

# TRANSACCIONES EN BASES DE DATOS DISTRIBUIDAS

---

Prof. Alejandro Reyes Ortiz

# Objetivos

- Identificar y explicar las primitivas de las transacciones
- Crear transacciones en Base de datos: MySQL
- Ejemplo transacciones en Base de datos: MySQL

# Primitivas para el manejo de transacciones

Las transacciones consisten de una secuencia de operaciones primitivas encerradas entre las palabras clave **Begin Transaction** y **End Transaction**. Por ejemplo:

**BeginTransaction** Reservación

...

**EndTransaction** {Reservación}

# Primitivas para el manejo de transacciones

- ABORT\_TRANSACTION (deshacer operación)
- READ (leer datos)
- WRITE (escribir datos)
- COMMIT (Consumación)

# Transacciones en MySQL

- MySQL tiene sus instrucciones para ejecutar un sistema de transacciones.
- El efecto de todas las sentencias SQL en una transacción es que queden exitosas (commit) o que todas regresen a su estado original (rollback).

# Transacciones en MySQL

- Por defecto, MySQL se ejecuta en modo autocommit.
- Esto significa que tan pronto como se ejecuta una sentencia se actualiza (modifica) la tabla, MySQL almacenará la actualización en disco.
- Se puede poner MySQL en modo no-autocommit con el comando siguiente:

```
SET AUTOCOMMIT=0;
```

# Transacciones en MySQL

- Iniciar una transacción

**BEGIN** y **BEGIN WORK** están disponibles desde MySQL 3.23.17 y 3.23.19, respectivamente.

**START TRANSACTION** fue añadido en MySQL 4.0.11;

# Sintaxis de una transacción en MySQL 5.5

START TRANSACTION;

← Inicio de la Transacción

SET AUTOCOMMIT = 0;

← No-autocommit

SELECT, INSERT, UPDATE o DELETE

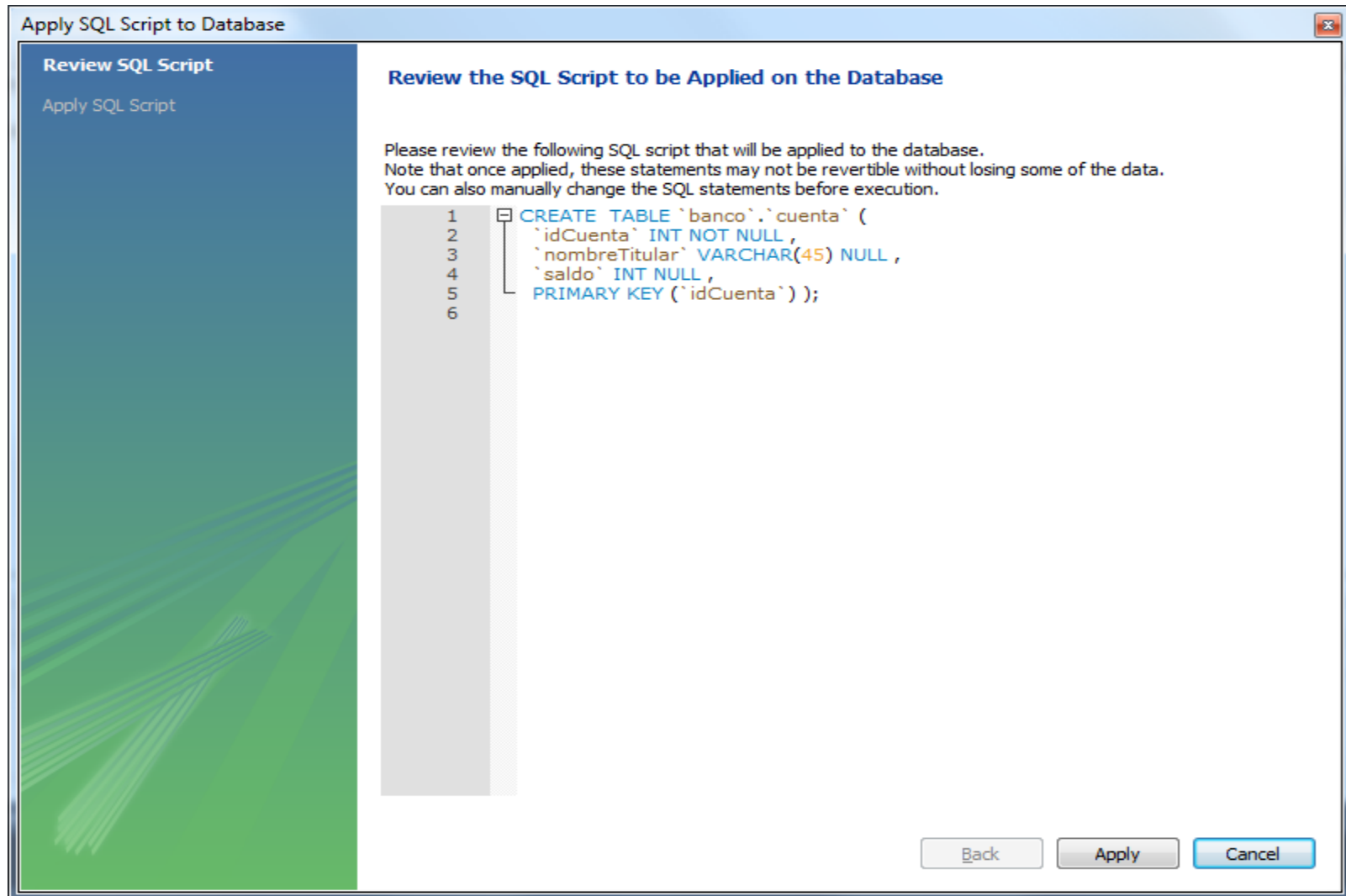
← Bloque de operaciones

COMMIT o ROLLBACK

← Terminación



# Crear una tabla cuenta



MySQL Workbench

SQL Editor (Local instance M... x) SQL Editor (Local instance MyS... x)

File Edit View Query Database Plugins Scripting Help

ORACLE

Object Browser

SCHEMAS

Search objects

- academia
- banco
  - Tables
    - cuentas
  - Views
  - Routines
- bdservlet
- escuela
- investigadores
- redsocial
- spanishwordnet
- test

Information

Table: **cuentas**

Columns:

- idCuenta int(11) PK
- nombreTitular varchar(45)
- saldo int(11)

SQL File 3\* Query 6 x

```
1 SELECT * FROM banco.cuentas;
```

Filter:

| idCuenta | nombreTitular      | saldo |
|----------|--------------------|-------|
| 2222     | Juan Pérez         | 4700  |
| 5555     | Ana Luisa Martínez | 6500  |
| 6666     | José Rodríguez     | 8900  |
| NULL     | NULL               | NULL  |

cuentas 1 x

Apply Cancel Snippets

Output

Action Output

| Time       | Action                                    | Message           | Duration / Fetch      |
|------------|---|-------------------|-----------------------|
| 4 10:31:51 | Apply changes to banco                    | Changes applied   |                       |
| 5 10:32:43 | Apply changes to cuentas                  | Changes applied   |                       |
| 6 10:32:58 | SELECT * FROM banco.cuentas LIMIT 0, 1000 | 0 row(s) returned | 0.000 sec / 0.000 sec |
| 7 10:34:20 | SELECT * FROM banco.cuentas LIMIT 0, 1000 | 3 row(s) returned | 0.000 sec / 0.000 sec |
| 8 10:36:58 | SELECT * FROM banco.cuentas LIMIT 0, 1000 | 3 row(s) returned | 0.000 sec / 0.000 sec |

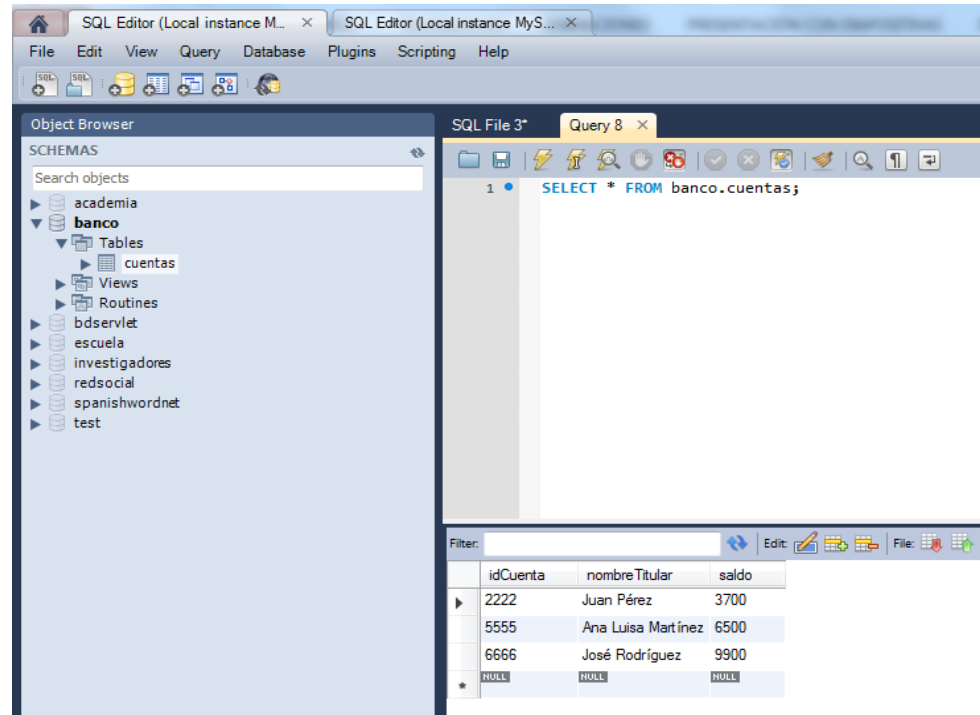
Object Info Session

Query Completed

# Transacciones en MySQL: almacenamiento temporal

```
START TRANSACTION;  
SET AUTOCOMMIT = 0;  
UPDATE cuentas SET saldo=saldo-1000 WHERE idCuenta=2222;  
UPDATE cuentas SET saldo=saldo+1000 WHERE idCuenta=6666;
```

Existe el registro en la tabla de manera temporal pero no en el almacenamiento permanente.



The screenshot shows the MySQL SQL Editor interface. The 'Object Browser' on the left displays a tree view of schemas, with 'banco' expanded to show tables, views, and routines. The 'SQL Editor' window on the right contains a query: `SELECT * FROM banco.cuentas;`. Below the query, a result set is displayed in a table format.

| idCuenta | nombreTitular      | saldo |
|----------|--------------------|-------|
| 2222     | Juan Pérez         | 3700  |
| 5555     | Ana Luisa Martínez | 6500  |
| 6666     | José Rodríguez     | 9900  |
| NULL     | NULL               | NULL  |

# Transacciones en MySQL: RollBack

Almacenamiento temporal  
(misma instancia)

Almacenamiento permanente  
(nueva instancia)

The screenshot shows the MySQL Workbench interface. The Object Browser on the left displays the 'banco' schema with a 'cuentas' table. The SQL Editor window shows the query: `SELECT * FROM banco.cuentas;`. The results pane at the bottom displays the following data:

| idCuenta | nombreTitular      | saldo |
|----------|--------------------|-------|
| 2222     | Juan Pérez         | 3700  |
| 5555     | Ana Luisa Martínez | 6500  |
| 6666     | José Rodríguez     | 9900  |
| NULL     | NULL               | NULL  |

The screenshot shows the MySQL Workbench interface. The Object Browser on the left displays the 'banco' schema with a 'cuentas' table. The SQL Editor window shows the query: `SELECT * FROM banco.cuentas;`. The results pane at the bottom displays the following data:

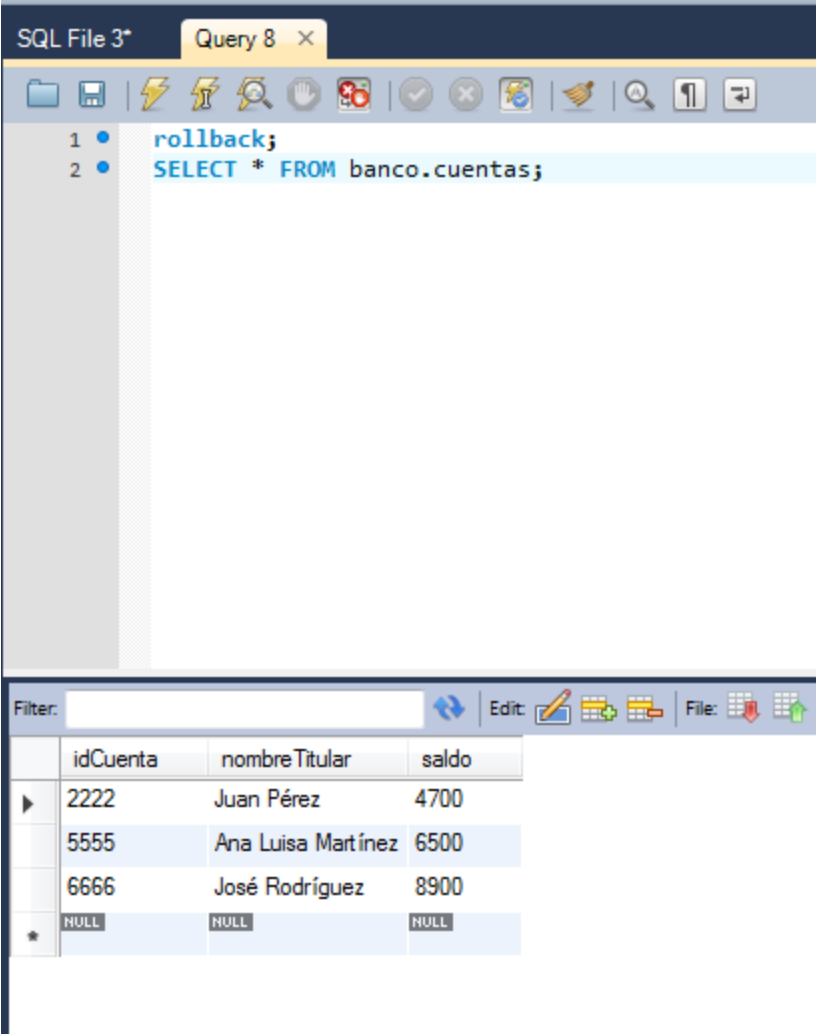
| idCuenta | nombreTitular      | saldo |
|----------|--------------------|-------|
| 2222     | Juan Pérez         | 4700  |
| 5555     | Ana Luisa Martínez | 6500  |
| 6666     | José Rodríguez     | 8900  |
| NULL     | NULL               | NULL  |

# Transacciones en MySQL: RollBack

ROLLBACK;

SELECT \* FROM cuentas;

La transacción se termina ROLLBACK.



The screenshot shows a MySQL query editor window titled "Query 8". The query text is as follows:

```
1 rollback;  
2 SELECT * FROM banco.cuentas;
```

Below the query editor, a table view displays the results of the query. The table has three columns: idCuenta, nombreTitular, and saldo. The data is as follows:

|   | idCuenta | nombreTitular      | saldo |
|---|----------|--------------------|-------|
| ▶ | 2222     | Juan Pérez         | 4700  |
|   | 5555     | Ana Luisa Martínez | 6500  |
|   | 6666     | José Rodríguez     | 8900  |
| * | NULL     | NULL               | NULL  |

# Transacciones en MySQL: Exitosa

```
START TRANSACTION;  
  SET AUTOCOMMIT = 0;  
  UPDATE cuentas SET saldo=saldo-1000 WHERE  
idCuenta=2222;  
  UPDATE cuentas SET saldo=saldo+1000 WHERE  
idCuenta=6666;
```

Almacenamiento temporal  
(misma instancia)

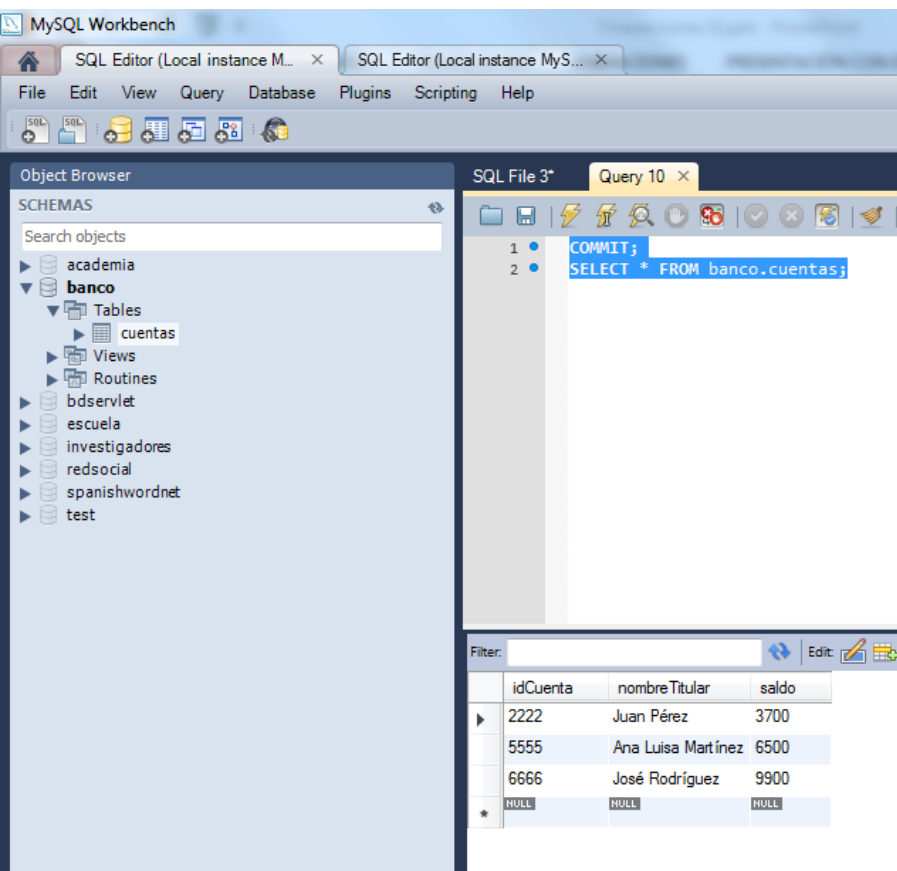
```
COMMIT;  
SELECT * FROM banco.cuentas;
```

Se verá reflejada en almacenamiento  
permanente

La transacción se termina con la instrucción  
COMMIT.

# Después de COMMIT

- Se verá reflejada la actualización en cualquier instancia



MySQL Workbench

SQL Editor (Local instance M... x) SQL Editor (Local instance MyS... x)

File Edit View Query Database Plugins Scripting Help

Object Browser

SCHEMAS

Search objects

academia

**banco**

Tables

cuentas

Views

Routines

bdservlet

escuela

investigadores

redsocial

spanishwordnet

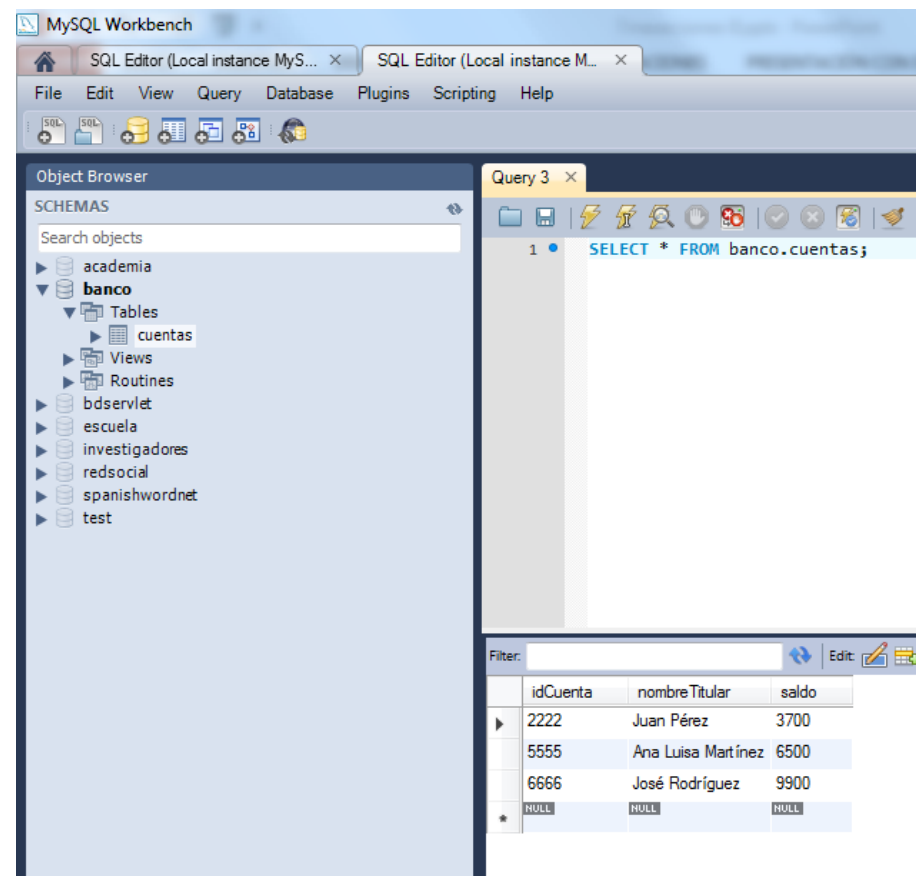
test

SQL File 3\* Query 10 x

```
1 • COMMIT;  
2 • SELECT * FROM banco.cuentas;
```

Filter:

| idCuenta | nombre Titular     | saldo |
|----------|--------------------|-------|
| 2222     | Juan Pérez         | 3700  |
| 5555     | Ana Luisa Martínez | 6500  |
| 6666     | José Rodríguez     | 9900  |
| NULL     | NULL               | NULL  |



MySQL Workbench

SQL Editor (Local instance MyS... x) SQL Editor (Local instance M... x)

File Edit View Query Database Plugins Scripting Help

Object Browser

SCHEMAS

Search objects

academia

**banco**

Tables

cuentas

Views

Routines

bdservlet

escuela

investigadores

redsocial

spanishwordnet

test

Query 3 x

```
1 • SELECT * FROM banco.cuentas;
```

Filter:

| idCuenta | nombre Titular     | saldo |
|----------|--------------------|-------|
| 2222     | Juan Pérez         | 3700  |
| 5555     | Ana Luisa Martínez | 6500  |
| 6666     | José Rodríguez     | 9900  |
| NULL     | NULL               | NULL  |

# Después de COMMIT

- Intentar hacer *rollback* y verificar qué sucede.