

Práctica 07 transacciones

Semestre: Sexto semestre

Ciclo: Enero – Junio 2024

Nombre del alumno

Jorge Parra Hidalgo

I.- Propósito de la practica

Comprender y practicar la creación y el uso de transacciones en MySQL utilizando una base de datos de cuenta bancaria.

II.- Actividades.

1. Cree una tabla "cuentas" con los siguientes campos:
 - a. id (int, auto-increment, primary key)
 - b. nombre (varchar(50))
 - c. saldo (decimal(10,2))

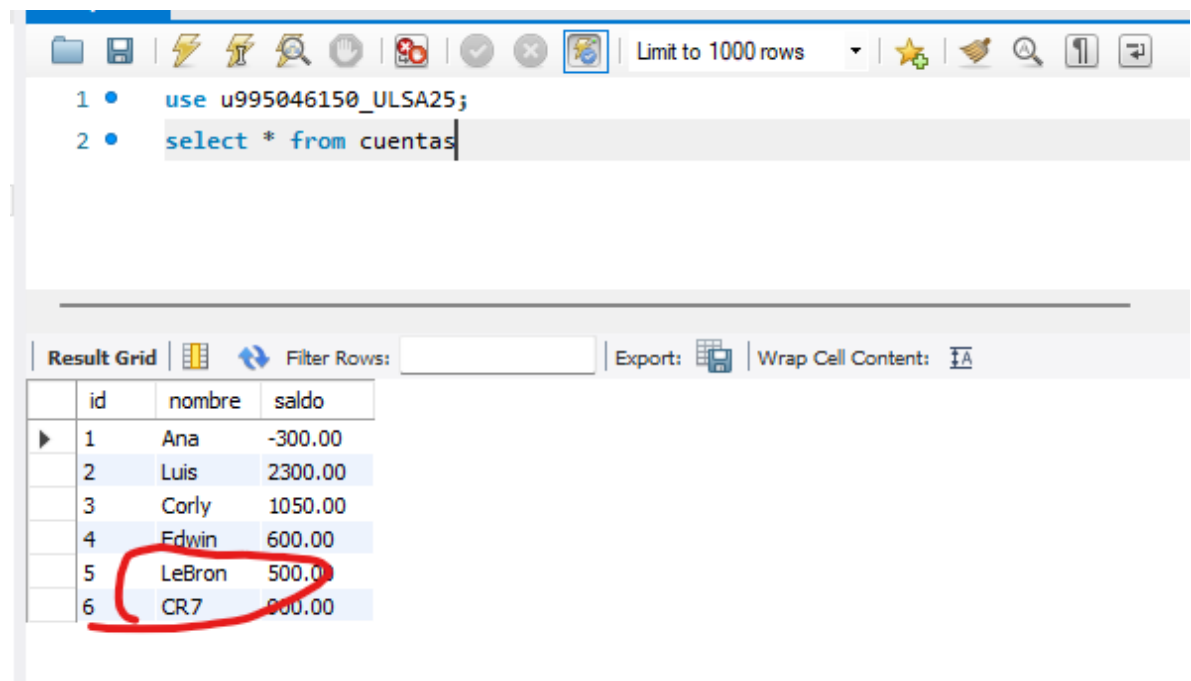
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view with 'u995046150_ULSA25' expanded, containing 'Tables', 'Columns', 'Indexes', 'Foreign Keys', 'Triggers', 'Views', 'Stored Procedures', and 'Functions'. The 'Tables' pane shows a table named 'cuentas'. The main area shows the 'Table Name' as 'cuentas', 'Schema' as 'u995046150_ULSA25', 'Charset/Collation' as 'utf8mb4', and 'Engine' as 'InnoDB'. The 'Column Name' table is as follows:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nombre	VARCHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
saldo	DECIMAL(10,2)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

2. Inserta algunos registros de ejemplo en la tabla "cuentas".

Práctica

Tecnologías de la Información y
Telecomunicaciones
Bases de datos II



The screenshot shows a SQL Server Enterprise Manager window. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The query editor contains two lines of SQL code:

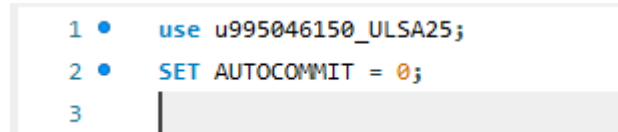
```
1 • use u995046150_ULSA25;  
2 • select * from cuentas
```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid displays a table with three columns: 'id', 'nombre', and 'saldo'. The data is as follows:

	id	nombre	saldo
▶	1	Ana	-300.00
	2	Luis	2300.00
	3	Corly	1050.00
	4	Edwin	600.00
	5	LeBron	500.00
	6	CR7	800.00

A red circle is drawn around the row with 'id' 5 and 'nombre' 'LeBron'.

3. Deshabilita el modo AUTOCOMMIT para iniciar una transacción manualmente: SET AUTOCOMMIT = 0;



The screenshot shows the SQL query editor with the following code:

```
1 • use u995046150_ULSA25;  
2 • SET AUTOCOMMIT = 0;  
3 •
```

4. Realiza una transferencia de saldo entre dos cuentas:
- Inicia la transacción: START TRANSACTION;
 - Resta el monto a transferir del saldo de la cuenta origen.
 - Suma el monto transferido al saldo de la cuenta destino.

Name: `jp_transferencia` The nam
DDL: `statement`

```

1 CREATE DEFINER='u995046150_root'@'%' PROCEDURE `jp_transferencia` (
2     IN cuenta_origen INT,
3     IN cuenta_destino INT,
4     IN monto DECIMAL(10,2)
5 )
6 BEGIN
7     START TRANSACTION;
8
9     UPDATE cuentas
10    SET saldo = saldo - monto
11    WHERE id = cuenta_origen;
12
13    UPDATE cuentas
14    SET saldo = saldo + monto
15    WHERE id = cuenta_destino;
16
17    COMMIT;
18 END
    
```

5. Verifica que los saldos se hayan actualizado correctamente:
 - a. `SELECT * FROM cuentas;`

(Hice un procedimiento almacenado para mayor facilidad para consultar *mis* registros que son el 5 y 6)

Result Grid | Filter Rows:

	id	nombre	saldo
▶	5	LeBron	350.00
	6	CR7	1050.00

	id	nombre	saldo
▶	5	LeBron	200.00
	6	CR7	1200.00

6. Confirma los cambios realizados: `COMMIT;`
Si se realizaron los cambios
7. Reinicia la secuencia iniciando otra transacción: `START TRANSACTION;`
 - a. Realiza alguna operación incorrecta, como restar un monto mayor al saldo disponible.
 - b. Verifica que el saldo no se actualizó correctamente.
 - c. Deshace los cambios realizados: `ROLLBACK;`

Name: `sp_transferencia_con_validacion` The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```

1 CREATE DEFINER='u995046150_root'@'%' PROCEDURE `sp_transferencia_con_validacion` (
2     IN cuenta_origen INT,
3     IN cuenta_destino INT,
4     IN monto DECIMAL(10,2)
5 )
6 BEGIN
7     DECLARE saldo_origen DECIMAL(10,2);
8     DECLARE saldo_destino DECIMAL(10,2);
9
10    -- Desactivar autocommit
11    SET AUTOCOMMIT = 0;
12
13    -- Llamar al SP jp_consultar_cuentas antes de la transacción (opcional)
14    CALL jp_consultar_cuentas();
15
16    -- Iniciar transacción
17    START TRANSACTION;
18
19    -- Transferir monto de la cuenta origen a la cuenta destino
20    UPDATE cuentas
21    SET saldo = saldo - monto
22    WHERE id = cuenta_origen;
23
24    UPDATE cuentas
25    SET saldo = saldo + monto
26    WHERE id = cuenta_destino;
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```

Name: `sp_transferencia_con_validacion` The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

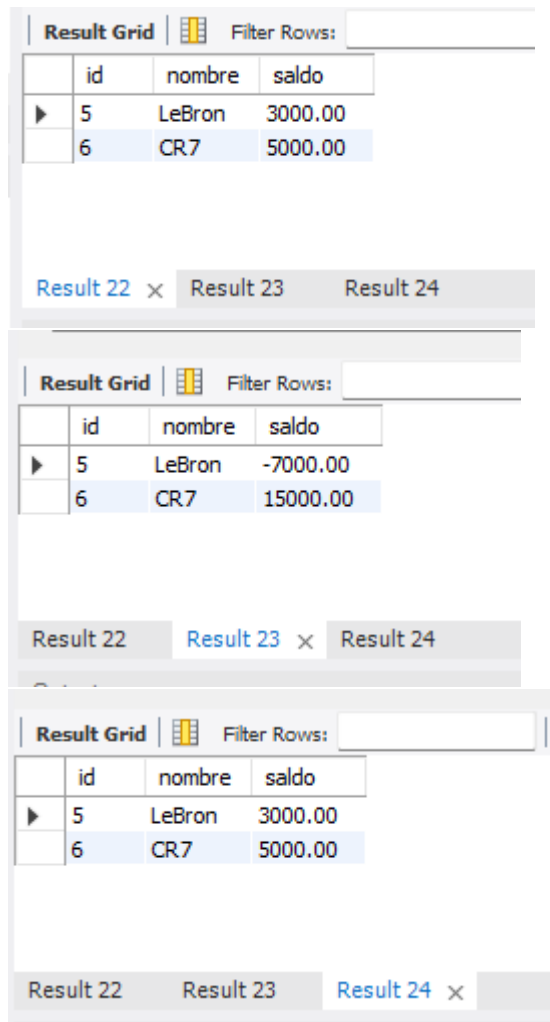
DDL:

```

21     SET saldo = saldo - monto
22     WHERE id = cuenta_origen;
23
24     UPDATE cuentas
25     SET saldo = saldo + monto
26     WHERE id = cuenta_destino;
27
28     CALL jp_consultar_cuentas();
29
30     -- Verificar saldos de ambas cuentas
31     SELECT saldo INTO saldo_origen FROM cuentas WHERE id = cuenta_origen;
32     SELECT saldo INTO saldo_destino FROM cuentas WHERE id = cuenta_destino;
33
34     -- Si alguna cuenta tiene saldo negativo, hacer ROLLBACK
35     IF saldo_origen < 0 OR saldo_destino < 0 THEN
36         ROLLBACK; -- Revertir la transacción si alguna cuenta tiene saldo negativo
37     ELSE
38         COMMIT; -- Confirmar la transacción si ambas cuentas tienen saldo positivo
39     END IF;
40
41     -- Llamar al SP jp_consultar_cuentas después de la transacción (opcional)
42     CALL jp_consultar_cuentas();
43
44     -- Habilitar nuevamente autocommit
45     SET AUTOCOMMIT = 1;
46 END

```

8. Verifica que los saldos volvieron a su estado original después del ROLLBACK.



The image displays three sequential screenshots of a database application's 'Result Grid' window, illustrating the effect of a ROLLBACK operation on account balances.

Top Screenshot: The 'Result Grid' shows the initial state with two rows:

	id	nombre	saldo
▶	5	LeBron	3000.00
	6	CR7	5000.00

The tabs below the grid are 'Result 22' (selected), 'Result 23', and 'Result 24'.

Middle Screenshot: The 'Result Grid' shows the state after a transaction. The balances have changed:

	id	nombre	saldo
▶	5	LeBron	-7000.00
	6	CR7	15000.00

The tabs below the grid are 'Result 22', 'Result 23' (selected), and 'Result 24'.

Bottom Screenshot: The 'Result Grid' shows the state after a ROLLBACK operation. The balances have returned to their original values:

	id	nombre	saldo
▶	5	LeBron	3000.00
	6	CR7	5000.00

The tabs below the grid are 'Result 22', 'Result 23', and 'Result 24' (selected).

9. Prueba a realizar transacciones anidadas (una transacción dentro de otra) y observa el comportamiento.

```
SET AUTOCOMMIT = 0;

START TRANSACTION;

-- Operación normal: Transferir dinero de la cuenta 5 a la cuenta 6
UPDATE cuentas SET saldo = saldo - 100 WHERE id = 5;

-- Guardar punto de control (SAVEPOINT) para poder revertir
SAVEPOINT sp1;

-- Operación errónea: Intentamos transferir dinero de la cuenta 5 por un monto muy alto (10000)
UPDATE cuentas SET saldo = saldo - 10000 WHERE id = 5;

-- Verificación del saldo de la cuenta 5
SELECT saldo INTO @saldo_cuenta_5 FROM cuentas WHERE id = 5;

-- Comprobamos si el saldo es suficiente. Si no lo es, reversion al SAVEPOINT
IF @saldo_cuenta_5 < 10000 THEN
    ROLLBACK TO sp1;
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Saldo insuficiente en la cuenta 5, operación revertida';
ELSE
    COMMIT;
END IF;

SET AUTOCOMMIT = 1;
```

10. Configura un tiempo de espera (lock wait timeout) corto y realiza operaciones que puedan causar interbloqueos (deadlocks). Observa cómo MySQL maneja estos casos.

```
SET SESSION lock_wait_timeout = 1; -- Establece el tiempo de espera para el bloqueo a 1 segundo solo para esta sesión

-- Transacción 1 (Cuenta 5 y luego consulta la cuenta 6)
START TRANSACTION;
UPDATE cuentas SET saldo = saldo - 100 WHERE id = 5; -- Resta 100 de la cuenta 5
SELECT * FROM cuentas WHERE id = 6; -- Consulta la cuenta 6

-- Transacción 2 (Cuenta 6 y luego consulta la cuenta 5)
START TRANSACTION;
UPDATE cuentas SET saldo = saldo - 100 WHERE id = 6; -- Resta 100 de la cuenta 6
SELECT * FROM cuentas WHERE id = 5; -- Consulta la cuenta 5
```

11. Experimenta con diferentes niveles de aislamiento (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE) y observa cómo afectan la visibilidad de los datos entre transacciones concurrentes.

Práctica

Tecnologías de la Información y
Telecomunicaciones
Bases de datos II

```
1 • use u995046150_ULSA25;  
2 • SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;  
3 • START TRANSACTION;  
4 • SELECT * FROM cuentas WHERE id in (5,6);  
5 • COMMIT;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
id	nombre	saldo			
5	LeBron	-19300.00			
6	CR7	7000.00			