Integridad referencial en MySQL

MySQL 4.0 soporta cinco tipos de tablas: MyISAM, ISAM, HEAP, BDB (Base de datos Berkeley), e InnoDB. BDB e InnoDB son ambas tipos de tablas transaccionales. Además de poder trabajar con transacciones en MySQL, las tablas del tipo InnoDB también tienen soporte para la definición de claves foráneas, por lo que se nos permite definir reglas o restricciones que garanticen la integridad referencial de los registros.

A partir de la versión 4.0, MySQL ha agregado InnoDB a la lista de tipos de tablas soportados en una instalación típica. En este artículo se asume que se cuenta ya con un servidor MySQL con soporte para el tipo de tablas InnoDB. En nuestro caso haremos uso de un servidor MySQL 4.013 ejecutándose en un sistema MSWindows.

Nota: para asegurarnos que tenemos soporte para el tipo de tablas InnoDB podemos ejecutar la siguiente sentencia:

```
mysql> SHOW VARIABLES LIKE '%innodb%';
+---------------------------------+------------------------+
| Variable_name                   | Value                  |
+---------------------------------+------------------------+
| have_innodb                     | YES                    |
| innodb_additional_mem_pool_size | 1048576                |
| innodb_buffer_pool_size         | 8388608                |
| innodb_data_file_path           | ibdata1:10M:autoextend |
| innodb_data_home_dir            |                        |
| innodb_file_io_threads          | 4                      |
| innodb_force_recovery           | 0                      |
| innodb_thread_concurrency       | 8                      |
| innodb_flush_log_at_trx_commit  | 1                      |
| innodb_fast_shutdown            | ON                     |
| innodb_flush_method             |                        |
| innodb_lock_wait_timeout        | 50                     |
| innodb_log_arch_dir             | .                      |
| innodb_log_archive              | OFF                    |
| innodb_log_buffer_size          | 1048576                |
| innodb_log_file_size            | 5242880                |
| innodb_log_files_in_group       | 2                      |
| innodb_log_group_home_dir       | .                      |
| innodb_mirrored_log_groups      | 1                      |
| innodb_max_dirty_pages_pct      | 90                     |
+---------------------------------+------------------------+
20 rows in set (0.00 sec)
```

La variable más importante es por supuesto `have_innodb` que tiene el valor YES.

Claves primarias

Para entender lo que son las claves foráneas, tal vez sea necesario entender primero lo que son las claves primarias.

Es un hecho que las claves juegan un papel muy importante no sólo en MySQL, sino en cualquier base de datos relacional. De manera simple, las claves proporcionan una manera rápida y eficiente de buscar datos en una tabla, además de que permiten preservar la integridad de los datos.

Una **clave candidata** es un campo, o una combinación de campos, que identifican de manera única un registro de una tabla. Éstas no pueden contener valores nulos, y su valor debe ser único.

Una **clave primaria** es una clave candidata que ha sido diseñada para identificar de manera única a los registros de una tabla a través de toda la estructura de la base de datos.

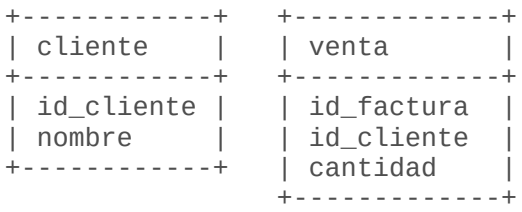
La selección de una clave primaria es muy importante en el diseño de una base de datos, ya que es un elemento clave de los datos que facilita la unión de tablas y el concepto total de una base de datos relacional.

Las claves primarias deben ser únicas y no nulas, de manera que garanticen que una fila de una tabla pueda ser siempre referenciada a través de su clave primaria.

MySQL requiere que se especifique NOT NULL para las columnas que se van a utilizar como claves primarias al momento de crear una tabla.

Claves foráneas e integridad referencial

Podemos decir de manera simple que integridad referencial significa que cuando un registro en una tabla haga referencia a un registro en otra tabla, el registro correspondiente debe existir. Por ejemplo, consideremos la relación entre una tabla cliente y una tabla venta.



Para poder establecer una relación entre dos tablas, es necesario asignar un campo en común a las dos tablas. Para este ejemplo, el campo id_cliente existe tanto en la tabla cliente como en la tabla venta. La mayoría de las veces, este campo en común debe ser una clave primaria en alguna de las tablas. Vamos a insertar algunos datos en estas tablas.

Tabla cliente

id_cliente	nombre
1	Juan penas
2	Pepe el Toro

Tabla venta

id_factura	id_cliente	cantidad
1	1	23
2	3	39
3	2	81

Hay dos registros en la tabla cliente, pero existen 3 id_cliente distintos en la tabla venta. Habíamos dicho que las dos tablas se relacionan con el campo id_cliente, por lo tanto, podemos decir que Juan Penas tiene una cantidad de 23, y Pepe el Toro 81, sin embargo, no hay un nombre que se corresponda con el id_cliente 3.

Las relaciones de claves foráneas se describen como relaciones padre/hijo (en nuestro ejemplo, cliente es el padre y venta es el hijo), y se dice que un registro es huérfano cuando su padre ya no existe.

Cuando en una base de datos se da una situación como esta, se dice que se tiene una integridad referencial pobre (pueden existir otra clase de problemas de integridad). Generalmente esto va ligado a un mal diseño, y puede generar otro tipo de problemas en la base de datos, por lo tanto debemos evitar esta situación siempre que sea posible.

En el pasado, MySQL no se esforzaba en evitar este tipo de situaciones, y la responsabilidad pasaba a la aplicación. Para muchos desarrolladores, esta no era una situación del todo grata, y por lo tanto no se consideraba a MySQL para ser usado en sistemas "serios". Por supuesto, esta fue una de las cosas más solicitadas en las anteriores versiones de MySQL; que se tuviera soporte para claves foráneas, para que MySQL mantenga la integridad referencial de los datos.

Una clave foránea es simplemente un campo en una tabla que se corresponde con la clave primaria de otra tabla. Para este ejemplo, el campo id_cliente en la tabla venta es la clave foránea. Nótese que este campo se corresponde con el campo id_cliente en la tabla cliente, en dónde este campo es la clave primaria.

Las claves foráneas tienen que ver precisamente con la integridad referencial, lo que significa que si una clave foránea contiene un valor, ese valor se refiere a un registro existente en la tabla relacionada.

Claves foráneas en MySQL

Estrictamente hablando, para que un campo sea una clave foránea, éste necesita ser definido como tal al momento de crear una tabla. Se pueden definir claves foráneas en cualquier tipo de tabla de MySQL, pero únicamente tienen sentido cuando se usan tablas del tipo InnoDB.

A partir de la versión 3.23.43b, se pueden definir restricciones de claves foráneas con el uso de tablas InnoDB. InnoDB es el primer tipo de tabla que permite definir estas restricciones para garantizar la integridad de los datos.

Para trabajar con claves foráneas, necesitamos hacer lo siguiente:

- Crear ambas tablas del tipo InnoDB.
- Usar la sintaxis FOREIGN KEY(campo_fk) REFERENCES nombre_tabla (nombre_campo)
- Crear un índice en el campo que ha sido declarado clave foránea.

InnoDB no crea de manera automática índices en las claves foráneas o en las claves referenciadas, así que debemos crearlos de manera explícita. Los índices son necesarios para que la verificación de las claves foráneas sea más rápida. A continuación se muestra como definir las dos tablas de ejemplo con una clave foránea.

```
CREATE TABLE cliente
(
    id_cliente INT NOT NULL,
    nombre VARCHAR(30),
    PRIMARY KEY (id_cliente)
) TYPE = INNODB;

CREATE TABLE venta
(
    id_factura INT NOT NULL,
    id_cliente INT NOT NULL,
    cantidad   INT,
    PRIMARY KEY(id_factura),
    INDEX (id_cliente),
    FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente)
) TYPE = INNODB;
```

La sintaxis completa de una restricción de clave foránea es la siguiente:

```
[CONSTRAINT símbolo] FOREIGN KEY (nombre_columna, ...)
                  REFERENCES nombre_tabla (nombre_columna, ...)
                  [ON DELETE {CASCADE | SET NULL | NO ACTION
                              | RESTRICT}]
                  [ON UPDATE {CASCADE | SET NULL | NO ACTION
                              | RESTRICT}]
```

Las columnas correspondientes en la clave foránea y en la clave referenciada deben tener tipos de datos similares para que puedan ser comparadas sin la necesidad de hacer una conversión de tipos. El tamaño y el signo de los tipos enteros debe ser el mismo. En las columnas de tipo caracter, el tamaño no tiene que ser el mismo necesariamente.

Si MySQL da un error cuyo número es el 1005 al momento de ejecutar una sentencia CREATE TABLE, y el mensaje de error se refiere al número 150, la creación de la tabla falló porque la restricción de la clave foránea no se hizo de la manera adecuada. De la misma manera, si falla una sentencia ALTER TABLE y se hace referencia al error número 150, esto significa que la definición de la restricción de la clave foránea no se hizo adecuadamente. A partir de la versión 4.0.13 de MySQL, se puede usar la sentencia SHOW INNODB STATUS para ver una explicación detallada del último error que se generó en relación a la definición de una clave foránea.

Si en una tabla, un registro contiene una clave foránea con un valor NULO, significa que no existe niguna relación con otra tabla.

A partir de la versión 3.23.50, se pueden agregar restricciones de clave foránea a una tabla con el uso de la sentencia ALTER TABLE. La sintaxis es:

```
ALTER TABLE nombre_tabla ADD [CONSTRAINT símbolo] FOREIGN KEY(...)
REFERENCES otra_tabla(...) [acciones_ON_DELETE][acciones_ON_UPDATE]
```

Por ejemplo, la creación de la clave foránea en la tabla venta que se mostró anteriormente pudo haberse hecho de la siguiente manera con el uso de una sentencia ALTER TABLE:

```
CREATE TABLE venta
(
    id_factura INT NOT NULL,
    id_cliente INT NOT NULL,
    cantidad   INT,
    PRIMARY KEY(id_factura),
    INDEX (id_cliente)
) TYPE = INNODB;

ALTER TABLE venta ADD FOREIGN KEY(id_cliente) REFERENCES
cliente(id_cliente);
```

En las versiones 3.23.50 y menores no deben usarse las sentencias ALTER TABLE o CREATE INDEX en tablas que ya tienen definidas restricciones de claves foráneas o bien, que son referenciadas en restricciones de claves foráneas: cualquier sentencia ALTER TABLE elimina todas las restricciones de claves foráneas definidas para la tabla.

No debe usarse una sentencia ALTER TABLE en una tabla que está siendo referenciada, si se quiere modificar el esquema de la tabla, se recomienda eliminar la tabla y volverla a crear con el nuevo esquema. Cuando MySQL hace un ALTER TABLE, puede que use de manera interna un RENAME TABLE, y por lo tanto, se confundan las restricciones de clave foránea que se refieren a la tabla. Esta restricción aplica también en el caso de la sentencia CREATE INDEX, ya que MySQL la procesa como un ALTER TABLE.

Cuando se ejecute un script para cargar registros en una base de datos, es recomendable agregar las restricciones de claves foráneas vía un ALTER TABLE. De esta manera no se tiene el problema de cargar los registros en las tablas de acuerdo a un orden lógico (las tablas referenciadas deberían ir primero).

Inserción de registros con claves foráneas

La integridad referencial se puede comprometer básicamente en tres situaciones: cuando se está insertando un nuevo registro, cuando se está eliminando un registro, y cuando se está actualizando un registro. La restricción de clave foránea que hemos definido se asegura que cuando un nuevo registro sea creado en la tabla venta, éste debe tener su correspondiente registro en la tabla cliente.

Una vez que hemos creado las tablas, vamos a insertar algunos datos que nos sirvan para demostrar algunos conceptos importantes:

```
mysql> INSERT INTO cliente VALUES(1,'Juan Penas');
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO cliente VALUES(2,'Pepe el toro');
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO venta VALUES(1,1,23);
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO venta VALUES(3,2,81);
Query OK, 1 row affected (0.03 sec)
```

En este momento no hay ningún problema, sin embargo, vamos a ver que sucede cuando intentamos insertar un registro en la tabla venta que se refiera a un cliente no existente cuyo id_cliente es 3:

```
mysql> INSERT INTO venta VALUES(2,3,39);
ERROR 1216: Cannot add or update a child row: a foreign key constraint
fails
```

El hecho es que MySQL no nos permite insertar este registro, ya que el cliente cuyo id_cliente es 3 no existe. La restricción de clave foránea asegura que nuestros datos mantienen su integridad. Sin embargo, ¿qué sucede cuando eliminamos algún registro?. Vamos a agregar un nuevo cliente, y un nuevo registro en la tabla venta, posteriormente eliminaremos el registro de nuestro tercer cliente:

```
mysql> INSERT INTO cliente VALUES(3,'Pepe pecas');
Query OK, 1 row affected (0.05 sec)
```

```
mysql> INSERT INTO venta VALUES(2,3,39);
Query OK, 1 row affected (0.05 sec)

mysql> DELETE FROM cliente WHERE id_cliente=3;
ERROR 1217: Cannot delete or update a parent row: a foreign key
constraint fails
```

Debido a nuestra restricción de clave foránea, MySQL no permite que eliminemos el registro de cliente cuyo id_cliente es 3, ya que se hace referencia a éste en la tabla venta. De nuevo, se mantiene la integridad de nuestros datos. Sin embargo existe una forma en la cuál podríamos hacer que la sentencia DELETE se ejecute de cualquier manera, y la veremos brevemente, pero primero necesitamos saber como eliminar (quitar) una clave foránea.

Eliminación de una clave foránea

No podemos sólo eliminar una restricción de clave foránea como si fuera un índice ordinario. Veamos que sucede cuando lo intentamos.

```
mysql> ALTER TABLE venta DROP FOREIGN KEY;
ERROR 1005: Can't create table '.test#sql-228_4.frm' (errno: 150)
```

Para eliminar la clave foránea se tiene que especificar el ID que ha sido generado y asignado internamente por MySQL a la clave foránea. En este caso, se puede usar la sentencia SHOW CREATE TABLE para determinar dicho ID.

```
mysql> SHOW CREATE TABLE venta;
+--------+-------------------------------------------------------+
| Table  | Create Table                                          |
+--------+-------------------------------------------------------+
| venta  | CREATE TABLE 'venta' (                                |
|        |  'id_factura' int(11) NOT NULL default '0',           |
|        |  'id_cliente' int(11) NOT NULL default '0',           |
|        |  'cantidad' int(11) default NULL,                     |
|        |  PRIMARY KEY  ('id_factura'),                         |
|        |  KEY 'id_cliente' ('id_cliente'),                     |
|        |  CONSTRAINT '0_22' FOREIGN KEY ('id_cliente')         |
|        |  REFERENCES 'cliente' ('id_cliente') ) TYPE=InnoDB    |
+-------+-------------------------------------------------------+
1 row in set (0.00 sec)
```

En nuestro ejemplo, la restricción tiene el ID 0_22 (es muy probable que este valor sea diferente en cada caso).

```
mysql> ALTER TABLE venta DROP FOREIGN KEY 0_22;
Query OK, 3 rows affected (0.23 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Eliminación de registros con claves foráneas

Una de las principales bondades de las claves foráneas es que permiten eliminar y actualizar registros en cascada.

Con las restriccciones de clave foránea podemos eliminar un registro de la tabla cliente y a la vez eliminar un registro de la tabla venta usando sólo una sentencia DELETE. Esto es llamado eliminación en cascada, en donde todos los registros relacionados son eliminados de acuerdo a las relaciones de clave foránea. Una alternativa es no eliminar los registros relacionados, y poner el valor de la clave foránea a NULL (asumiendo que el campo puede tener un valor nulo). En nuestro caso, no podemos poner el valor de nuestra clave foránea id_cliente en la tabla venta, ya que se ha definido como NOT NULL. Las opciones estándar cuando se elimina una registro con clave foránea son:

- ON DELETE RESTRICT
- ON DELETE NO ACTION
- ON DELETE SET DEFAULT
- ON DELETE CASCADE
- ON DELETE SET NULL

ON DELETE RESTRICT es la acción predeterminada, y no permite una eliminación si existe un registro asociado, como se mostró en el ejemplo anterior. ON DELETE NO ACTION hace lo mismo.

ON DELETE SET DEFAULT actualmente no funciona en MySQL - se supone que pone el valor de la clave foránea al valor por omisión (DEFAULT) que se definió al momento de crear la tabla.

Si se especifica ON DELETE CASCADE, y una fila en la tabla padre es eliminada, entonces se eliminarán las filas de la tabla hijo cuya clave foránea sea igual al valor de la clave referenciada en la tabla padre. Esta acción siempre ha estado disponible en MySQL.

Si se especifica ON DELETE SET NULL, las filas en la tabla hijo son actualizadas automáticamente poniendo en las columnas de la clave foránea el valor NULL. Si se especifica una acción SET NULL, debemos asegurarnos de no declarar las columnas en la tabla como NOT NULL.

A continuación se muestra un ejemplo de eliminación en cascada:

```
mysql> ALTER TABLE venta ADD FOREIGN KEY(id_cliente)
    -> REFERENCES cliente(id_cliente) ON DELETE CASCADE;
Query OK, 3 rows affected (0.23 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Vamos a ver como están nuestros registros antes de ejecutar la sentencia DELETE:

```
mysql> SELECT * FROM cliente;
+------------+--------------+
| id_cliente | nombre       |
+------------+--------------+
|          1 | Juan Penas   |
|          2 | Pepe el toro |
|          3 | Pepe pecas   |
+------------+--------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM venta;
+------------+------------+----------+
| id_factura | id_cliente | cantidad |
+------------+------------+----------+
|          1 |          1 |       23 |
|          2 |          3 |       39 |
|          3 |          2 |       81 |
+------------+------------+----------+
3 rows in set (0.00 sec)
```

Ahora eliminaremos a Pepe Pecas de la base de datos:

```
mysql> DELETE FROM cliente WHERE id_cliente=3;
Query OK, 1 row affected (0.05 sec)

mysql> SELECT * FROM venta;
+------------+------------+----------+
| id_factura | id_cliente | cantidad |
+------------+------------+----------+
|          1 |          1 |       23 |
|          3 |          2 |       81 |
+------------+------------+----------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM cliente;
+------------+--------------+
| id_cliente | nombre       |
+------------+--------------+
|          1 | Juan Penas   |
|          2 | Pepe el toro |
+------------+--------------+
```

```
2 rows in set (0.00 sec)
```

Con la eliminación en cascada, se ha eliminado el registro de la tabla venta al que estaba relacionado Pepe Pecas.

Actualización de registros con claves foráneas

Las opciones correspondientes a ON UPDATE están disponibles a partir de la versión 4.0.8.

Estas opciones son muy similares cuando se ejecuta una sentencia UPDATE, en lugar de una sentencia DELETE. Estas son:

- ON UPDATE CASCADE
- ON UPDATE SET NULL
- ON UPDATE RESTRICT

Vamos a ver un ejemplo, pero antes que nada, tenemos que eliminar la restricción de clave foránea (debemos usar el ID específico de nuestra tabla).

```
mysql> ALTER TABLE venta DROP FOREIGN KEY 0_26;
Query OK, 2 rows affected (0.22 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE venta ADD FOREIGN KEY(id_cliente)
    -> REFERENCES cliente(id_cliente) ON DELETE RESTRICT ON UPDATE
CASCADE;
Query OK, 2 rows affected (0.22 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

NOTA: Se debe especificar ON DELETE antes de ON UPDATE, ya que de otra manera se recibirá un error al definir la restricción.

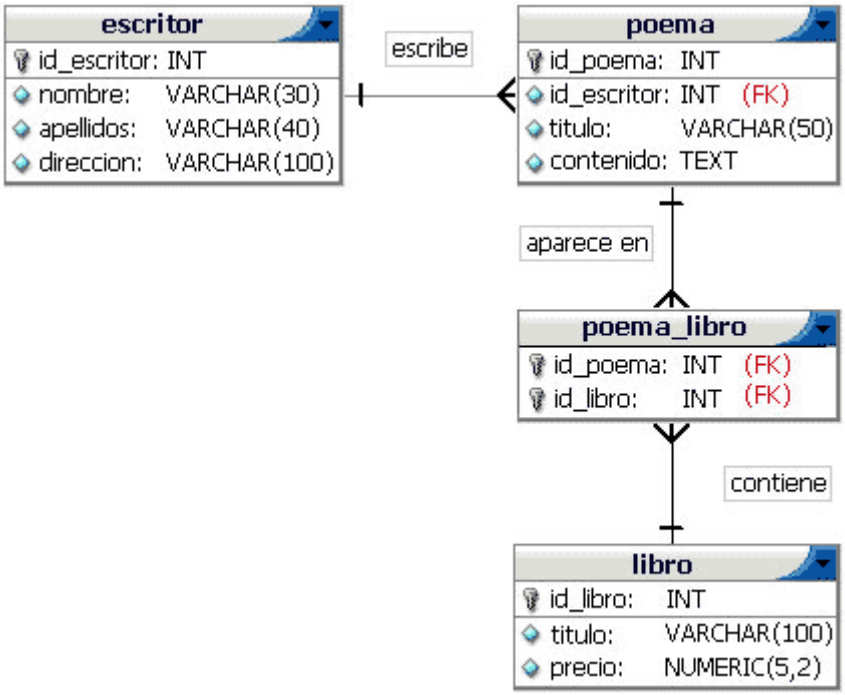
Ahora está lista la clave foránea para una actualización en cascada. Este es el ejemplo:

```
mysql> SELECT * FROM venta;
+------------+------------+----------+
| id_factura | id_cliente | cantidad |
+------------+------------+----------+
|          1 |          1 |       23 |
|          3 |          2 |       81 |
+------------+------------+----------+
2 rows in set (0.00 sec)

mysql> UPDATE cliente SET id_cliente=10 WHERE id_cliente=1;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM venta;
+------------+------------+----------+
| id_factura | id_cliente | cantidad |
+------------+------------+----------+
|          1 |         10 |       23 |
|          3 |          2 |       81 |
+------------+------------+----------+
2 rows in set (0.00 sec)
```

En este caso, al actualizar el valor de id_cliente en la tabla cliente, se actualiza de manera automática el valor de la clave foránea en la tabla venta. Esta es la actualización en cascada.

Un ejemplo más

Observar y estudiar detenidamente el diagrama entidad/relación de la figura que se muestra a continuación.



Queda como ejercicio al lector verificar que a partir de este diagrama se genera un código SQL similar al mostrado a continuación, que nos sirve para la creación de las tablas con sus correspondientes definiciones de claves foráneas:

Considerar que se desean hacer eliminaciones y actualizaciones en cascada, y que en la tabla poema_libro la clave primaria está formada por ambos campos (id_poema y id_libro).

```
CREATE TABLE libro (
  id_libro INT NOT NULL,
  titulo VARCHAR(100) NULL,
  precio NUMERIC(5,2) NULL,
  PRIMARY KEY(id_libro)
) TYPE=InnoDB;

CREATE TABLE escritor (
  id_escritor INT NOT NULL,
  nombre VARCHAR(30) NULL,
  apellidos VARCHAR(40) NULL,
  direccion VARCHAR(100) NULL,
  PRIMARY KEY(id_escritor)
) TYPE=InnoDB;

CREATE TABLE poema (
  id_poema INT NOT NULL,
  id_escritor INT NOT NULL,
  titulo VARCHAR(50) NULL,
  contenido TEXT NULL,
  PRIMARY KEY(id_poema),
  INDEX(id_escritor),
  FOREIGN KEY(id_escritor) REFERENCES escritor(id_escritor)
        ON DELETE CASCADE ON UPDATE CASCADE
) TYPE=InnoDB;
```

```
CREATE TABLE poema_libro (
  id_poema INT NOT NULL,
  id_libro INT NOT NULL,
  PRIMARY KEY(id_poema, id_libro),
  INDEX (id_poema), INDEX(id_libro),
  FOREIGN KEY(id_poema) REFERENCES poema(id_poema)
        ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY(id_libro) REFERENCES libro(id_libro)
        ON DELETE CASCADE ON UPDATE CASCADE
) TYPE=InnoDB;
```