALUES ('AIRBUS A300',120);

INSERT INTO TAVIONES VALUES ('BOEING 707',210);

INSERT INTO TAVIONES VALUES ('DOUGLAS DC-9',190);

INSERT INTO TVUELOS VALUES ('LHE 100','AIRBUS A300',110,'ROMA-FIUMICINO');

INSERT INTO TVUELOS VALUES ('IBE 398','BOEING 707',209,'BARCELONA-EL PRAT');
```

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Condiciones de la tutoría

- El número de pasajeros siempre debe ser menor que la capacidad del avión.
- Simularemos la utilización de la base de datos desde 2 agencias de viajes
  - Agencia 1
  - Agencia 2

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

Modos de acceso:
Read Write/Read Only

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

read write

por defecto

Suponemos que en el terminal A1 se quiere saber, en primer lugar, el número total de vuelos que hay, para desglosar, posteriormente, este total por clases.

AGENCIA 1

```
SELECT TIPOAVION, COUNT(*)
FROM TVUELOS GROUP BY TIPOAVION;



SELECT TIPOAVION,COUNT(*)
FROM TVUELOS GROUP BY TIPOAVION;

SELECT TIPOAVION,COUNT(*)
FROM TVUELOS GROUP BY TIPOAVION;
```

AGENCIA 2

```
BEGIN;
INSERT INTO TVUELOS VALUES
('TWA','AIRBUS A300',89,'MADRID-BARAJAS');


COMMIT;
```

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

read write

por defecto

AGENCIA 1

```
SQL> SELECT TIPOAVION, COUNT(*)
FROM TVUELOS GROUP BY TIPOAVION;
  2
TIPOAVION      COUNT(*)
-------------- ----------
Boeing 707          1
Airbus A300         1

SQL> SELECT TIPOAVION,COUNT(*)
FROM TVUELOS GROUP BY TIPOAVION;
  2
TIPOAVION      COUNT(*)
-------------- ----------
Boeing 707          1
Airbus A300         1

SQL> SELECT TIPOAVION,COUNT(*)
FROM TVUELOS GROUP BY TIPOAVION;
  2
TIPOAVION      COUNT(*)
-------------- ----------
Boeing 707          1
Airbus A300         2

SQL>
```

AGENCIA 2

```
SQL> BEGIN;

SQL> INSERT INTO TVUELOS VALUES ('TWA', 'AIRBUS A300', 89,
'MADRID-BARAJAS');

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

La agencia 1 sólo ve los cambios que hace la agencia 2 cuando ésta hace **commit**, es decir cuando finaliza su transacción.

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

read only

AGENCIA 1

```
scott=> begin;
BEGIN
scott=*> show transaction_read_only;
 transaction_read_only
-----------------------
 off
(1 fila)

scott=*> INSERT INTO TVUELOS VALUES
('TWB','AIRBUS A300',89,'MADRID-BARAJAS');
INSERT 0 1
scott=*> set transaction read only;
SET
scott=*> INSERT INTO TVUELOS VALUES
('TWC','AIRBUS A300',89,'MADRID-BARAJAS');
ERROR:  no se puede ejecutar INSERT en una transacción de sólo lectura
scott=!> rollback;
scott=> select * from tvuelos;
   id   |  tipoavion   | pasajeros |   destino
--------+-------------+-----------+-------------------
 LHE 100 | AIRBUS A300 |       110 | ROMA-FIUMICINO
 IBE 398 | BOEING 707  |       209 | BARCELONA-EL PRAT
 TWA    | AIRBUS A300 |        89 | MADRID-BARAJAS
(3 filas)
```

En el modo de acceso READ ONLY no se pueden llevar a cabo operaciones de modificación de datos.

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

NIVELES DE AISLAMIENTO

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Niveles de aislamiento

A mayor nivel de aislamiento menos posibilidad de que se produzcan interacciones, por tanto mayor seguridad pero también menor grado de concurrencia.

SERIALIZABLE

Los cambios realizados en otros terminales no afectan, aunque las otras sesiones hayan hecho commit y por tanto hayan grabado físicamente los cambios.

Si se intenta modificar una fila ya modificada por otra sesión se obtiene un error "No se puede serializar el acceso para esa transacción".

READ COMMITTED

Cada sesión ve los cambios de las otras cuando éstas han hecho commit.

Es el valor por defecto.

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Niveles de aislamiento

SERIALIZABLE

AGENCIA 1

```
scott=> BEGIN;
BEGIN
scott=*> UPDATE TVUELOS SET PASAJEROS=50 WHERE ID='LHE
100';
UPDATE 1
scott=*> SELECT * FROM TVUELOS;
   id   |  tipoavion   | pasajeros |   destino
--------+-------------+-----------+-------------------
 IBE 398 | BOEING 707   |       209 | BARCELONA-EL PRAT
 TWA     | AIRBUS A300 |        89 | MADRID-BARAJAS
 LHE 100 | AIRBUS A300 |       50 | ROMA-FIUMICINO
(3 filas)
```

AGENCIA 2

```
scott=> BEGIN;
BEGIN
scott=*> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET
```

```
scott=*> UPDATE TVUELOS SET PASAJEROS=0 WHERE ID='LHE 100';
▌  ← TERMINAL BLOQUEADO
```

Esto se debe a que UPDATE bloquea filas. Cuando la agencia 1 hace update bloquea, A2 intenta bloquear lo mismo y queda en espera de que A1 desbloquee. Este comportamiento no está relacionado con serializable.

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Niveles de aislamiento

SERIALIZABLE

AGENCIA 1

```
scott=> BEGIN;
BEGIN
scott=*> UPDATE TVUELOS SET PASAJEROS=50 WHERE ID='LHE
100';
UPDATE 1
scott=*> SELECT * FROM TVUELOS;
   id   |  tipoavion   | pasajeros |   destino
--------+-------------+-----------+-------------------
 IBE 398 | BOEING 707   |       209 | BARCELONA-EL PRAT
 TWA     | AIRBUS A300 |        89 | MADRID-BARAJAS
 LHE 100 | AIRBUS A300 |        50 | ROMA-FIUMICINO
(3 filas)


scott=*> COMMIT;← desbloquea el terminal de la A.2
COMMIT
```

AGENCIA 2

```
scott=> BEGIN;
BEGIN
scott=*> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET




scott=*> UPDATE TVUELOS SET PASAJEROS=0 WHERE ID='LHE 100';
ERROR:  no se pudo serializar el acceso debido a un update
concurrente
scott=!> rollback;
```

Esto se debe a que UPDATE bloquea filas. Cuando la agencia 1 hace update bloquea, A2 intenta bloquear lo mismo y queda en espera de que A1 desbloquee. Este comportamiento no está relacionado con serializable.

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Niveles de aislamiento

READ COMMITTED

AGENCIA 1

```
scott=> BEGIN;
BEGIN




Sscott=*> UPDATE TVUELOS SET PASAJEROS=50 WHERE ID='LHE 100';
  ← terminal bloqueado



1 row updated. ← con READ COMMITTED no devuelve error

scott=*>
```

AGENCIA 2

```
scott=> BEGIN;
BEGIN

scott=*> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
scott=*> UPDATE TVUELOS SET PASAJEROS=0 WHERE ID='LHE 100';
UPDATE 1




scott=*> COMMIT; ← desbloquea el terminal de la A.1
COMMIT

scott=*>
```

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

Bloqueos

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Bloqueos

Supongamos que ahora llegan 2 clientes, uno a cada agencia, y que ambos quieren reservar plaza en el vuelo 'IBE 398'.

El empleado de cada agencia mira a ver si el vuelo aún está libre y si lo está indica que ha llegado un nuevo pasajero, incrementando por tanto el número de pasajeros en el vuelo. Suponemos además que en la segunda agencia se van realizando las acciones un poco más tarde.

Agencia 1

```
scott=> BEGIN;
BEGIN
scott=*> SET TRANSACTION READ WRITE;
SET
scott=*>
```

Agencia 2

```
scott=> BEGIN;
BEGIN
scott=*> SET TRANSACTION READ WRITE;
SET
scott=*>
```

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Bloqueos

AGENCIA 1

```
scott=*> SELECT CAPACIDAD,PASAJEROS
FROM TVUELOS,TAVIONES
WHERE TVUELOS.TIPOAVION=TAVIONES.TIPO
AND ID='IBE 398';
 capacidad | pasajeros
-----------+-----------
       210 |         209
(1 fila)




scott=*> UPDATE TVUELOS SET PASAJEROS = PASAJEROS + 1
WHERE ID='IBE 398';
UPDATE 1


scott=*> COMMIT;
COMMIT
```

AGENCIA 2

```
cott=*> SELECT CAPACIDAD,PASAJEROS
FROM TVUELOS,TAVIONES
WHERE TVUELOS.TIPOAVION=TAVIONES.TIPO
AND ID='IBE 398';
 capacidad | pasajeros
-----------+-----------
       210 |         209
(1 fila)


scott=*> UPDATE TVUELOS SET PASAJEROS = PASAJEROS + 1
WHERE ID='IBE 398';
                ← TERMINAL BLOQUEADO



UPDATE 1
scott=*> COMMIT;
COMMIT  ← Hemos vendido más pasajes que plazas disponibles
scott=>
```

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Bloqueos

SELECT FOR UPDATE

AGENCIA 1

```
scott=> BEGIN;
BEGIN

scott=*> UPDATE TVUELOS SET PASAJEROS = 209
WHERE  ID='IBE 398';
UPDATE 1

scott=*> SELECT CAPACIDAD,PASAJEROS
FROM TVUELOS,TAVIONES
WHERE TVUELOS.TIPOAVION=TAVIONES.TIPO
AND ID='IBE 398'
FOR UPDATE NOWAIT;

 capacidad | pasajeros
-----------+-----------
       210 |       209
(1 fila)



scott=*> UPDATE TVUELOS SET PASAJEROS = PASAJEROS + 1
WHERE ID='IBE 398';
UPDATE 1

scott=*> COMMIT;
COMMIT

scott=>
```

AGENCIA 2

```
scott=> BEGIN;
BEGIN
scott=*> SELECT CAPACIDAD,PASAJEROS
FROM TVUELOS,TAVIONES
WHERE TVUELOS.TIPOAVION=TAVIONES.TIPO
AND ID='IBE 398'
FOR UPDATE NOWAIT;
ERROR:  no se pudo bloquear un candado en la fila de la relación
«tvuelos»



scott=!> ROLLBACK;
ROLLBACK
scott=> BEGIN;
BEGIN
scott=*> SELECT CAPACIDAD,PASAJEROS FROM TVUELOS,TAVIONES
WHERE TVUELOS.TIPOAVION=TAVIONES.TIPO AND ID='IBE 398'
FOR UPDATE NOWAIT;
 capacidad | pasajeros
-----------+-----------
       210 |       210
(1 fila)

scott=*>
```

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

Deadlocks (Interbloqueos)

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Deadlocks

Dos operaciones concurrentes quedan esperando cada una a la otra.

Terminal 1

```
CREATE TABLE deadlock (id INTEGER, fld VARCHAR(1));
INSERT INTO deadlock VALUES (1,'A');
INSERT INTO deadlock VALUES (2,'B');

scott=> BEGIN;
BEGIN
scott=*> UPDATE deadlock SET fld = 'M' WHERE id = 1;
UPDATE 1


UPDATE deadlock SET fld = 'X' WHERE id = 2;
▌  ← TERMINAL BLOQUEADO


ERROR at line 1:
ORA-00060: deadlock detected while waiting for
resource
```

Terminal 2

```
BEGIN;
UPDATE deadlock SET fld = 'N' WHERE id = 2;


UPDATE deadlock SET fld = 'Y' WHERE id = 1;
▌  ← TERMINAL BLOQUEADO
.
ERROR:  se ha detectado un deadlock
DETALLE:  El proceso 103854 espera ShareLock en transacción
5918; bloqueado por proceso 78456.
El proceso 78456 espera ShareLock en transacción 5919;
bloqueado por proceso 103854.
SUGERENCIA:  Vea el registro del servidor para obtener detalles
de las consultas.
CONTEXTO:  mientras se actualizaba la tupla (0,1) en la
relación «deadlock»
scott=!> ROLLBACK;
```

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

Rollbacks y Savepoints

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES
Centro Integrado de Formación Profesional

Rollbacks

La instrucción rollback permite al usuario finalizar la transacción sin éxito. De esta forma, ninguno de los cambios realizados tras el último commit se llegará a grabar físicamente en la base de datos.

1) Iniciamos la reserva

```
scott=> BEGIN;
BEGIN
scott=*> UPDATE TVUELOS SET PASAJEROS=PASAJEROS +1 WHERE
ID='LHE 100';
UPDATE 1
scott=*> INSERT INTO TPASAJEROS VALUES('HERMINIA','LHE 100');
INSERT 0 1
scott=*> SELECT * FROM TVUELOS;
   id     | tipoavion   | pasajeros |  destino
---------+-------------+-----------+-------------------
 TWA     | AIRBUS A300 |        89 | MADRID-BARAJAS
 IBE 398 | BOEING 707  |       210 | BARCELONA-EL PRAT
 LHE 100 | AIRBUS A300 |         1 | ROMA-FIUMICINO
(3 filas)

scott=*> SELECT * FROM TPASAJEROS;
  nombre  |  vuelo
----------+---------
 HERMINIA | LHE 100
(1 fila)

scott=*>
```

2) Deshacemos la reserva

```
scott=*> ROLLBACK;
ROLLBACK
scott=> SELECT * FROM TVUELOS;
   id     | tipoavion   | pasajeros |  destino
---------+-------------+-----------+-------------------
 TWA     | AIRBUS A300 |        89 | MADRID-BARAJAS
 IBE 398 | BOEING 707  |       210 | BARCELONA-EL PRAT
 LHE 100 | AIRBUS A300 |         0 | ROMA-FIUMICINO
(3 filas)

scott=> SELECT * FROM TPASAJEROS;
 nombre | vuelo
--------+-------
(0 filas)

scott=>
```

CIFP Carlos III
INFORMÁTICA Y COMUNICACIONES

Centro Integrado de Formación Profesional

Savepoints

Dentro de una transacción se pueden además definir puntos intermedios de rollback, nombrando un **“savepoint”**

```
scott=> BEGIN;
BEGIN
scott=*> INSERT INTO TAVIONES VALUES('BAE HARRIER', 2);
INSERT 0 1
scott=*> INSERT INTO TAVIONES VALUES('CONCORDE', 240);
INSERT 0 1
scott=*> SAVEPOINT TURURU;
SAVEPOINT
scott=*> INSERT INTO TVUELOS VALUES('UNA 100', 'CONCORDE',10, 'BARAJAS');
INSERT 0 1
scott=*> INSERT INTO TVUELOS VALUES('UTA 600', 'BOEINJ', 20,'PEKÍN');
ERROR:  inserción o actualización en la tabla «tvuelos» viola la llave foránea
«tvuelos_tipoavion_fkey»
DETALLE:  La llave (tipoavion)=(BOEINJ) no está presente en la tabla «taviones».
scott=!> ROLLBACK TO TURURU;
ROLLBACK
scott=*> INSERT INTO TPASAJEROS VALUES('NICASIO','LHE 100');
INSERT 0 1
scott=*> COMMIT;
COMMIT
scott=>
```