Manejo de Tablas y Llaves Foráneas

Llaves Foráneas

Para manejar las llaves foráneas, deben crearse al menos dos tablas, la primera tal y como se ha hecho anteriormente

```
CREATE TABLE nombre_tabla (
nombre_columna_1 TIPO AUTO_INCREMENT PRIMARY KEY,
nombre_columna_2 TIPO,
nombre_columna_3 TIPO
) ENGINE=InnoDB
```

La segunda debe hacer referencia a una llave de la primera, ésto se logra con la sentencia **FOREIGN KEY** (llave_referenciada) **REFERENCES** tabla_referenciada (llave_referenciada) (Debe ser la llave primaria).

```
CREATE TABLE nombre_tabla_1 (
nombre_columna_1 TIPO,
nombre_llave_foranea TIPO NOT NULL,
FOREIGN KEY (nombre_llave_foranea) REFERENCES
nombre_tabla_referencia (nombre_columna_referencia)
) ENGINE=InnoDB
```

También es posible crear una llave primaria para la tabla que contiene una llave foránea, ésta llave primaria normalmente será de tipo Auto incrementable.

```
CREATE TABLE nombre_tabla_1 (
nombre_columna_primaria TIPO AUTO_INCREMENT PRIMARY KEY,
nombre_columna_1 TIPO,
nombre_llave_foranea TIPO NOT NULL,
FOREIGN KEY (nombre_llave_foranea) REFERENCES
nombre_tabla_referencia (nombre_columna_referencia)
) ENGINE=InnoDB
```

Restricciones con las Llaves Foráneas

Es posible indicar restricciones para cuando se realicen operaciones entre las tablas que están unidas a través de llaves foráneas. La primera restricción es que no se puede insertar un elemento en una tabla que contiene una llave foránea a menos que el valor al que se haga referencia exista en la tabla a la que se hace referencia.

Esa restricción es automática, sin embargo hay otras que se deben especificar, en particular cuando se elimina o se actualiza un valor, para esto se utiliza la estructura **ON DELETE** para la eliminación u **ON UPDATE** para una actualización seguido de alguno de los siguientes modificadores:

- **RESTRICT**: no permite eliminar o modificar filas en la tabla referenciada si existen filas con el mismo valor de clave foránea.
- **CASCADE**: el borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado de clave, implica borrar las filas con el mismo valor de clave foránea o modificar los valores de esas claves foráneas.
- **SET NULL**: borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado de clave, implica asignar el valor NULL a las claves foráneas con el mismo valor.
- **NO ACTION**: las claves foráneas no se modifican, ni se eliminan filas en la tabla que las contiene.
- **SET DEFAULT**: borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado implica asignar el valor por defecto a las claves foráneas con el mismo valor.

Ejemplo:

Si se quiere impedir que se pueda borrar una tupla con una cierta llave primaria de la tabla referenciada si esa llave tiene valores en otra tabla como una llave foránea, se debería escribir:

```
CREATE TABLE nombre_tabla_1 (
nombre_columna_1 TIPO,
nombre_llave_foranea TIPO NOT NULL,
FOREIGN KEY (nombre_llave_foranea) REFERENCES
nombre_tabla_referencia (nombre_columna_referencia)
ON DELETE RESTRICT) ENGINE=InnoDB
```

Si se quiere que al actualizar el valor de una llave primaria, esto se refleje en los valores de las llaves foráneas que tienen a la tabla como referencia, se debería escribir:

```
CREATE TABLE nombre_tabla_1 (
nombre_columna_1 TIPO,
nombre_llave_foranea TIPO NOT NULL,
FOREIGN KEY (nombre_llave_foranea) REFERENCES
nombre_tabla_referencia (nombre_columna_referencia)
ON UPDATE CASCADE
) ENGINE=InnoDB
```

También es posible asignar valores de UPDATE Y DELETE al crear la tabla.

```
CREATE TABLE nombre_tabla_1 (
nombre_columna_1 TIPO,
nombre_llave_foranea TIPO NOT NULL,
FOREIGN KEY (nombre_llave_foranea) REFERENCES
nombre_tabla_referencia (nombre_columna_referencia)
ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB
```

Para crear una tabla que tenga referencias de otra tabla, la tabla referenciada, ya deberá existir, de lo contrario se indicará un error.

Manejo de Llaves Foráneas

No es posible cambiar nombre o eliminar una tabla o columna cuando ésta es usada como referencia por otra a través de una Llave Foránea. Si se desean eliminar o modificar tablas o columnas a las que se hace referencia, se debe eliminar la relación entre las tablas.

Para esto se debe conocer el nombre de la restricción (*Constraint*) que se creó al momento de ligar las tablas, aunque también es posible asignar nombres a las restricciones de las relaciones al momento de crearlas.

Para esto se agrega la palabra **CONSTRAINT** seguido del nombre de la restricción antes de declarar la llave foránea como se vio anteriormente.

```
CREATE TABLE nombre_tabla_1 (
nombre_columna_1 TIPO,
nombre_llave_foranea TIPO NOT NULL,
CONSTRAINT nombre_restriccion FOREIGN KEY (nombre_llave_foranea) REFERENCES
nombre_tabla_referencia (nombre_columna_referencia)
ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB
```

Para eliminar la relación, no se eliminan las columnas como tal, se elimina la relación (eliminando la restricción).

```
ALTER TABLE nombre_tabla DROP FOREIGN KEY nombre_restriccion
```

A pesar de que se utiliza la sentencia **DROP FOREIGN KEY**, no se coloca el nombre de la columna que es una llave foránea, se coloca el nombre de la restricción.

Una vez hecho esto, ya es posible eliminar o cambiar el nombre de la columna o tabla que es referenciada.

Consultas entre Múltiples Tablas

Se puede involucrar varias tablas al momento de realizar consultas o especificar condiciones para una consulta, eliminación o actualización .

Producto cartesiano

```
SELECT * FROM nombre_tabla_1,nombre_tabla_2
```

Regresará el producto Cartesiano de las dos tablas, es equivalente a la operación **CROSS JOIN**

Para consultar varias columnas de diferentes tablas se puede utiliza la siguiente sintaxis :

```
SELECT nombre_tabla_1.columna_1, nombre_tabla_1.columna_2,
nombre_tabla_1.columna_3, nombre_tabla_2.columna_1, nombre_tabla_2.columna_2
FROM nombre_tabla_1,nombre_tabla_2
```

Esto regresará los elementos seleccionados de las tablas, lo más recomendable es colocar una sentencia que involucre a las llaves que relaciona a estas tablas, de lo contrario se estará obteniendo un producto cartesiano.

Operación Join

Esto regresará los elementos de la tabla tabla_A y de la tabla_B que coincidan en el atributo (columna).

```
SELECT * FROM tabla_A JOIN tabla_B USING(columna)
```

Si no existe el mismo atributo, se puede realizar lo mismo usando la expresión:

```
SELECT * FROM tabla_A JOIN tabla_B WHERE (tabla_A .columna_A = tabla_B.columna_B)
```

Insertando Datos

Insertar datos en una tabla con llaves foráneas, es igual que en una tabla que no las tiene, pero se debe estar seguro de que antes de insertar los datos, sus correspondientes datos referentes, estén presentes en las tablas a las que se hace referencia.

Actualizando y Eliminado Datos

Es posible actualizar y eliminar datos en las tablas siempre y cuando éstos no sean referencia o referenciados de otras tablas.

No se puede eliminar el atributo de llave primaria si existe una tabla que la usa como llave foránea, primero será necesario quitar la propiedad de llave foránea de la tabla que la contiene.

No es posible eliminar una propiedad de Llave Foránea de manera directa, se debe eliminar a través de la restricción asociada a ella. Ésta restricción se crea al momento de especificar la propiedad de Llave Foránea. En caso de que no se haya indicado ningún nombre, se le asigna uno de manera automática, por lo que se debe consultar cuál es:

Esto se realiza con la sentencia:

```
SHOW CREATE TABLE nombre_tabla
```

Con esto se puede conocer el nombre de la restricción asociada a la Llave Foránea. Finalmente para eliminarla se utiliza:

```
ALTER TABLE nombre_tabla DROP FOREIGN KEY nombre_restriccion
```

Considerar que el quitar el atributo de llave primaria o llave foránea no implica quitar la columna.

Actualizando y Eliminado Datos con Clausula “Where”

Es posible utilizar los resultados de una consulta WHERE (en una o múltiples tablas) como criterio para actualizar o eliminar ciertos datos. Solo se debe especificar que el valor de la columna a utilizar como criterio se llena con los resultados de una clausula WHERE.

Al realizar una asignación con igual (=), solo funciona si se espera obtener un resultado, es mejor utilizar la sentencia IN, de esa forma se puede manejar más de un resultado.

Considerar que algunos manejadores no permiten utilizar la misma tabla en la que se actualiza o elimina como parte de la sentencias FROM que obtendrá los resultados.