Definición de tablas

En este ejemplo de tablas vinculadas con integridad relacional vamos a proponer la situación siguiente.

Crearemos una primera tabla (**alumnos**) que va a contener datos personales de un grupo de personas.

Para evitar duplicar un mismo alumno y alumnos sin DNI, vamos a utilizar como índice primario (PRIMARY KEY) el campo DNI (que es único para cada persona).

La condición de que el campo DNI sea PRIMARY KEY nos obliga a definirlo con el *flag* NOT NULL, dado que esta es una condición necesaria para la definición de índices primarios.

Crearemos una segunda tabla (**domicilios**) con los domicilios de cada uno de los alumnos, identificándolos también por su DNI. Para evitar que un mismo alumno pueda tener dos domicilios asignamos a al campo DNI de esta tabla la condición de índice único (UNIQUE).

El índice UNIQUE podríamos haberlo creado también como PRIMARY KEY ya que la única diferencia de comportamiento entre ambos es el hecho que aquel admitiría valores NULOS pero, dado que debemos evitar *domicilios sin alumnos*, insertaremos el *flag* NOT NULL en este campo.

El hecho de utilizar dos tablas no tiene otro sentido que la ejemplificación ya que lo habitual sería que todos los datos estuvieran en una misma tabla.

Vincularemos ambas tablas de modo que no puedan crearse direcciones de alumnos inexistentes y les pondremos la opción de *actualización* y *borrado* en cascada. De esta forma las acciones en la tabla de alumno se reflejarían automáticamente en la tabla de domicilios.

La tercera de las tablas (**evaluaciones**) tiene como finalidad definir las distintas evaluaciones que puede realizarse a cada alumno. Tendrá dos campos. Uno descriptivo (el nombre de la evaluación) y otro identificativo (no nulo y único) que será tratado como PRIMARY KEY para evitar duplicidades y porque, además, va a ser utilizado como

Estructura de tablas y relaciones

Para desarrollar los ejemplos de este capítulo vamos a crear las tablas, cuyas estructuras e interrelaciones que puedes ver en el código fuente siguiente:

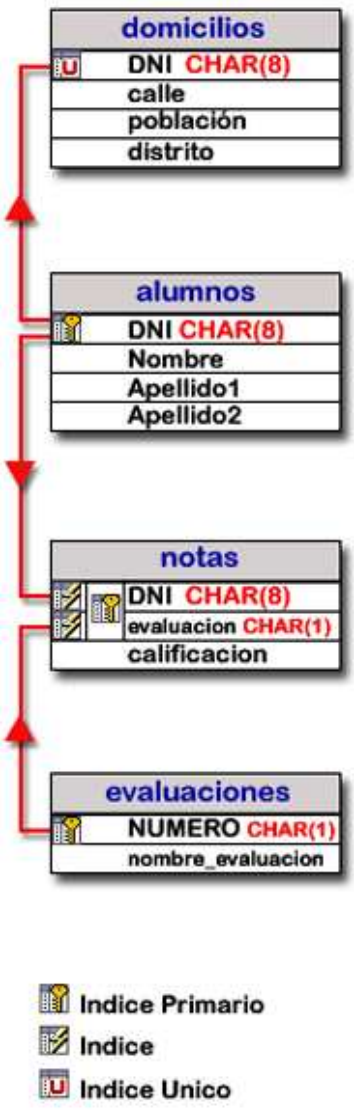
```
<?
$base="ejemplos";
$c=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $c);
################################################
# Creación de la tabla nombres con indice primario DNI
# tabla de nombres con índice primario en DNI
################################################
$crear="CREATE TABLE IF NOT EXISTS alumnos (";
$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.=")";
$crear.=" Type=InnoDB";
if(mysql_query ($crear ,$c)){
        print "tabla <b>nombres</b> creada<BR>";
}else{
        print "ha habido un error al crear la tabla  <b>alumnos</b><BR>";
}
################################################
# Creación de la tabla direcciones
# tabla de nombres con índice único en DNI
# para evitar dos direcciones al mismo alumno
# y clave foránea nombres(DNI)
# para evitar direcciones no asociadas a un alumno
# concreto. Se activa la opción de actualizar
# en cascada y de borrar en cascada
################################################
$crear="CREATE TABLE IF NOT EXISTS domicilios (";
$crear.="DNI CHAR(8) NOT NULL, ";
$crear.="calle VARCHAR (20), ";
$crear.="poblacion VARCHAR (20), ";
$crear.="distrito VARCHAR(5), ";
$crear.=" UNIQUE identidad (DNI), ";
$crear.="FOREIGN KEY (DNI) REFERENCES alumnos(DNI) ";
$crear.="ON DELETE CASCADE ";
$crear.="ON UPDATE CASCADE ";
$crear.=") TYPE = INNODB";
if(mysql_query ($crear ,$c)){
        print "tabla   <b>domicilios</b> creada<br>";
}else{
    print "ha habido un error al crear la tabla <b>domicilios</b><BR>";
}
################################################
# Creación de la tabla nombres con indice primario EVALUACIONES
# tabla de nombres con índice primario en NUMERO
################################################
$crear="CREATE TABLE IF NOT EXISTS evaluaciones (";
$crear.="NUMERO CHAR(1) NOT NULL,  ";
$crear.="nombre_evaluacion VARCHAR (20)  NOT NULL, ";
$crear.=" PRIMARY KEY(NUMERO)";
$crear.=")";
$crear.=" Type=InnoDB";
if(mysql_query ($crear ,$c)){
        print "tabla <b>evaluaciones</b> creada<BR>";
}else{
    print "ha habido un error al crear la tabla <b>evaluaciones</b><BR>";
}
################################################
# Creación de la tabla notas
# indice UNICO para los campos DNI y evaluacion
# con ello se impide calificar dos veces al mismo
```

clave foránea en la tabla **notas**.

La tabla **notas** va a tener tres campos: DNI, nº de evaluación y calificación.

Estableceremos restricciones para **evitar que**:
Podamos calificar a un alumno inexistente.
Podamos calificar una evaluación no incluida entre las previstas.
Podamos poner más a cada alumnos más de una calificación por evaluación.
Creando una índice primario formado por los campos DNI y evaluación evitaremos la última de las situaciones y añadiendo una vinculación con las tablas alumnos y evaluaciones estaremos en condiciones de evitar las dos primeras.

En este gráfico puedes ver un esquema de la definición de estas tablas.



Importación y exportación de datos

Es esta una opción interesante por la posibilidad que ofrece de

```
# alumno en la misma evaluacion
# claves foráneas (DOS)
# el DNI de nombres para evitar calificar a alumnos inexistentes
# el NUMERO de la tabla evaluaciones para evitar calificar
# DOS VECES en una evaluación a un alumno
################################################
$crear="CREATE TABLE IF NOT EXISTS notas (";
$crear.="DNI CHAR(8) NOT NULL, ";
$crear.="evaluacion CHAR (1) NOT NULL, ";
$crear.="calificacion TINYINT (2), ";
/* observa que este indice primario está formado
   por dos campos (DNI y evalucion) y que, como siempre
   en el caso de PRIMARY KEY ambos son de tipo NOT NULL */
$crear.=" PRIMARY KEY  vemaos(DNI,evaluacion), ";
/* Fijate en la secuencia siguiente:
  1º.- Creamos el índice
  2º.- Establecemos la clave foránea
  3º.- Establecemo las condiciones ON DELETE
  4º.- Establecemos las condiciones ON UPDTE
  Es muy importe mantener esta secuencia para evitar
  errores MySQL */
$crear.=" INDEX identico (DNI), ";
$crear.="FOREIGN KEY (DNI) REFERENCES alumnos(DNI) ";
$crear.="ON DELETE CASCADE ";
$crear.="ON UPDATE CASCADE,";
/* Esta tabla tiene dos claves foráneas asociadas a dos tablas
   la anterior definida sobre alumnos como tabla principal
   y esta que incluimos a continuación asociada con evaluaciones
   Como ves repetimos la secuencia descrita anteriormente
   Es importante establecer estas definiciones de una en una
   (tal como ves en este ejemplo) y seguir la secuencia
   comentada anteriormente  */
$crear.=" INDEX evalua (evaluacion),";
$crear.="FOREIGN KEY (evaluacion) REFERENCES evaluaciones(NUMERO) ";
$crear.="ON DELETE CASCADE ";
$crear.="ON UPDATE CASCADE";
$crear.=") TYPE = INNODB";
if(mysql_query ($crear ,$c)){
        print "tabla <b>notas</b> creada <BR>";
}else{
    print "ha habido un error al crear la tabla <b>notas</b><BR>";
        echo mysql_error ($c)."<br>";
    echo mysql_errno ($c);
}

mysql_close();
?>
```

Crear ejemplos tablas vinculadas

¡Cuidado!

Como puedes observar en la imagen de la izquierda, al definir la estructura de las tablas es muy importante prestar atención a que los campos vinculados sean del **mismo tipo y dimensión**.

Observa también que los campos de referencia de los vínculos que se establecen (en las tablas primarias) tienen que ser definidos como PRIMARY KEY y que, por tanto, han de establecerse como no nulos (NOT NULL).

Inserción de datos en tablas

MySQL permite **importar** ficheros externos utilizando la siguiente sintaxis:

LOAD DATA INFILE "*nombre del fichero***' *[REPLACE | IGNORE]***
INTO TABLE *nombre de la tabla*
FIELDS
TERMINATED BY '*indicador de final de campo***'**
ENCLOSED BY '*caracteres delimitadores de campos***'**
LINES

intercambiar datos entre diferentes fuentes y aplicaciones.

Importación de ficheros

MySQL permite insertar en sus tablas los contenidos de ficheros de texto. Para ello utiliza la sentencia que tienes al margen y que detallaremos a continuación.

LOAD DATA INFILE
Es un contenido obligatorio que defina la opción de insertar datos desde un fichero externo.

nombre del fichero
Se incluye inmediatamente después de la anterior, es obligatorio y debe contener (entre comillas) la ruta, el nombre y la extensión del fichero que contiene los datos a insertar.

[REPLACE|IGNORE]
Es opcional. Si se omite se producirá un mensaje de error si el fichero contiene valores iguales a los contenidos en los campos de la tabla que no admiten duplicados. Con la opción **REPLACE** sustituiría los valores existentes en la tabla y con la opción **IGNORE**

INTO TABLE *nombre*
Tiene carácter obligatorio y debe incluir como *nombre* el de la tabla a la que se pretende agregar los registros.

FIELDS
Tiene carácter OPCIONAL y permite incluir especificaciones sobre cuales son los caracteres delimitadores de campos y los que indican el final del campo. Si se omite **FIELDS** no podrán incluirse los **ENCLOSED BY** ni **TERMINATED BY** de campo.

ENCLOSED BY
Permite espeficar (encerrados entre comillas) los caracteres delimitadores de los campos. Estos caracteres deberán encontrarse en el fichero original **al principio** y **al final** de los contenidos de cada uno de los campos (por ejemplo, si el carácter fueran *comillas* los campos deberían aparecer así en el fichero a importar algo como esto: **"32.45"**). Si se omite esta especificación (o se omite FIELDS) se entenderá que los campos **no tienen caracteres delimitadores**.

Cuando se incluyen como delimitadores de campo las comillas (dobles o sencillas) es necesario utilizar una sintaxis como esta: '\'' ó "\"" de forma que no quepa la ambigüedad de si se trata de un carácter o de las comillas de cierre de una cadena previamente abierta.

TERMINATED BY
Se comporta de forma similar al anterior.
Permite qué caracteres son usados en el fichero original como

STARTING BY '*caracteres indicadores de comienzo de registro*'
TERMINATED BY '*caracteres indicadores del final de registro*'

En este ejemplo pueder un caso práctico de inserción de datos en las tablas creadas anteriormente.

```
<?
$base="ejemplos";
$c=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $c);
# hemos creado un fichero de texto (datos_alumnos.txt)
# que contiene datos de algunos alumnos. Los diferentes
# campos están entre comillas y separados unos de otros
# mediante un punto y coma.
# Cada uno de los registros comienza por un asterios
# y los registros están separados por un salto de líneas (\r\n)
# Incluimos estás especificaciones en la sentencia de inserción
$query="LOAD DATA INFILE ";
$query.="'".$_SERVER['DOCUMENT_ROOT'].'/cursophp/datos_alumnos.txt'."'";
$query.= " REPLACE INTO TABLE alumnos";
$query.= " FIELDS ENCLOSED BY '\"' TERMINATED BY ';' ";
$query.=" LINES STARTING BY '*' TERMINATED BY '\r\n' ";
if(mysql_query($query,$c)){
   print "Datos de alumnos cargados<br>";
                          }else{
   print mysql_error ($c)."<br>".mysql_errno ($c);
}
#  Para esta tabla usaremos el fichero datos_evaluaciones.txt
# Los diferentes
# campos están entre comillas y separados unos de otros
# mediante una coma.
# Cada uno de los registros comienza por un espacio
# y los registros están separados por un salto de líneas (\r\n)
# Incluimos estás especificaciones en la sentencia de inserción
$query="LOAD DATA INFILE ";
$query.="'".$_SERVER['DOCUMENT_ROOT'].'/cursophp/datos_evaluaciones.txt'."
$query.= " REPLACE INTO TABLE evaluaciones";
$query.= " FIELDS ENCLOSED BY '\'' TERMINATED BY ',' ";
$query.=" LINES STARTING BY ' ' TERMINATED BY '\r\n' ";
 if(mysql_query($query,$c)){
   print "Datos de evaluaciones cargados<br>";
                          }else{
   print mysql_error ($c)."<br>".mysql_errno ($c);
}
/* En este caso no incluimos especificación alguna.
   Bajo este suspuesto MySQL interpreta los valores por defecto
   que son: los campos no van encerrados, las líneas no tienen
   ningún carácter indicador de comienzo, los campos están separados
   mediante tabulaciones (carácter de escape \t) y el final de línea
   está señalado por un caracter de nueva línea (\n) */
$query="LOAD DATA INFILE ";
$query.="'".$_SERVER['DOCUMENT_ROOT'].'/cursophp/datos_notas.txt'."'";
$query.= " IGNORE INTO TABLE notas";
if(mysql_query($query,$c)){
   print "Datos de notas cargados<br>";
                          }else{
  print mysql_error ($c)."<br>".mysql_errno ($c);
}
/* Se comporta como los casos anteriores con distincos caracteres
   para los diferentes eventos, tal como puedes ver en el código */
$query="LOAD DATA INFILE ";
 $query.="'".$_SERVER['DOCUMENT_ROOT'].'/cursophp/datos_domicilios.txt'."'
 $query.= " IGNORE INTO TABLE domicilios";
 $query.= " FIELDS ENCLOSED BY '|' TERMINATED BY '*' ";
 $query.=" LINES STARTING BY '#' TERMINATED BY '}' ";
if(mysql_query($query,$c)){
   print "Datos de domicilios cargados
";
                          }else{
 print mysql_error ($c)."<br>".mysql_errno ($c);
}  mysql_close();
?>
```

Cargar datos	Consultar tablas

Guardar datos en ficheros

separadores de campos (indicador de final de campo). Si se omite, se interpretará con tal el carácter *tabulador* (\t).

El uso de esta opción no requiere que se especifique previamente **ENCLOSED BY** pero si necesita que se haya incluido **FIELDS**. **LINES**
Si el fichero de datos contiene caracteres (distintos de los valores por defecto) para señalar el comienzo de un registro, el final del mismo o ambos, debe incluirse este parámetro y, después de él, las especificaciones de esos valores correspondientes a:

STARTING BY
Permite especificar una carácter como indicador de comienzo de un registro. Si se omite o se especifica como " se interpretará que no hay ningún carácter que señale en comienzo de línea.

TERMINATED BY
Es el indicador del **final de un registro**. Si se omite será considerado como un *salto de línea* (\n).

Exportación de ficheros

Se comporta de forma similar al supuesto anterior. Utiliza la sintaxis siguiente:

SELECT * INTO OUTFILE
Inicia la consulta de los campos especificados después de **SELECT** (si se indica * realiza la consulta sobre todos los campos y por el orden en el que fue creada la tabla) y redirige la salida a un fichero.

nombre del fichero
Es la ruta, nombre y extensión del fichero en el que serán almacenados los resultados de la consulta.

FIELDS
ENCLOSED BY
TERMINATED BY
LINES
STARTING BY
TERMINATED BY
Igual que ocurría en el caso de importación de datos, estos parámetros son opcionales. Si no se especifican se incluirán los valores por defecto.

FROM *nombre*
Su inclusión tiene carácter obligatorio. El valor de *nombre* ha de ser el de la tabla sobre la que se realiza la consulta.

¡Cuidado!

Al importar ficheros habrá de utilizarse el mismo formato con el que fueron creados tanto FIELDS como LINES.

MySQL permite **los contenidos de sus tablas** a ficheros de texto. Para ello utiliza la siguiente sintaxis:

SELECT * INTO OUTFILE "*nombre del fichero*'
FIELDS
TERMINATED BY '*indicador de final de campo*'
ENCLOSED BY '*caracteres delimitadores de campos*'
LINES
STARTING BY '*caracteres indicadores de comienzo de registro*'
TERMINATED BY '*caracteres indicadores del final de registro*'
FROM *nombre de la tabla*

```
<?
$base="ejemplos";
$c=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $c);
$query="SELECT * INTO OUTFILE ";
$query.="'".$_SERVER['DOCUMENT_ROOT'].'/cursophp/alumnos.txt'."'";
$query.= " FIELDS ENCLOSED BY '\"' TERMINATED BY ';' ";
$query.=" LINES STARTING BY '*' TERMINATED BY '\r\n' ";
$query.=" FROM alumnos";
if(mysql_query($query,$c)){
    print "fichero  alumnos.txt creado<br>";
}else{
    print mysql_error ($c)."<br>". mysql_errno ($c);
}
$query="SELECT * INTO OUTFILE ";
$query.="'".$_SERVER['DOCUMENT_ROOT'].'/cursophp/domicilios.txt'."'";
$query.= " FIELDS ENCLOSED BY '|' TERMINATED BY '*;' ";
$query.=" LINES STARTING BY '#' TERMINATED BY '}' ";
$query.=" FROM domicilios";
if(mysql_query ($query,$c)){
    print "fichero domicilios.txt creado<br>";
}else{
    print mysql_error ($c)."<br>".mysql_errno ($c);
}
$query="SELECT * INTO OUTFILE ";
$query.="'".$_SERVER['DOCUMENT_ROOT'].'/cursophp/notas.txt'."'";
$query.=" FROM notas";
if(mysql_query ($query,$c)){
    print "fichero notas.txt creado<br>";
}else{
    print mysql_error ($c)."<br>".mysql_errno ($c);
}
$query="SELECT * INTO OUTFILE ";
$query.="'".$_SERVER['DOCUMENT_ROOT'].'/cursophp/evaluaciones.txt'."'";
$query.= " FIELDS ENCLOSED BY '\'' TERMINATED BY ',' ";
$query.=" LINES STARTING BY ' ' TERMINATED BY '\r\n' ";
$query.=" FROM evaluaciones";
if(mysql_query ($query,$c)){
    print "fichero evaluaciones.txt creado<br>";
}else{
    print mysql_error ($c)."<r>".mysql_errno ($c);
}
mysql_close();
?>
```

Crear ficheros de datos

¡Cuidado!

Al exportar ficheros en entornos Windows, si se pretende que en el fichero de texto aparezca un salto de línea no basta con utilizar la opción por defecto de **LINES TERMINATED BY '\n'** sino **LINES TERMINATED BY '\r\n'** (salto de línea y retorno) que son los caracteres que necesita Windows para producir ese efecto.
Habrá de seguirse este mismo criterio cuando se trata de importar datos desde un fichero de texto.

Consultas de unión (JOIN)

Consultas usando JOIN

La claúsula JOIN es opción aplicable a consultas en tablas que tiene diversas opciones de uso. Iremos viéndolas de una en una. Todas ellas han de ir incluidas como parámetros de una consulta. Por tanto han de ir precedidas de:

SELECT *
o de
SELECT *nom_tab.nom_cam,..*
donde *nom_tab* es un nombre de tabla y *nom_camp* es el nombre del campo de esa tabla que pretendemos visualizar para esa consulta. Esta sintaxis es idéntica a la ya comentada en páginas anteriores cuando tratábamos de consultas en varias tablas.

Ahora veremos las diferentes posibilidades de uso de **JOIN**

FROM *tbl1* **JOIN** *tbl2*
Suele definirse como el *producto cartesiano* de los elementos de la primera tabla (*tbl1*) por lo de la segunda (*tbl2*).

Dicho de una forma más vulgar, esta consulta devuelve con resultado una lista de cada uno de los registros de los registros de la primera tabla asociados sucesivamente con todos los correspondientes a la segunda. Es decir, aparecerá una línea conteniendo el primer registro de la primera tabla seguido del primero de la segunda. A continuación ese mismo registro de la primera tabla acompañado del segundo de la segunda tabla, y así, sucesivamente hasta acabar los registros de esa segunda tabla. En ese momento, repite el proceso con el segundo registro de la primera tabla y, nuevamente, todos los de la segunda. Así, sucesivamente, hasta llegar al último registro de la primera tabla asociado con el último de la segunda.

En total, devolverá un número de líneas igual al resultado de multiplicar el número de registros de la primera tabla por los de la segunda.

FROM *tbl2* **JOIN** *tbl1*
Si permutamos la posición de las tablas, tal como indicamos aquí, obtendremos el mismo resultado que en el caso anterior pero, como es lógico pensar, con una ordenación diferente de los resultados.

FROM *tbl2* **JOIN** *tbl1* **ON** *cond*
El parámetro ON permite añadir una condición (*cond*)a la consulta de unión. Su comportamiento es idéntico al de WHERE en las consultas ya estudiadas y permite el uso de las mismas procedimientos de establecimiento de condiciones que aquel operador.

```
<?
$base="ejemplos";
$c=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $c);
# vamos a crear un array con las diferente consultas
# posteriormente lo leeremos y la ejecutaremos secuencialmente
/* Devuelve todos los campos de ambas tablas.
  Cada registro de alumnos es asociado con todos los de domicilios*/
$query[]="SELECT * FROM alumnos JOIN domicilios";
/* Devuelve todos los campos de ambas tablas. Cada registro de domicilios
   es asociado con todos los de alumnos */
$query[]="SELECT * FROM domicilios JOIN alumnos";
/* Devuelve todos los campos de los registros de ambas tablas
en los que coinciden los numeros del DNI*/
$query[]="SELECT * FROM alumnos JOIN domicilios
                        ON domicilios.DNI=alumnos.DNI";
/* Idéntica a la anterior. Solo se diferencia en que ahora
se visualizan antes los campos domicilios*/
$query[]="SELECT * FROM domicilios JOIN alumnos
                        ON domicilios.DNI=alumnos.DNI";
/* devuelve cada uno de los registro de la tabla alumnos. Si existe
   un domicilio con igual DNI lo insertará. Si no existiera
   insertará valores nulos en esos campos
$query[]="SELECT * FROM alumnos LEFT JOIN domicilios
                        ON domicilios.DNI=alumnos.DNI";
/* Se comporta de forma idéntica al anterior.
   Ahora insertará todos los registros  de domicilios
   y los alumnos coincidentes o en su defecto campos nulos.*/
$query[]="SELECT * FROM domicilios LEFT JOIN alumnos
                        ON domicilios.DNI=alumnos.DNI";
/* Al utilizar RIGHT será todos los registros de la tabla de la derecha
(domicilios) los que aparezcan junto con las coincidencias o
junto a campos nulos. Aparecerán primero los campos de alumnos
y detrá los de domicilios*/
$query[]="SELECT * FROM alumnos RIGHT JOIN domicilios
          ON (domicilios.DNI=alumnos.DNI AND alumnos.Nombre LIKE 'A%')";
/* Consulta de nombre, apellido y localidad de todos los alumnos
cuyo nombre empieza por A */
$query[]="SELECT alumnos.Nombre, alumnos.Apellido1,alumnos.Apellido2,
         domicilios.poblacion FROM alumnos JOIN domicilios
                        ON (domicilios.DNI=alumnos.DNI
                        AND alumnos.Nombre LIKE 'A%')";
# una consulta resumen de nos permitirá visualizar una lista con nombre
# y apellidos de alumnos su dirección y localidad del domicilio
# el nombre de la evaluación y su calificación.
# Si no hay datos de población insertará ---- en vez del valor nulo
# y si no hay calificación en una evaluación aparecerá N.P.
# La consulta aparecerá agrupada por evaluaciones
/* iniciamos el select especificando los campos de las diferentes
   tablas que prentendemos visualizar
$q="(SELECT alumnos.Nombre,alumnos.Apellido1,";
$q.=" alumnos.Apellido2,domicilios.calle,";
# al incluir IFNULL visualizaremos ---- en los campos cuyo resultado
# sea nulo
$q.=" IFNULL(domicilios.poblacion,'----'),";
$q.=" evaluaciones.nombre_evaluacion,";
# con este IFNULL aparecerá N.P. en las evaluaciones no calificadas.
$q.=" IFNULL(notas.calificacion,'N.P.')";
# especificamos el primer JOIN con el que tendremos como resultado una lis
# de todos los alumnos con sus direcciones correspondientes
# por efecto de la clausula ON.
# Al poner LEFT se incluirían los alumnos que no tuvieran
# su dirección registrada en la tabla de direccione
$q.=" FROM (alumnos LEFT JOIN domicilios";
$q.="        ON alumnos.DNI=domicilios.DNI)";
# al unir por la izquierda con notas tendríamos todos los resultados
# del JOIN anterior asociados con con todas sus calificaciones
# por efecto de la claúsula ON
$q.="         LEFT JOIN notas ON notas.DNI=alumnos.DNI";
# al añadir esta nueva unión por la DERECHA con la tabla evaluaciones
# se asociaría cada uno de los resultados de las uniones anteriores
# con todos los campos de la tabla evaluaciones con lo que resultaría
# una lista de todos los alumnos con todas las calificaciones
# incluyendo un campo en blanco (sería sustituido por N.P:)
# en aquellas que no tuvieran calificación registrada
$q.="         RIGHT JOIN evaluaciones";
$q.="         ON evaluaciones.NUMERO=notas.evaluacion";
/* la clausula WHERE nos permite restringir los resultados a los valores
```

FROM *tbl1* **LEFT JOIN** *tbl2* **ON** *cond*
Cuando se incluye la cláusula **LEFT** delante de JOIN el resultado de la consulta es el siguiente:
– Devolvería cada uno los registros de la tabla especificada a la izquierda de LEFT JOIN -sin considerar las restricciones que puedan haberse establecido en las claúsulas ON para los valores de esa tabla– asociándolos con aquellos de la otra tabla que cumplan las condiciones establecidas en la claúsula ON. Si ningún registro de la segunda tabla cumpliera la condición devolvería valores nulos.

FROM *tbl1* **RIGHT JOIN** *tbl2* **ON** *cond*
Se comporta de forma similar al anterior. Ahora los posibles valores nulos serán asignados a la tabla indicada a la izquierde de RIGHT JOIN y se visualizarían todos los registros de la tabla indicada a la derecha.

JOIN múltiples

Tal como puedes observar en el ejemplo, es perfectamente factible utilizar conjuntamente varios JOIN, LEFT JOIN y RIGHT JOIN. Las diferentes uniones irán ejecutándose de izquierda a derecha (según el orden en el que estén incluidos en la sentencia) y el resultado del primero será utilizado para la segunda unión y así sucesivamente.

En cualquier caso, es posible alterar ese orden de ejecución estableciendo otras prioridades mediante paréntesis.

UNION de consultas

MySQL permite juntar en una sola salida los resultados de varias consultas. La sintaxis es la siguiente:

(SELECT ...)
UNION ALL
(SELECT ...)
UNION ALL
(SELECT ...)

Cada uno de los SELECT ha de ir encerrado entre paréntesis.

```
 correspondientes únicamente a la evaluación número 1*/
$q.="              WHERE evaluaciones.NUMERO=1)";
# cerramos la consulta anterior con el paréntesis. Observa que lo
# hemos abierto delante del SELECT e insertamos UNION ALL
# para que el resultado de la consulta anterior aparezca
# seguido del correspondiente a la incluida después de UNION ALL
$q.=" UNION ALL";
#iniciamos (también con paréntesis) la segunda consulta
# que será identica a la anterior salvo el WHERE
# será modificado para extraer datos de la evaluación n°2
$q.="(SELECT alumnos.Nombre,alumnos.Apellido1,";
$q.=" alumnos.Apellido2,domicilios.calle,";
$q.=" IFNULL(domicilios.poblacion,'----'),";
$q.=" evaluaciones.nombre_evaluacion,";
$q.=" IFNULL(notas.calificacion,'N.P.')";
$q.=" FROM (alumnos LEFT JOIN domicilios";
$q.=" ON alumnos.DNI=domicilios.DNI)";
$q.=" LEFT JOIN notas ON notas.DNI=alumnos.DNI";
$q.=" RIGHT JOIN evaluaciones";
$q.=" ON evaluaciones.NUMERO=notas.evaluacion";
$q.=" WHERE evaluaciones.NUMERO=2)";
# hemos cerrado el parentesis de la consulta anterior
# e incluimos un nuevo UNION ALL para consultar los datos
# correspondientes a la tercera evaluación
$q.=" UNION ALL";
$q.="(SELECT alumnos.Nombre,alumnos.Apellido1,";
$q.=" alumnos.Apellido2,domicilios.calle,";
$q.=" IFNULL(domicilios.poblacion,'----'),";
$q.=" evaluaciones.nombre_evaluacion,";
$q.=" IFNULL(notas.calificacion,'N.P.')";
$q.=" FROM (alumnos LEFT JOIN domicilios";
$q.=" ON alumnos.DNI=domicilios.DNI)";
$q.=" LEFT JOIN notas ON notas.DNI=alumnos.DNI";
$q.=" RIGHT JOIN evaluaciones";
$q.=" ON evaluaciones.NUMERO=notas.evaluacion";
$q.=" WHERE evaluaciones.NUMERO=3)";
# incluimos la variable $q en el array de consultas
$query[]=$q;
# leemos el array y visualizamos el resultado
# cada consulta a traves de la llamada a la funcion visualiza
# a la que pasamos el resultado de la consulta
# y la cadena que contiene las sentencias de dicha consulta
foreach($query as $v){
    visualiza(mysql_query($v,$c),$v);
}
function visualiza($resultado,$query){
           PRINT "<BR><BR><i>Resultado de la sentencia:</i><br>";
           print "<b><font color=#ff0000>";
           print $query."</font></b><br><br>";
       PRINT "<table align=center border=2>";
          while ($registro = mysql_fetch_row($resultado)){
               echo "<tr>";
               foreach($registro  as $valor){
                     echo "<td>",$valor,"</td>";
               }
         }
       echo "</table><br>";
}
?>
```

Ejecutar script