Tipos de tablas

Aunque en los temas anteriores no hemos hecho alusión a ello, MySQL permite usar diferentes tipos de tablas tales como:
ISAM
MyISAM
InnoDB
Las tablas **ISAM** son las de formato más antiguo. Están limitadas a tamaños que no superen los 4 gigas y no permite copiar tablas entre máquinas con distinto sistema operativo.

Las tablas **MySAM** son el resultado de la evolución de las anteriores, ya que resuelven el problema que planteaban las anteriores y son **el formato por defecto** de MySQL a partir de su versión 3.23.

Las tablas del tipo **InnoDB** tienen una estructura distinta que MyISAM, ya que utilizan **un sólo archivo** por tabla en ver de los tres habituales en los tipos anteriores.

Incorporan un par de ventajas importantes, ya que permiten **realizar transacciones** y definir reglas de **integridad referencial**.

Creación y uso de tablas InnoDB

La creación de tablas de este tipo no presenta ninguna dificultad añadida. El proceso es idéntico a las tablas habituales sin más que añadir *Type=InnoDB* después de cerrar el paréntesis de la sentencia de creación de la tabla.

Una vez creadas, las tablas InnoDB se comportan –a efectos de uso– exactamente igual que las que hemos venido utilizando en las páginas anteriores. No es preciso hacer ningún tipo de modificación en la sintaxis. Por tanto, es totalmente válido todo lo ya comentado respecto a: altas, modificaciones, consultas y bajas.

Las transacciones

Uno de los riesgos que se plantean en la gestión de bases de datos es que pueda producirse una interrupción del proceso mientras se está actualizando una o varias tablas. Pongamos como ejemplo el cobro de nuestra nómina. Son necesarias dos anotaciones simultáneas: Il *cargo* en la cuenta del Organismo pagador y el *abono* en nuestra cuenta bancaria. Si se interrumpiera fortuitamente el

Creación de una tabla InnoDB

La creación de tablas tipo InnoDB requiere una de estas dos sentencias:

CREATE TABLE IF NOT EXISTS *tabla* (*campo1, campo2,...*) Type=InnoDB

CREATE TABLE *tabla* (*campo1, campo2,...*) Type=InnoDB

Este script crear una tabla *InnoDB* con idénticos campos a los utilizados en el caso de la tabla **demo4** con la que hemos venido trabajando hasta ahora. La sintaxis, muy similar a la utilizada allí es esta:

```
<?
$base="ejemplos";
$tabla="demoINNO";
$c=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $c);

$crear="CREATE TABLE  $tabla (";
$crear.="Contador TINYINT(8)  UNSIGNED ZEROFILL NOT NULL AUTO_INCREMENT,";
$crear.="DNI CHAR(8) NOT NULL,  ";
$crear.="Nombre VARCHAR (20)  NOT NULL, ";
$crear.="Apellido1 VARCHAR (15)  not null, ";
$crear.="Apellido2 VARCHAR (15)  not null, ";
$crear.="Nacimiento DATE DEFAULT '1970-12-21', ";
$crear.="Hora TIME DEFAULT '00:00:00', ";
$crear.="Sexo Enum('M','F') DEFAULT 'M' not null, ";
$crear.="Fumador CHAR(0) , ";
$crear.="Idiomas SET(' Castellano',' Francés','Inglés',
         ' Alemán',' Búlgaro',' Chino'), ";
$crear.=" PRIMARY KEY(DNI), ";
$crear.=" UNIQUE auto (Contador)";
$crear.=")";
# esta es la única diferencia con el proceso de
# creación de tablas MyISAM
 $crear.=" Type=InnoDB";

if(mysql_query ($crear ,$c)) {
echo "<h2> Tabla $tabla creada con EXITO </h2><br>";
    }else{
echo "<h2> La tabla $tabla NO HA PODIDO CREARSE ";
# echo mysql_error ($c)."<br>";
$numerror=mysql_errno ($c);
    if ($numerror==1050){echo "porque YA EXISTE</h2>";}
};
        mysql_close($c);
?>
```

Crear tabla tipo InnoDB

¡Cuidado!

Bajo Windows, al crear una base de datos o tabla InnoDB el nombre de la misma aparecerá en minúsculas independientemente de la sintaxis que hayamos utilizado en su creación.

Si observas el ejemplo anterior, hemos puesto **demoINNO** como nombre de la tabla. Sin embargo, si miras el directorio c:\mysql verás que aparece el fichero **demoinno.frm** con minúsculas.

proceso en el intermedio de las dos operaciones podría darse la circunstancia de que apareciera registrado el pago sin que se llegaran a anotar los haberes en nuestra cuenta.

Las transacciones evitan este tipo de situaciones ya que los registros de los datos se registran de manera provisional y no toman carácter definitivo hasta que una instrucción confirme que esas anotaciones tienen carácter definitivo. Para ello, MySQL dispone de tres sentencias: **BEGIN**, **COMMIT** y **ROLLBACK**.

Sintaxis de las transacciones

Existen tres sentencias para gestionar las transacciones. Son las siguientes:

*mysql_query("**BEGIN**",$c)*

Su ejecución requiere que este activa la conexión **$c** con el servidor de base de datos e indica a MySQL que **comienza una transacción**.

Todas las sentencias que se ejecuten a partir de ella tendrán carácter *provisional* y no se ejecutarán de forma efectiva hasta que encuentre una sentencia de finalización.

*mysql_query("**ROLLBACK**",$c)*

Mediante esta sentencia *advertimos* a MySQL que *finaliza la transacción* pero que **no debe hacerse efectiva** ninguna de las modificaciones incluidas en ella.

*mysql_query("**COMMIT**",$c)*

Esta sentencia advierte a MySQL que ha finalizado la transacción y que **debe hacer efectivos** todos los cambios incluidos en ella.

Precauciones a tener en cuenta

Cuando se utilizan campos autoincrementales en tablas InnoDB los contadores se van incrementando al añadir registros (incluso de forma provisional) con lo cual si se *aborta* la inclusión con un *ROLLBACK* ese contador mantiene el incremento y en inserciones posteriores partirá de ese valor acumulado.

Por ejemplo. Si partimos de una tabla vacía y hacemos una transacción de dos registros (número 1 y número 2 en el campo autoincremental) y la finalizamos con ROLLBACK, no se insertarán pero en una inserción posterior el contador autoincremental comenzará a partir del valor 2.

MySQL anuncia que a partir de la versión 5.0.3 se incluirá una nueva sentencia para permitir que se puedan *renumerar* los campos

Las primeras transacciones

```
 <?
$base="ejemplos";
# escribimos el nombre de la tabla en MINUSCULAS
# para asegurar la compatibilidad entre plataformas
$tabla="demoinno";
$conexion=mysql_connect("localhost","pepe","pepa");
mysql_select_db ($base, $conexion);
# insertamos la sentencia BEGIN para indicar el comienzo
# de una transacción
mysql_query("BEGIN",$conexion);
/* hasta que no aparezca una sentencia que finalice la transacción
   (ROLLBACK ó COMMIT) las modificaciones en la tabla serán registradas
   de forma "provisional") */
mysql_query("INSERT $tabla (DNI,Nombre,Apellido1,Apellido2,
                Nacimiento,Sexo,Hora,Fumador,Idiomas)
             VALUES
                 ('111111','Alvaro','Alonso','Azcárate','1954-11-23',
                   'M','16:24:52',NULL,3)",$conexion);
if (mysql_errno($conexion)==0){
            echo "Registro AÑADIDO<br>";
      }else{
        if (mysql_errno($conexion)==1062){
                   echo "No ha podido añadirse el registro <br>";
                   echo "Ya existe un registro con este DNI<br>";
                  }else{
                   $numerror=mysql_errno($conexion);
                   $descrerror=mysql_error($conexion);
                   echo "Se ha producido un error nº $numerror<br>";
                   echo "<br>que corresponde a: $descrerror<br>";
                }
}
# indicamos el final de la transacción, en este caso con ROLLBACK
# por lo tanto el registro con DNI 111111 no será insertado en la tabla
mysql_query("ROLLBACK",$conexion);
# incluyamos una nueva transacción
mysql_query("BEGIN",$conexion);
mysql_query("INSERT $tabla (DNI,Nombre,Apellido1,Apellido2,
                Nacimiento,Sexo,Hora,Fumador,Idiomas)
             VALUES
                 ('222222','Genoveva','Zarabozo','Zitrón','1964-01-14',
                   'F','16:18:20',NULL,2)",$conexion);
if (mysql_errno($conexion)==0){
            echo "Registro AÑADIDO";
      }else{
        if (mysql_errno($conexion)==1062){
                   echo "No ha podido añadirse el registro <br>";
                   echo "Ya existe un registro con este DNI";
                  }else{
                   $numerror=mysql_errno($conexion);
                   $descrerror=mysql_error($conexion);
                   echo "Se ha producido un error nº $numerror";
                   echo "<br>que corresponde a: $descrerror";
                }
}
# indicamos el final de la transacción, en este caso con COMMIT
# por lo tanto el registro con DNI 222222 si será insertado en la tabla
mysql_query("COMMIT",$conexion);
# leamos el contenido de la tabla para ver el resultado
$resultado= mysql_query("SELECT * FROM $tabla" ,$conexion);
print "<br>Lectura de la tabla depués del commit<br>";
while ($registro = mysql_fetch_row($resultado)){
       foreach($registro  as $clave){
           echo $clave,"<br>";
       }
}
mysql_close();
?>
```

Ejecutar el ejemplo

Integridad referencial en tablas InnoDB

autoincrementales.

Elementos necesarios para la integridad referencial

La integridad referencial ha de establecerse siempre entre dos tablas. Una de ellas ha de comportarse como **tabla principal** (suele llamarse *tabla padre* y la otra sería la **tabla vinculada** ó *tabla hijo*.

Es imprescindible:
Que la **tabla principal** tenga un índice primario (PRIMARY KEY)
Que la **tabla vinculada** tenga un índice (no es necesario que sea ni **único** ni **primario**) asociado a campos de tipo idéntico a los que se usen para índice de la tabla principal.
Si observas el código fuente del ejemplo que tienes a la derecha podrás observar que utilizamos el número del DNI (único para alumno) como elemento de vinculación de la tabla de datos personales con la que incluye las direcciones.

Borrado de tablas vinculadas

Si pretendemos eliminar una tabla principal recibiremos un mensaje de error tal como puedes ver si ejecutas este ejemplo cuyo código fuente tienes aquí. cómo es lógico, antes de ejecutarlo habrás de tener creada la tabla cuyo código fuente tienes a la derecha.

Las tablas vinculadas si permiten el borrado y una vez que éstas ya han sido eliminadas (o quitada la vinculación) ya podrán borrarse sin problemas las tablas principales. Si ejecutas este ejemplo podrás observar que borramos ambas tablas siguiendo el orden que permite hacerlo. Primero se borra la vinculada y luego la principal. Este es el código fuente del script.

¡Cuidado!

Si has borrado las tablas con los ejemplos anteriores no olvides crearlas de nuevo para poder visualizar los ejemplos siguientes.

Modificación o borrado de campos vinculados

Las sentencias MySQL que deban modificar o eliminar campos utilizados para establecer vínculos entre tablas requieren de un parámetro especial (CONSTRAINT) -puede ser distinto en cada una de las tablas- que es necesario conocer previamente.

La forma de visualizarlo es ejecutar

Cuando se trabaja con varias tablas que tienen algún tipo de vínculo resulta interesante disponer de mecanismos que protejan o impidan acciones no deseadas. Supongamos, como veremos en los ejemplos posteriores que pretendemos utilizar una tabla con datos de alumnos y otra tabla distinta para las calificaciones de esos alumnos. Si no tomamos ninguna precaución (bien sea mediante los script o mediante el diseño de las tablas) podría darse la circunstancia de que incluyéramos calificaciones a alumnos inexistentes, en materias de las que no están matriculados, etcétera. También podría darse la circunstancia de que diéramos de baja a un alumno pero que se mantuvieran las calificaciones vinculadas a él. Todas estas circunstancias suelen producir efectos indeseados y las tablas InnoDB pueden ser diseñadas para prever este tipo de situaciones.

Sintaxis para la vinculación de tablas

Los vínculos entre tablas suelen establecer en el momento de la creación de la *tabla vinculada*.

CREATE TABLE *tabla* (*campo1, campo2,...*
KEY *nombre* (*campo de vinculacion*),
FOREIGN KEY (*campo de vinculacion*)
REFERENCES *nombre_de la tabla principal* (*Indice primario de la tabla principal*)
) Type=InnoDB

donde el *campo de vinculacion* ha de ser un índice (no es necesario que sea PRIMARY KEY ni UNIQUE) y donde *Indice primario de la tabla principal* ha de ser un índice primario (PRIMARY KEY) de la tabla principal. Debe haber plena coincidencia (tanto en tipos como contenidos) entre ambos índices.

```
<?
$base="ejemplos";
$tabla1="principal";
$tabla2="vinculada";
$c=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $c);
# creación de la tabla principal type InnoDB
$crear="CREATE TABLE IF NOT EXISTS $tabla1 (";
$crear.="DNI CHAR(8) NOT NULL,  ";
$crear.="Nombre VARCHAR (20)  NOT NULL, ";
$crear.="Apellido1 VARCHAR (15)  not null, ";
$crear.="Apellido2 VARCHAR (15)  not null, ";
# el indice primario es imprescindible. Recuerda que debe
# estar definido sobre campos NO NULOS
$crear.=" PRIMARY KEY(DNI) ";
$crear.=")";
$crear.=" Type=InnoDB";
# creamos la tabla principal comprobando el resultado
if(@mysql_query ($crear ,$c)){
        print "La tabla ".$tabla1." ha sido creada<br>";
}else{
        print "No se ha creado ".$tabla1." ha habido un error<br>";
}
# crearemos la tabla vinculada
$crear="CREATE TABLE IF NOT EXISTS $tabla2 (";
$crear.="IDENTIDAD CHAR(8) NOT NULL, ";
$crear.="calle VARCHAR (20), ";
$crear.="poblacion VARCHAR (20), ";
$crear.="distrito VARCHAR(5), ";
# creamos el índice (lo llamamos asociador) para la vinculación
# en este caso no será ni primario ni único
# Observa que el campo IDENTIDAD de esta tabla CHAR(8)
# es idéntico al campo DNI de la tabla principal
$crear.=" KEY asociador(IDENTIDAD), ";
#establecemos la vinculación de ambos índices
$crear.=" FOREIGN KEY (IDENTIDAD) REFERENCES $tabla1(DNI) ";
$crear.=") TYPE = INNODB";
#  creamos (y comprobamos la creación) la tabla vinculada
if(@mysql_query ($crear ,$c)){
        print "La tabla ".$tabla2." ha sido creada<br>";
}else{
        print "No se ha creado ".$tabla2." ha habido un error<br>";
}
mysql_close();
?>
```

Crear tablas vinculadas

Modificación de estructuras

TABLE *nombre tabla* que devuelve como resultado un array asociativo con dos índices. Uno de ellos -llamado *Table*- que contiene el nombre de la tabla y el otro -Create Table- que contiene la estructura con la que ha sido creada la tabla pero incluyendo el parámetro **CONSTRAINT** seguido de su valor. Ese valor es precisamente el que necesitamos para hacer modificaciones en los campos asociados de las tablas vinculadas.

Pulsando en este enlace cuyo código fuente tienes aquí podrás visualizar el resultado de la ejecución de esa sentencia.

Conocido el valor de parámetro anterior el proceso de **borrado** del vínculo actual requiere la siguiente sintaxis:

ALTER TABLE *nombre de la tabla* **DROP FOREIGN KEY** ***parametro***

Cuando se trata de añadir un nuevo vínculo con una tabla principal habremos de utilizar la siguiente sentencia:

ALTER TABLE *nombre de la tabla* **ADD [CONSTRAINT** *parametro***] FOREIGN KEY** *parametro* **REFERENCES** *tabla principal(clave primaria)*

El parámetro CONSTRAIT (encerrado en corchetes en el párrafo anterior) es OPCIONAL y solo habría de utilzarse en el caso de que existiera ya una vinculación previa de esa tabla.

La función preg_match

En el ejemplo de la derecha utilizamos esta función cuya sintaxis es la siguiente:
preg_match(*pat, cad, coin* **)**

donde **pat** es un patrón de búsqueda, **cad** es la cadena que la han de realizarse las búsquedas y **coin** es un arroay que recoge todas las coincidencias encontradas en la cadena.

El patrón de búsqueda que hemos utilizado en el ejemplo /CONSTRAINT.*FOREIGN KEY/ debe interpretarse de la siguiente forma. Los caracteres / indican el comienzo y el final del patrón, el . indica que entre CONSTRAIT y FOREING se permite cualquer carácter para admitir la coincidencia y, además, * indica que ese caracter cualquier puede repetirse cero o más veces.

¡Cuidado!

Es posible que el script de la derecha te de un error como consecuencia

La modificación de estructuras en tablas vinculadas puede hacerse de forma idéntica a la estudiada para los casos generales de MySQL siempre que **esas modificaciones no afecten a los campos mediante los que se establecen las vinculaciones entre tablas**.

Aquí tienes un ejemplo en se borran y añaden campos en ambas tablas. Como puedes ver la sintaxis es exactamente la misma que utilizamos en temas anteriores.

```
<?
$base="ejemplos";
$tabla="principal";
$tabla1="vinculada";
$c=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $c);

if(mysql_query("ALTER TABLE $tabla ADD Segundo_Apellido VARCHAR(40)",$c)){
        print "Sea ha creado el nuevo campo en ".$tabla."<br>";
        }
if(mysql_query("ALTER TABLE $tabla DROP Apellido2",$c)){
        print "Sea ha borrado el campo Apellido 2 en ".$tabla."<br>";
        }

if(mysql_query("ALTER TABLE $tabla1 ADD DP VARCHAR(5)",$c)){
        print "Sea ha creado el nuevo campo en ".$tabla1."<br>";
        }
if(mysql_query("ALTER TABLE $tabla1 DROP distrito",$c)){
        print "Sea ha borrado el campo distrito en ".$tabla1."<br>";
        }

mysql_close();
?>
```

Ejecutar el ejemplo

En este otro ejemplo determinaremos el valor de CONSTRAINT y modificaremos campos asociados de la tabla vinculada.

```
<?
$base="ejemplos";
$tabla="vinculada";
$tabla1="principal";
$c=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $c);
$resultado=mysql_query( "SHOW CREATE TABLE $tabla",$c);
$v=mysql_fetch_array($resultado);
# extreamos de Create Table la cadena que empiza por CONSTRAINT
# y que acaba por FOREING KEY lo guardamos en el array $coin
preg_match ('/CONSTRAINT.*FOREIGN KEY/', $v['Create Table'], $coin);
# extraemos el parametro quitando el CONSTRAINT que lleva delante
# y el FOREIGN KEY que lleva al final
$para=str_replace("FOREIGN KEY",'',str_replace("CONSTRAINT",'',$coin[0]));
print "El valor de CONSTRAINT es: ".$para."<br>";
# eliminamos el vinculo con la clave externa incluyendo en la sentancia
#  el valor del parametro obtenido el proceso anterior
if(mysql_query("ALTER TABLE $tabla DROP FOREIGN KEY $para",$c)){
   print "Se ha realizado con éxito el borrado del vínculo<br>";
}
# añadimos el nuevo vínculo (en este caso rehacemos el anterior
# pero el proceso es idéntico)
if(mysql_query("ALTER TABLE $tabla ADD CONSTRAINT $para
                FOREIGN KEY(IDENTIDAD) REFERENCES $tabla1(DNI)",$c)){
 print "Se ha reestablecido con éxito vínculo<br>";
}

mysql_close();
?>
```

Ejecutar el ejemplo

Añadir registros a la tabla vinculada

Si ejecutas este ejemplo habiendo seguido la secuencia de estos materiales verás que se produce un error nº **1216**. Es lógico que así sea porque estamos intentando añadir un registro a la tabla

de que hemos podido modificar campos en los ejemplos anteriores. Si eso ocurre, borra aquí las tablas y genéralas de nuevo pulsando aquí.

¡Cuidado!

Los resultados que obtengas el ejecutar los ejemplos de borrado y modificación de datos pueden arrojar resultados distintos según los contenidos de las tablas que, a su vez, serán consecuencia de los ejemplos que hayas ejecutado anteriormente y de la secuencia de los mismos. Siempre puedes volver a las condiciones iniciales de los enlaces de la advertencia anterior.

Opciones adicionaes de FOREIGN KEY

La claúsula FOREIGN KEY permite añadirte -detrás de la definición ya comentada y **sin poner coma separándola de ella**- los parámetros ON DELETE y ON UPDATE en las que se permite especificar una de las siguientes opciones:

ON DELETE RESTRICT
Esta condición (es la condición por defecto de MySQL y no es preciso escribirla) indica a MySQL que **interrumpa el proeceso de borrado** y de un mensaje de error cuando se intente borrar un registro de la tabla principal cuando en la tabla vinculada existan registros asociados al valor que se pretende borrar.

ON DELETE NO ACTION
Es un sinónimo de la anterior.

ON DELETE CASCADE
Cuando se especifica esta opción, al borrar un registro de la tabla principal se borrarán de forma automática todos los de la tabla vinculada que estuvieran asociados al valor de la clave foránea que se trata de borrar. Con ello se conseguiría una actualización automática de la segunda tabla y se mantendría la identidad referencial.

ON DELETE SET NULL
Con esta opción, al borrar el registro de la tabla principal **no se borrarían** los que tuviera asociados la tabla secundaria pero tomarían valor **NULL** todos los índices de ella vinculada y ello requeriría que en el campo **DNI de la tabla principal** existiera un registro con un valor igual al que pretendemos introducir en la tabla vinculada.

Ver script	Insertar en tabla vinculada

Añadiremos un registro a la tabla principal con el DNI anterior:

Ver script	Insertar en tabla principal

y ahora ya podremos ejecutar -sin errores- el script que inserta datos en la tabla vinculada. Podremos ejecutar aquel script tantas veces como queramos ya que -el campo IDENTIDAD está definido como KEY y por tanto permite duplicados- no hemos establecido la condición de índice PRIMARIO ó UNICO.

Borrar o modificar registros en la tabla principal

Sin intentamos borrar un registro de la tabla principal mediante un script como el que tienes en este ejemplo verás que se produce un error nº **1217** advirtiéndonos de que no se realiza el borrado porque existen registros en la tabla vinculada con valores asociados al índice del campo que pretendemos borrar y, de permitir hacerlo, se rompería la integridad referencial ya que quedarían registros *huérfanos* en la tabla vinculada.

Ver script	Borrar en tabla principal

Sin tratamos de modificar un registro de la tabla principal y la modificación afecta al índice que realiza la asociación con la tabla (o tablas) vinculadas se produciría -por las mismas razones y en las mismas circunstancias- un error nº **1217** que impediría la modificación.

Ver script	Modificar DNI en tabla principal

Si -al tratar de borrar o modificar un registro de la tabla principal– no existieran en la tabla (o tablas vinculadas) registros asociados con él, el proceso de modificación se realizaría sin ningún problema y sin dar ningún mensaje de error o advertencia.

Automatización de procesos

Creamos dos tablas idénticas a las anteriores incluyendo algunos datos en ellas.

```
<?
$base="ejemplos";
$tabla1="principal1";
$tabla2="vinculada1";
$c=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $c);
# creación de la tabla principal type InnoDB
$crear="CREATE TABLE IF NOT EXISTS $tabla1 (";
$crear.="DNI CHAR(8) NOT NULL,  ";
$crear.="Nombre VARCHAR (20)  NOT NULL, ";
$crear.="Apellido1 VARCHAR (15)  not null, ";
$crear.="Apellido2 VARCHAR (15)  not null, ";
$crear.=" PRIMARY KEY(DNI) ";
$crear.=")";
$crear.=" Type=InnoDB";
# creamos la tabla principal comprobando el resultado
if(@mysql_query ($crear ,$c)){
        print "La tabla ".$tabla1." ha sido creada<br>";
}else{
        print "No se ha creado ".$tabla1." ha habido un error<br>";
}
# crearemos la tabla vinculada
$crear="CREATE TABLE IF NOT EXISTS $tabla2 (";
$crear.="IDENTIDAD CHAR(8) NOT NULL, ";
$crear.="calle VARCHAR (20), ";
$crear.="poblacion VARCHAR (20), ";
$crear.="distrito VARCHAR(5), ";
#creamos la tabla vinculada las opciones de DELETE y UPDATE
$crear.=" KEY asociador(IDENTIDAD), ";
#establecemos la vinculación de ambos índices
$crear.=" FOREIGN KEY (IDENTIDAD) REFERENCES $tabla1(DNI) ";
$crear.=" ON DELETE CASCADE ";
$crear.=" ON UPDATE CASCADE ";
$crear.=") TYPE = INNODB";
#  creamos (y comprobamos la creación) la tabla vinculada
if(@mysql_query ($crear ,$c)){
```

coincidentes con la clave primaria de la tabla principal.

Para el caso de ON UPDATE las opciones son estas:

ON UPDATE RESTRICT
ON UPDATE CASCADE
ON UPDATE SET NULL

Su comportamiento es idéntico a sus homónimas del caso anterior.

¡Cuidado!

El uso de la opción SET NULL requiere que el campo indicado en FOREIGN KEY esté permita valores nulos. Si está definido con *flag* NOT NULL (como ocurre en los ejemplos que tienes al margen) daría un mensaje de error.

¡Cuidado!

Al incluir ON DELETE y ON UPTADE (si se incluyen ambas) han de hacerse por este mismo orden.
Si se cambiara este orden daría un mensaje de error y no se ejecutarían.

```
        print "La tabla ".$tabla2." ha sido creada<br>";
}else{
        print "No se ha creado ".$tabla2." ha habido un error<br>";
}
# añadimos registros a la tabla principa1
mysql_query("INSERT $tabla1 (DNI,Nombre,Apellido1,Apellido2)
                VALUES ('111111','Robustiano','Iglesias','Pérez')",$c);
mysql_query("INSERT $tabla1 (DNI,Nombre,Apellido1,Apellido2)
                VALUES ('222222','Ambrosio','Morales','Gómez')",$c);
# añadimos registros a la tabla vinculada1
mysql_query("INSERT $tabla2 (IDENTIDAD,calle,poblacion,distrito)
                VALUES ('111111','Calle Asturias,3','Oviedo','33001')",$c)
mysql_query("INSERT $tabla2 (IDENTIDAD,calle,poblacion,distrito)
      VALUES ('111111','Calle Palencia,3','Logroño','78541')",$c);
mysql_query("INSERT $tabla2 (IDENTIDAD,calle,poblacion,distrito)
      VALUES ('222222','Calle Anunciación,3','Algeciras','21541')",$c);
mysql_close();
?>
```

Crear tablas y datos | Ver contenidos de tablas | Ver codigo fuente

Modificar registros en cascada

```
<?
$base="ejemplos";
$tabla="principal1";
$conexion=mysql_connect("localhost","pepe","pepa");
mysql_select_db($base,$conexion);
# modificamos un registro
mysql_query("UPDATE $tabla SET DNI='123456' WHERE DNI='111111'",$conexion)
# borramos un registro
mysql_query("DELETE FROM $tabla WHERE (DNI='222222')",$conexion);
if (mysql_errno($conexion)==0){echo "<h2>Tablas actualizadas</b></H2>";
             }else{
         print "Ha habido un error al actualizar";
        }
mysql_close();
?>
```

Actualizar en cascada | Ver contenido de la tabla

Para que puedas retornar a las condiciones iniciales, desde este enlace podrás borrar las tablas creadas para actualización en cascada. De esta forma podrás volver a crearlas, cuando desees, en las condiciones iniciales.