

EXERCISES QUE TRABAJAREMOS EN EL CUE:

- EXERCISE 1: CREAR UNA BASE DE DATOS EN POSTGRES.
- EXERCISE 2: REALIZAR MODIFICACIONES EN LA BASE DE DATOS SIN TRANSACCIONES.
- EXERCISE 3: REALIZAR TRANSACCIONES CON BEGIN, COMMIT Y ROLLBACK.

El objetivo de este ejercicio es plantear una guía de entendimiento sobre las transacciones en una base de datos, específicamente con node-postgres, y el uso de las instrucciones: Begin, Commit, y Rollback.

EXERCISE 1: CREAR UNA BASE DE DATOS EN POSTGRES

Para realizar la siguiente práctica, haremos uso del ejemplo clásico de transacciones (tx) en un banco. Supongamos que deseamos transferir saldo de un cliente A, a un cliente B; para ello, actualizamos las cuentas disminuyendo el saldo en la cuenta A, y aumentándolo en la B. Para ejemplificar el ejercicio, crearemos la base de datos.

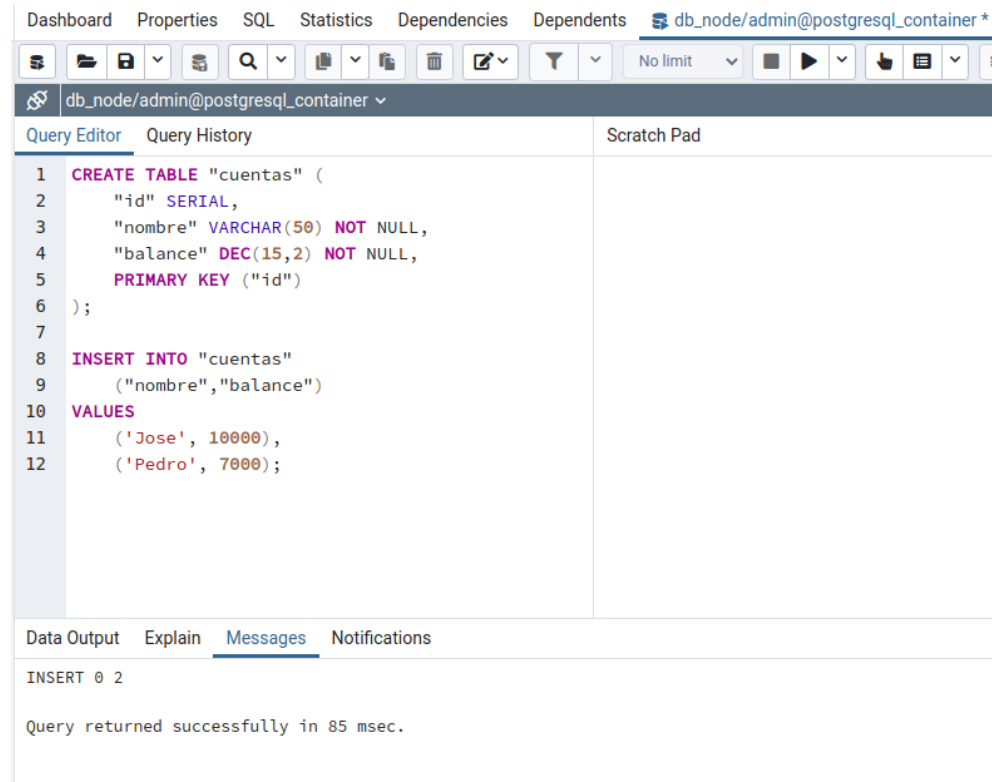
Abrimos pgAdmin, y allí crearemos la siguiente tabla:

```
1 CREATE TABLE "cuentas" (  
2     "id" SERIAL,  
3     "nombre" VARCHAR(50) NOT NULL,  
4     "balance" DEC(15,2) NOT NULL,  
5     PRIMARY KEY ("id")  
6 );
```

Insertamos los siguientes datos de pruebas:

```
1 INSERT INTO "cuentas" ("nombre","balance")  
2 VALUES  
3     ('Jose', 10000),  
4     ('Pedro', 7000);
```

En pgAdmin procedemos a ejecutar el script, presionando el botón o con la tecla F5.



EXERCISE 2: REALIZAR MODIFICACIONES EN LA BASE DE DATOS SIN TRANSACCIONES

Realizaremos las modificaciones correspondientes de actualización de balance de cuentas de un cliente a otro. Para este ejemplo, vamos a transferir 1.000 de saldo del cliente con id 1 de nombre José, al cliente con id 2 de nombre Pedro; secuencialmente, las actualizaciones serían de la siguiente manera:

Movimiento 1:





```
1 UPDATE cuentas  
2 SET balance= balance - 1000  
3 WHERE id = 1;
```

Movimiento 2:

```
1 UPDATE cuentas
2 SET balance= balance + 1000
3 WHERE id = 2;
```

Si se realizan tanto el movimiento 1, como el movimiento 2, y no existen fallos, significa que todas las operaciones de transacción son ejecutadas por completo si se cumple con la propiedad de Atomicidad.

Ahora, supongamos que el movimiento 2 no se realiza. En este caso, tendremos una inconsistencia en la base de datos, ya que se descuenta del cliente con id 1, pero no se suma el balance al cliente con id 2, esto es:

Data Output	Explain	Messages	Notifications
	id [PK] integer 	nombre character varying (50) 	balance numeric (15,2) 
1	2	Pedro	7000.00
2	1	Jose	9000.00

Observamos que en el balance del cliente con id 1 se descontaron los 1.000, quedando en saldo 9.000; pero, al cliente con id 2 no se le sumó este monto para estar con un balance de 8.000, por lo que suponemos que existió algún fallo en la transacción, y por algún motivo no se ejecutó el movimiento 2.

También se puede presentar el caso de que se le acredite al cliente con id 2, pero no se descuenta al cliente con id 1. En este sentido, no se mantiene la consistencia de operaciones en la base de datos.

Data Output	Explain	Messages	Notifications												
<table border="1"> <thead> <tr> <th></th><th>id [PK] integer</th><th>nombre character varying (50)</th><th>balance numeric (15,2)</th></tr> </thead> <tbody> <tr> <td>1</td><td>1</td><td>Jose</td><td>10000.00</td></tr> <tr> <td>2</td><td>2</td><td>Pedro</td><td>8000.00</td></tr> </tbody> </table>		id [PK] integer	nombre character varying (50)	balance numeric (15,2)	1	1	Jose	10000.00	2	2	Pedro	8000.00			
	id [PK] integer	nombre character varying (50)	balance numeric (15,2)												
1	1	Jose	10000.00												
2	2	Pedro	8000.00												

EXERCISE 3: REALIZAR TRANSACCIONES CON BEGIN, COMMIT Y ROLLBACK

Para evitar la inconsistencia de la información en la base de datos, y cumplir con la propiedad de ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) para evitar fallos en los movimientos con fines de recuperación, el sistema necesita mantenerse al tanto cuando una transacción inicia, termina, y se confirma o aborta. El sistema de recuperación hace uso de las siguientes operaciones: BEGIN, COMMIT, y ROLLBACK.





En nuestra operación o movimiento de transferencia de un cliente con id 1, al cliente con id 2, observamos que se actualizan correctamente para aquel que inició la transacción:

```

1 BEGIN;
2
3 UPDATE cuentas
4 SET balance= balance - 1000
5 WHERE id = 1;
6 SELECT * FROM cuentas;
7
8
9 UPDATE cuentas
10 SET balance= balance + 1000
11 WHERE id = 2;
12 SELECT * FROM cuentas;
  
```

Data Output	Explain	Messages	Notifications												
<table border="1"> <thead> <tr> <th></th><th>id [PK] integer</th><th>nombre character varying (50)</th><th>balance numeric (15,2)</th></tr> </thead> <tbody> <tr> <td>1</td><td>1</td><td>Jose</td><td>9000.00</td></tr> <tr> <td>2</td><td>2</td><td>Pedro</td><td>8000.00</td></tr> </tbody> </table>		id [PK] integer	nombre character varying (50)	balance numeric (15,2)	1	1	Jose	9000.00	2	2	Pedro	8000.00			
	id [PK] integer	nombre character varying (50)	balance numeric (15,2)												
1	1	Jose	9000.00												
2	2	Pedro	8000.00												

Para otro cliente, si realiza un `SELECT * FROM cuentas`, no tiene actualizado el balance:

Data Output	Explain	Messages	Notifications
	id [PK] integer 	nombre character varying (50) 	balance numeric (15,2) 
1	1	Jose	10000.00
2	2	Pedro	7000.00

Esto es porque se ha iniciado una transacción con `BEGIN`, pero no ha terminado satisfactoriamente con el comando `COMMIT`. Se verá así:

```
1 BEGIN;
2
3 UPDATE cuentas
4 SET balance= balance - 1000
5 WHERE id = 1;
6
7
8 UPDATE cuentas
9 SET balance= balance + 1000
10 WHERE id = 2;
11 SELECT * FROM cuentas;
12
13 COMMIT;
```

Si consultamos con otro cliente, observaremos que la tabla está actualizada correctamente con los balances que le corresponden.

Para hacer uso del `ROLLBACK`, vamos a suponer que la transacción no terminó con éxito, y que cualquier cambio en la base de datos se debe cancelar y recuperar; es decir, reestablecer la base de datos al punto de integridad.

```
1 BEGIN;
2
3 UPDATE cuentas
4 SET balance= balance - 1000
5 WHERE id = 1;
6
```



```
7 UPDATE cuentas
8 SET balance= balance + 1000
9 WHERE id = 2;
10
11 SELECT * FROM cuentas;
12
13 -- Restablecemos los cambios
14 ROLLBACK;
15
16 SELECT * FROM cuentas;
```