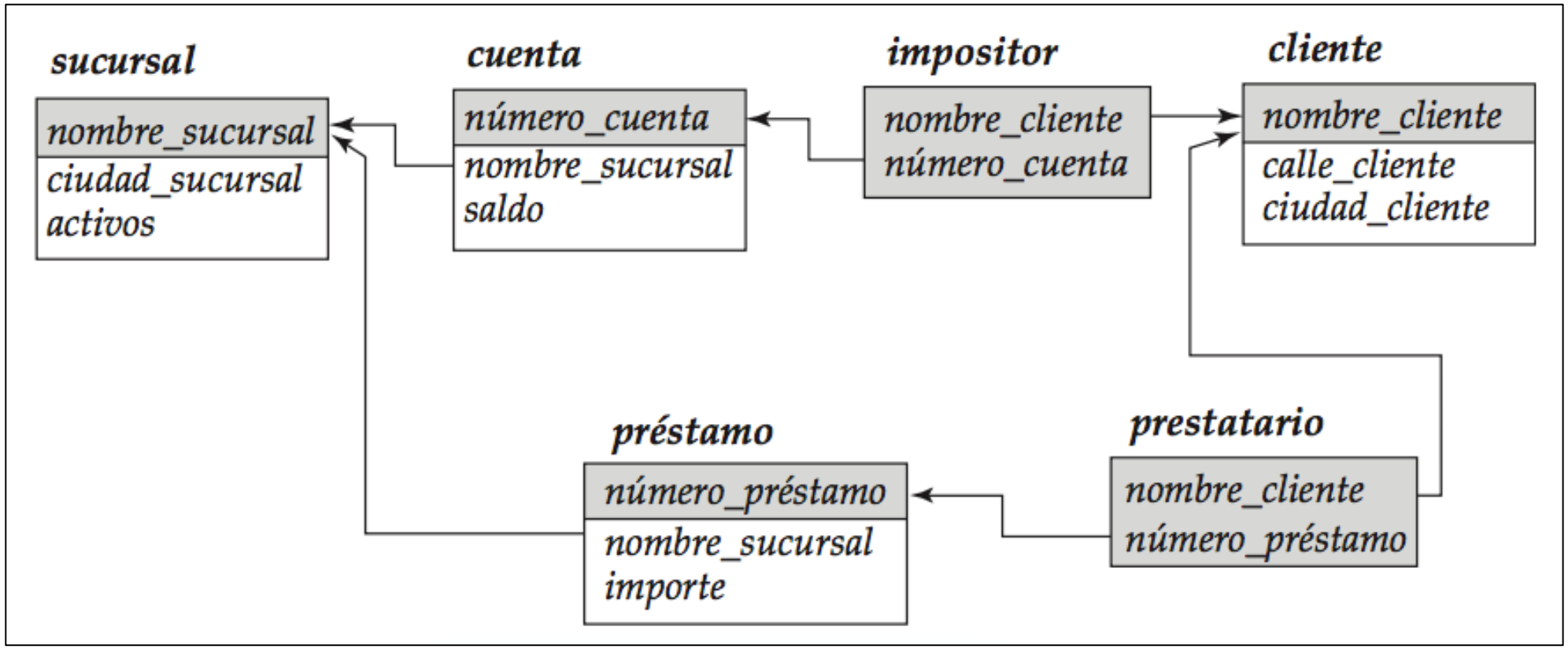
Bases de Datos

- LENGUAJE DE CONSULTAS (SQL)

Lenguaje de Consultas (SQL)

- Procedimientos almacenados
- Disparadores

Base de datos ejemplo. Banco



Lenguaje de Consultas (SQL)

❖ Procedimientos almacenados

- Los constructores procedimentales de SQL permiten que se registre la “lógica de negocio” en forma de procedimientos almacenados de la base de datos y se ejecute en la propia base de datos.
- La parte de la norma SQL que trata de estos constructores se denomina módulo de almacenamiento persistente (Persistent Storage Module, PSM).
- Los constructores procedimentales son necesarios para permitir que se codifiquen como procedimientos almacenados las reglas de negocio complejas.

Lenguaje de Consultas (SQL)

❖ Procedimientos almacenados

- Por ejemplo, los bancos suelen tener muchas reglas sobre la manera y el momento en que se pueden hacer los pagos a los clientes como
- Los límites máximos de retirada de efectivo
- Las exigencias de saldo mínimo
- Las facilidades que permiten a los clientes retirar más dinero que el saldo disponible mediante la concesión automática de un préstamo,
- Etcétera

Lenguaje de Consultas (SQL)

Procedimientos almacenados

Aunque esa lógica de negocio puede codificarse en forma de procedimientos de los lenguajes de programación almacenados completamente fuera de las bases de datos, su definición como procedimientos almacenados de la base de datos presenta varias ventajas.

Permite que varias aplicaciones tengan acceso a los procedimientos

Solo se requiere modificar un solo punto en caso de cambio de las reglas de negocio, sin que haga falta modificar la aplicación.

El código de la aplicación puede llamar a los procedimientos almacenados, en lugar de actualizar directamente las relaciones de la base de datos.

Lenguaje de Consultas (SQL)

Procedimientos almacenados

- Las instrucciones compuestas son de la forma begin…end, y pueden contener varias instrucciones de SQL entre begin y end.
- Soporta las instrucciones while y repeat con la sintaxis siguiente:

```
declare n integer default 0;
while n < 10 do
    set n = n + 1;
end while
repeat
    set n = n − 1;
until n = 0
end repeat
```

Lenguaje de Consultas (SQL)

❖ Procedimientos almacenados

- Asimismo, se tiene el bucle **for** que permite la iteración por todos los resultados de una consulta:

```
declare n integer default 0;
for r as
        select saldo from cuenta
        where nombre_sucursal = 'Navacerrada'
do
    set n = n+ r.saldo
end for
```

Lenguaje de Consultas (SQL)

❖ Procedimientos almacenados

- Entre las instrucciones condicionales se encuentra **if-then-else** con la sintaxis:

if $r.saldo < 1000$
  then set $l = l + r.saldo$
elseif $r.saldo < 5000$
  then set $m = m + r.saldo$
else set $h = h + r.saldo$
end if

Lenguaje de Consultas (SQL)

Procedimientos almacenados

- Crear un procedimiento almacenado en SQL:

```
CREATE PROCEDURE nombre_proc (parámetros)
BEGIN
        Lista de sentencias
END

Parámetros
[IN|OUT |INOUT ] nombre_param type
```

- Llamada a un procedimiento almacenado en SQL:

```
CALL nombre_proc (parámetros)
```

Lenguaje de Consultas (SQL)

❖ Procedimientos almacenados

- Crear un procedimiento almacenado en SQL:

```
Parámetros
[OUT |IN|INOUT ] nombre_param type
```

- Un parámetro **OUT** pasa un valor del procedimiento a la persona que llama. Su valor inicial es NULL dentro del procedimiento, y su valor es visible para la persona que llama cuando el procedimiento vuelve.
- Un parámetro **IN** pasa un valor a un procedimiento. El procedimiento puede modificar el valor, pero la modificación no es visible para la persona que llama cuando el procedimiento vuelve.

Lenguaje de Consultas (SQL)

❖ Procedimientos almacenados

- Crear un procedimiento almacenado en SQL:

```
Parámetros
[IN|OUT |INOUT ] nombre_param type
```

- Un parámetro **INOUT** es inicializado por la persona que llama, puede ser modificado por el procedimiento, y cualquier cambio realizado por el procedimiento es visible para la persona que llama cuando el procedimiento regresa.

Lenguaje de Consultas (SQL)

- **Procedimientos almacenados**

```
MariaDB [(none)]> use banco;
Database changed
MariaDB [banco]> delimiter //
MariaDB [banco]> CREATE PROCEDURE simpleproc (OUT param1 INT)
    -> BEGIN
    -> SELECT COUNT(*) INTO param1 FROM banco.cuenta;
    -> END
    -> //
```

Lenguaje de Consultas (SQL)

Procedimientos almacenados

- L

```
MariaDB [banco]> delimiter ;
MariaDB [banco]> CALL simpleproc(@a);
Query OK, 1 row affected (0.03 sec)

MariaDB [banco]> SELECT @a;
+------+
| @a   |
+------+
|    7 |
+------+
1 row in set (0.00 sec)

MariaDB [banco]> select * from cuenta;
+---------------+-----------------+-------+
| numero_cuenta | nombre_sucursal | saldo |
+---------------+-----------------+-------+
| C-101         | Centro          |   950 |
| C-102         | Navacerrada     |   400 |
| C-201         | Galapagar       |   900 |
| C-215         | Becerril        |   700 |
| C-217         | Galapagar       |   750 |
| C-222         | Moralzarzal     |   700 |
| C-305         | Collado Mediano |   350 |
+---------------+-----------------+-------+
7 rows in set (0.00 sec)
```

Ejemplo parámetro IN

```
MariaDB [(none)]> use banco;
Database changed
MariaDB [banco]> DELIMITER //
MariaDB [banco]> CREATE PROCEDURE obtenerPromedioCuentaPorSucursal (IN nombreSucursal VARCHAR (255))
    -> BEGIN
    -> SELECT AVG(saldo)
    -> FROM cuenta
    -> WHERE nombre_sucursal=nombreSucursal;
    -> END//
Query OK, 0 rows affected (0.09 sec)

MariaDB [banco]> DELIMITER ;
MariaDB [banco]> CALL obtenerPromedioCuentaPorSucursal('Galapagar');
+------------+
| AVG(saldo) |
+------------+
|   825.0000 |
+------------+
1 row in set (0.05 sec)

Query OK, 0 rows affected (0.06 sec)
```

Ejemplo parámetro INOUT

```
MariaDB [banco]> CREATE PROCEDURE sumaPrestamo (IN nprestamo VARCHAR(10),INOUT cantidad INT(10))
    -> BEGIN
    -> SELECT importe+cantidad into cantidad
    -> FROM prestamo
    -> WHERE numero_prestamo=nprestamo;
    -> END//
Query OK, 0 rows affected (0.01 sec)
```

Ejemplo parámetro INOUT

```
MariaDB [banco]> SET @suma=0;
Query OK, 0 rows affected (0.00 sec)

MariaDB [banco]> CALL sumaPrestamo('P-23',@suma);
Query OK, 1 row affected (0.00 sec)

MariaDB [banco]> select @suma;
+-------+
| @suma |
+-------+
|  2000 |
+-------+
1 row in set (0.00 sec)

MariaDB [banco]> CALL sumaPrestamo('P-93',@suma);
Query OK, 1 row affected (0.00 sec)

MariaDB [banco]> select @suma;
+-------+
| @suma |
+-------+
|  2500 |
+-------+
1 row in set (0.00 sec)
```

```
MariaDB [banco]> select * from prestamo;
+-----------------+-----------------+---------+
| numero_prestamo | nombre_sucursal | importe |
+-----------------+-----------------+---------+
| P-11            | Collado Mediano |     909 |
| P-14            | Centro          |    1500 |
| P-15            | Navacerrada     |    1500 |
| P-16            | Navacerrada     |    1300 |
| P-17            | Centro          |    1000 |
| P-23            | Moralzarzal     |    2000 |
| P-93            | Becerril        |     500 |
+-----------------+-----------------+---------+
7 rows in set (0.00 sec)
```

Lenguaje de Consultas (SQL)

Procedimientos almacenados

- L

Lenguaje de Consultas (SQL)

Disparadores

- Un disparador define una acción que la base de datos debe llevar a cabo cuando se produce algún suceso relacionado con la misma.
- Para diseñar un mecanismo disparador es necesario:

Lenguaje de Consultas (SQL)

Disparadores

evento-condición-acción

Especificar las condiciones en las que se va a ejecutar el disparador.

Esto se descompone en un **evento** que provoca la comprobación del disparador y una **condición** que se debe cumplir para que se ejecute el disparador.

Especificar las acciones que se van a realizar cuando se ejecute el disparador.

Lenguaje de Consultas (SQL)

Disparadores

- La base de datos almacena los disparadores como si fuesen datos normales, por lo que son persistentes y están accesibles para todas las operaciones de la base de datos.
- Una vez que se introduce un disparador en la base de datos, el sistema de bases de datos asume la responsabilidad de ejecutarlo cada vez que se produzca el evento especificado y se satisfaga la condición correspondiente.

Lenguaje de Consultas (SQL)

Disparadores

- **¿Cuándo son necesarios?**
- Son mecanismos útiles para alertar a los usuarios o para iniciar de manera automática ciertas tareas cuando se cumplen determinadas condiciones.
- Ejemplo
- Supongamos que en lugar de permitir saldos de cuenta negativos, el banco trata los descubiertos dejando a cero el saldo de la cuenta y creando un préstamo por el importe del descubierto.
- El banco da a este préstamo un número de préstamo idéntico al número de cuenta que ha tenido el descubierto.

Lenguaje de Consultas (SQL)

❖ Disparadores

- En este ejemplo la condición para ejecutar el disparador es una actualización de la relación cuenta que da lugar a un valor negativo de saldo.
- Supóngase que Santos retiró cierta cantidad de dinero de una cuenta que dio lugar a que el saldo de la cuenta fuera negativo y que *t* denota la tupla de cuenta con un valor negativo de saldo.
- Las acciones que hay que emprender son:

Lenguaje de Consultas (SQL)

❖ Disparadores

1. Insertar una nueva tupla ***S*** en la relación ***préstamo*** con:

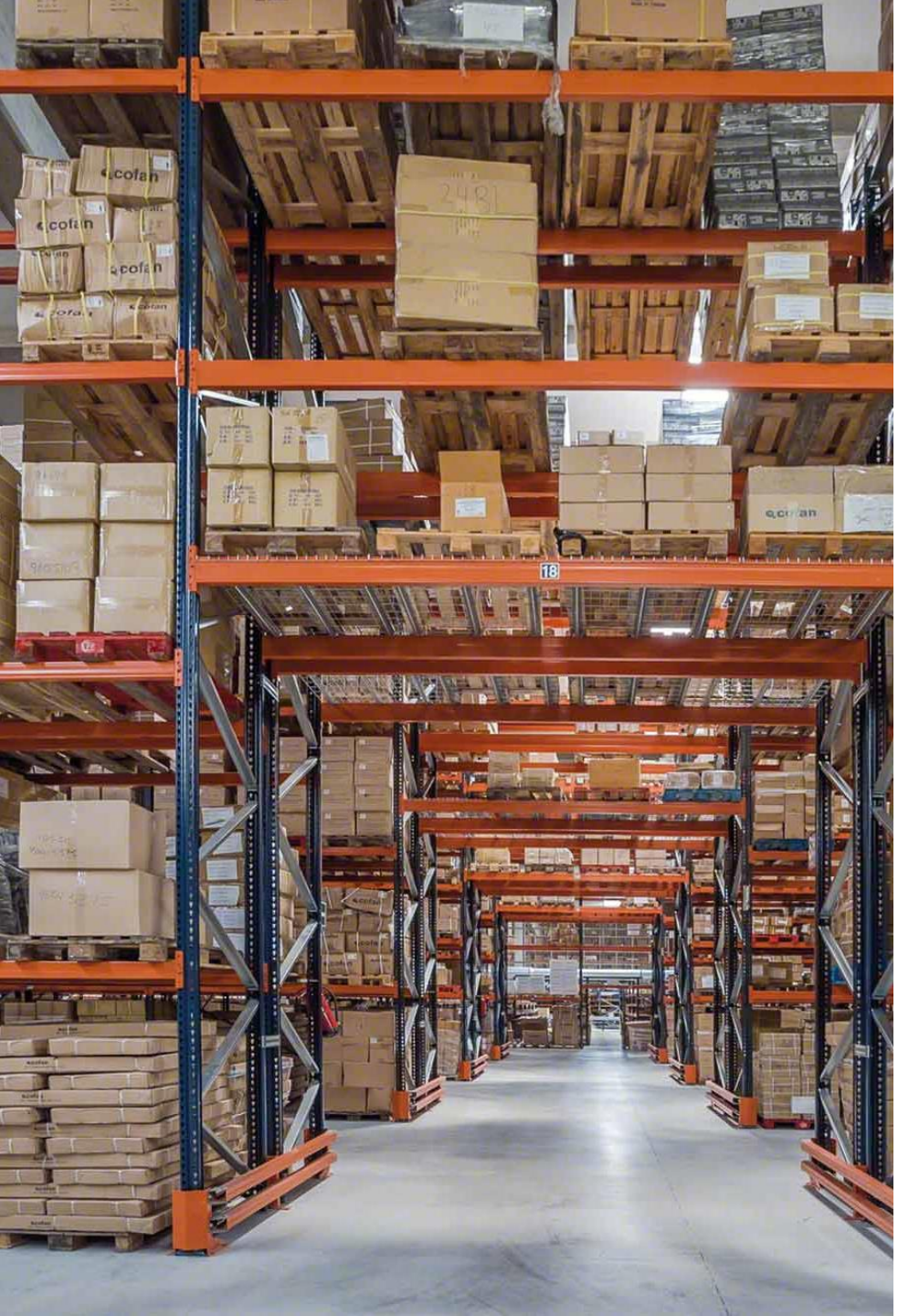
$$s[número\_préstamo] = t[número\_cuenta]$$
$$s[nombre\_sucursal] = t[nombre\_sucursal]$$
$$s[importe] = -t[saldo]$$

(hay que cambiar el signo de t[saldo] para obtener el importe del préstamo—un número positivo).

2. Insertar una nueva tupla *u* en la relación ***prestatario*** con:

$$u[nombre\_cliente] = \text{“Santos”}$$
$$u[número\_préstamo] = t[número\_cuenta]$$

3. Dejar *t*[*saldo*] a 0.



Lenguaje de Consultas (SQL)

❖ **Disparadores**

- Otro ejemplo:
- Un almacén que desee mantener unas existencias mínimas de cada producto; cuando el nivel de existencias de un producto cae por debajo del nivel mínimo, se debe realizar un pedido de manera automática.
- Ésta es la manera en que esta regla empresarial se puede implementar mediante disparadores:
- Al actualizar el nivel de existencias de cada producto el disparador debe comparar ese nivel con el nivel de existencias mínimo del producto y, si se halla en el mínimo o por debajo de él, se añade un nuevo pedido a la relación pedidos.

Lenguaje de Consultas (SQL)

❖ Disparadores en SQL

- Se volvieron parte de la norma en SQL:1999. Cada sistema de bases de datos implementó su propia sintaxis para los disparadores.
- Para crear un disparador se emplea la instrucción **create trigger nombre_disparador**.
- Los disparadores se pueden habilitar y deshabilitar. De manera predeterminada, al crearlos se habilitan, pero se pueden deshabilitar empleando **alter trigger nombre_disparador disable** o **disable trigger nombre_disparador**.
- Los disparadores deshabilitados se pueden volver a habilitar.
- Los disparadores también se pueden descartar, lo que los elimina de manera permanente, usando la instrucción **drop trigger nombre_disparador**.

Disparadores en SQL

Disparador para cuentas menores a 0

```
create trigger disparador_descubierto after update on cuenta
referencing new row as fila_nueva
for each row
when fila_nueva.saldo < 0
begin atomic
   insert into prestatario
          (select nombre_cliente, número_cuenta
           from impositor
           where fila_nueva.número_cuenta = impositor.número_cuenta);
   insert into préstamo values
          (fila_nueva.número_cuenta, fila_nueva.nombre_sucursal, – fila_nueva.saldo);
   update cuenta set saldo = 0
          where cuenta.número_cuenta = fila_nueva.número_cuenta
end
```

El disparador se inicia después de la ejecución de cualquier actualización de la relación cuenta.
La cláusula **referencing new row as** crea la variable fila_nueva (denominada transition variable), que almacena el valor de la fila actualizada después de la actualización.
La cláusula **for each row** se itera luego de manera explícita para cada fila actualizada.
La instrucción **when** especifica una condición, en este caso fila_nueva.saldo < 0. El sistema ejecuta el resto del cuerpo del disparador sólo para las tuplas que satisfacen esa condición.
La cláusula **begin atomic**…**end** sirve para reunir varias instrucciones de SQL en una sola instrucción compuesta.
Las dos instrucciones **insert** de la estructura begin..end realizan las tareas específicas de creación de las nuevas tuplas de las relaciones prestatario y préstamo para que representen el nuevo préstamo.
La instrucción **update** sirve para volver a dejar a cero el saldo de la cuenta tras su valor negativo anterior.

Disparadores en SQL

Disparador inventario igual o por debajo del mínimo

```
create trigger disparador_nuevo_pedido after update of nivel on existencias
referencing old row as fila_vieja, new row as fila_nueva
for each row
when fila_nueva.nivel <= (select nivel
                          from nivel_mínimo
                          where nivel_minimo.producto = fila_vieja.producto)
and fila_vieja.nivel > (select nivel
                        from nivel_mínimo
                        where nivel_mínimo.producto = fila_vieja.producto)
begin
        insert into pedidos
                (select producto, cantidad
                 from nuevo_pedido
                 where nuevo_pedido.producto = fila_vieja.producto)
end
```

Supongamos que se tienen las siguientes relaciones:
existencias*(producto, nivel).*
nivel_mínimo*(producto, nivel)*, que denota la cantidad mínima que se debe mantener de cada producto.
nuevo_pedido*(producto, cantidad)*, que denota la cantidad de producto que se debe pedir cuando su nivel cae por debajo del mínimo.
pedidos*(producto, cantidad)*, que denota la cantidad de producto que se debe pedir.
La cláusula **referencing old row as** se puede utilizar para crear una variable que almacene el valor anterior de una fila actualizada o borrada.
La cláusula **referencing new row as** se puede usar con las inserciones además de con las actualizaciones.

Lenguaje de Consultas (SQL)

Disparadores en SQL

- Agregar una nueva columna a la tabla Prestamo

```
ALTER TABLE prestamo ADD interes INT(10) NOT NULL AFT
ER importe;
```

Lenguaje de Consultas (SQL)

Disparadores en SQL

```
MariaDB [banco]> CREATE TRIGGER actualizaInteresPrestamo
    -> BEFORE UPDATE ON prestamo
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.importe <> OLD.importe
    -> THEN
    -> SET NEW.interes = NEW.importe * 0.10;
    -> END IF;
    -> END//
Query OK, 0 rows affected (0.04 sec)

MariaDB [banco]> delimiter ;
```

Lenguaje de Consultas (SQL)

Disparadores

```
MariaDB [banco]> SELECT * FROM prestamo;
+-----------------+-----------------+---------+---------+
| numero_prestamo | nombre_sucursal | importe | interes |
+-----------------+-----------------+---------+---------+
| P-11            | Collado Mediano |     909 |       0 |
| P-14            | Centro          |    1500 |       0 |
| P-15            | Navacerrada     |    1500 |       0 |
| P-16            | Navacerrada     |    1300 |       0 |
| P-17            | Centro          |    1000 |       0 |
| P-23            | Moralzarzal     |    2000 |       0 |
| P-93            | Becerril        |     500 |       0 |
+-----------------+-----------------+---------+---------+
7 rows in set (0.00 sec)

MariaDB [banco]> UPDATE prestamo SET importe = 1000 WHERE numero_prestamo='P-11';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [banco]> SELECT * FROM prestamo;
+-----------------+-----------------+---------+---------+
| numero_prestamo | nombre_sucursal | importe | interes |
+-----------------+-----------------+---------+---------+
| P-11            | Collado Mediano |    1000 |     100 |
| P-14            | Centro          |    1500 |       0 |
| P-15            | Navacerrada     |    1500 |       0 |
| P-16            | Navacerrada     |    1300 |       0 |
| P-17            | Centro          |    1000 |       0 |
| P-23            | Moralzarzal     |    2000 |       0 |
| P-93            | Becerril        |     500 |       0 |
+-----------------+-----------------+---------+---------+
7 rows in set (0.00 sec)
```

Lenguaje de Consultas (SQL) Disparadores

```
MariaDB [banco]> UPDATE prestamo SET importe = 1800 WHERE numero_prestamo='P-14';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [banco]> SELECT * FROM prestamo;
+-----------------+-----------------+---------+----------+
| numero_prestamo | nombre_sucursal | importe | interes |
+-----------------+-----------------+---------+----------+
| P-11            | Collado Mediano |    1000 |      100 |
| P-14            | Centro          |    1800 |      180 |
| P-15            | Navacerrada     |    1500 |        0 |
| P-16            | Navacerrada     |    1300 |        0 |
| P-17            | Centro          |    1000 |        0 |
| P-23            | Moralzarzal     |    2000 |        0 |
| P-93            | Becerril        |     500 |        0 |
+-----------------+-----------------+---------+----------+
7 rows in set (0.00 sec)
```

Lenguaje de Consultas (SQL)

Gracias por su atención