TEMA 6: DISEÑO FÍSICO DE BASES DE DATOS

1. El lenguaje de definición de datos (DDL).

El lenguaje de definición de datos proporciona los medios para definir los elementos que integran la base de datos (tablas, vistas, índices, etc.). Más concretamente, los DDL dan la posibilidad de crear estos elementos (mediante sentencias del tipo CREATE), modificar su definición (mediante sentencias del tipo ALTER) y eliminarlos (mediante sentencias del tipo DROP). Antes de adentrarnos en el estudio de estas sentencias, es conveniente estudiar los tipos de datos permitidos en MySQL para atributos de las tablas.

2. Tipos de datos del lenguaje.

Se van a presentar a continuación los tipos de datos más utilizados que se pueden asignar a los atributos de tablas en MySQL. En los tipos que se indican a continuación lo que se especifica entre corchetes es opcional. Pues bien, los tipos de datos los vamos a clasificar en: numéricos, de fecha y hora y de cadenas de caracteres

1) Tipos numéricos: Permiten almacenar números, los cuales pueden ser enteros (sin parte decimal) o reales (con parte decimal). En las explicaciones que se dan a continuación la N indica el número total de dígitos y la D, el número de dígitos después de la coma. Los tipos de datos numéricos más empleados son los siguientes:
   a) TINYINT [(N)]: Sirve para almacenar números enteros muy pequeños. El rango es entre -128 y 127 con signo y entre 0 y 255 sin signo.
   b) BOOL o BOOLEAN: Es equivalente a TINYINT(1). Un valor 0 se considera falso y un valor diferente de 0, verdadero.
   c) SMALLINT [(N)]: Sirve para almacenar números enteros pequeños. El rango es entre -32.768 y 32.767 con signo y entre 0 y 65535 sin signo.
   d) MEDIUMINT [(N)]: Sirve para almacenar números enteros de tamaño medio. El rango es entre -8.388.608 y 8.388.607 con signo y entre 0 y 16.777.215 sin signo.
   e) INT [(N)] o INTEGER[(N)]: Sirve para almacenar números enteros de tamaño normal. El rango es entre -2.147.483.648 y 2.147.483.647 con signo y entre 0 y 4.294.967.295 sin signo.

f) BIGINT [(N)]: Sirve para almacenar números enteros grandes. El rango es entre -9.223.372.036.854.775.808 y 9.223.372.036.854.775.807 con signo y entre 0 y 18.446.744.073.709.551.615 sin signo.

g) FLOAT [(N,D)]: Sirve para almacenar números con decimales de precisión simple. Los valores permitidos van desde -3,402823466 . 10^{38} hasta 3,402823466 . 10^{38}.

h) DOUBLE [(N,D)]: Sirve para almacenar números con decimales de precisión doble. Los valores permitidos van desde -1.7976931348623157 . 10^{308} hasta 1.7976931348623157 . 10^{308}.

2) Tipos de fecha y hora: Permiten almacenar fechas y/o horas. Los tipos de fecha y hora más empleados son los siguientes:

a) DATE: Permite almacenar una fecha en el rango de '1001-01-01' a '9999-12-31'. Como se puede observar, las fechas se almacenan en el formato 'AAAA-MM-DD'.

b) DATETIME: Permite almacenar una fecha y una hora. Los valores permitidos oscilan entre '1001-01-01 00:00:00' a '9999-12-31 23:59:59'. Como se puede observar, estos datos se almacenan siguiendo el formato 'AAAA-MM-DD HH:MM:SS'.

c) TIMESTAMP: Permite almacenar una fecha y una hora. Los valores permitidos oscilan entre '1970-01-01 00:00:01' a '2038-01-19 03:14:07'. Se puede asignar como valor por defecto a un campo de este tipo el valor CURRENT_TIMESTAMP, que hace referencia a la fecha y hora actual del sistema.

d) TIME: Permite almacenar una hora en el rango de '-838:59:59' a '838:59:59'.

e) YEAR [(4)]: Permite almacenar un año en formato de cuatro dígitos. Admite valores entre 1901 y 2155 y también el 0000.

3) Tipos de cadenas de caracteres: Permiten almacenar cadenas de caracteres, que incluyen cualquier carácter incluido dentro del conjunto de caracteres correspondiente a la tabla. Los tipos de cadenas de caracteres más utilizados son los que se indican a continuación:

a) CHAR [(M)]: Sirve para almacenar cadenas de caracteres de longitud fija, esto es, cadenas que siempre ocupan el número de caracteres especificado en M. Si no se especifica M, la longitud por defecto es 1. Por ello, el tipo CHAR es sinónimo de CHAR(1). Si la cadena que se asigna a un dato con tipo de dato CHAR(M) tiene una longitud menor que M, se rellenará con espacios en blanco a la derecha hasta alcanzar la longitud M. El rango de M es de 0 a 255 caracteres.

b) VARCHAR(M): Permite almacenar cadenas de caracteres de longitud variable, siendo la longitud máxima permitida M. El rango de M va desde 0 hasta 65.535.

c) BLOB[(M)]: Permite almacenar objetos binarios de longitud variable, siendo la longitud máxima permitida en bytes la especificada en M. El rango de M es de 1 a 65.535 bytes. Para los atributos con tipo BLOB no es posible especificar valor por defecto.
d) TINYBLOB[(M)]: Permite almacenar objetos binarios pequeños de longitud variable, siendo la longitud máxima permitida en bytes la especificada en M. El rango de M es de 1 a 255 bytes.
e) MEDIUMBLOB[(M)]: Permite almacenar objetos binarios de tamaño medio y longitud variable, siendo la longitud máxima permitida en bytes la especificada en M. El rango de M es de 1 a 16.777.215 bytes.
f) LONGBLOB[(M)]: Permite almacenar objetos binarios grandes de longitud variable, siendo la longitud máxima permitida en bytes la especificada en M. El rango de M es de 4 GB (4.294.967.295 bytes).
g) TEXT[(M)]: Permite almacenar cadenas de caracteres con una longitud máxima de 65.535 caracteres. La longitud máxima deseada se puede especificar en M. Para los atributos con tipo TEXT no es posible especificar valor por defecto.
h) TINYTEXT[(M)]: Permite almacenar cadenas de caracteres con una longitud máxima de 255 caracteres. La longitud máxima deseada se puede especificar en M.
i) MEDIUMTEXT[(M)]: Permite almacenar cadenas de caracteres con una longitud máxima de 16.777.215 caracteres. La longitud máxima deseada se puede especificar en M.
j) LONGTEXT [(M)]: Permite almacenar cadenas de caracteres con una longitud máxima de 4.294.967.295 caracteres. La longitud máxima deseada se puede especificar en M.
k) ENUM (‘valor1’, ‘valor2’, …): Permite almacenar solo uno de los valores especificados como ‘valor1’, ‘valor2’, etc. El número máximo de valores que se pueden especificar es 65535.
l) SET (‘valor1’, ‘valor2’, …): Permite almacenar cero, uno o más valores de los especificados como ‘valor1’, ‘valor2’, etc. El número máximo de valores que se pueden especificar es 64.

3. Creación, modificación y borrado de bases de datos.

Antes de poder crear bases de datos, hemos de instalar en nuestro ordenador un SGBD en concreto. En este caso vamos a optar por un SGBD libre, como es MySQL. Para su instalación vamos a hacer uso de la aplicación XAMPP, que incorpora MySQL y permite trabajar con bases de datos MySQL mediante un entorno web llamado phpMyAdmin.

Una vez activados los servicios Apache y MySQL, tenemos dos opciones para trabajar con MySQL:

- Usar la interfaz gráfica que proporciona phpMyAdmin. Para hacer uso de esta interfaz gráfica, abriremos nuestro navegador y escribiremos en la barra de direcciones http://localhost:8088/phpmyadmin/. Accederemos a una página como la de la figura 1.

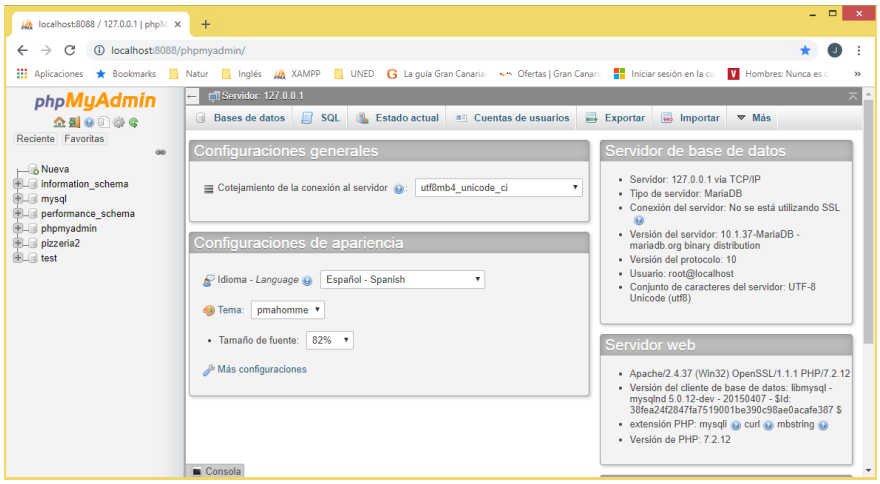


Figura 1: phpMyAdmin.

En la parte izquierda de la pantalla nos aparecen los nombres de las bases de datos existentes. Para escribir órdenes SQL se debe hacer clic en la pestaña que aparece en la parte superior cuyo título es precisamente SQL, mostrándose una pantalla como la de la figura 2. En este pantalla se deben escribir las órdenes SQL en el cuadro de texto con fondo blanco y tras escribir la orden correspondiente hacer clic en el botón *Continuar* para la ejecución de la orden correspondiente.

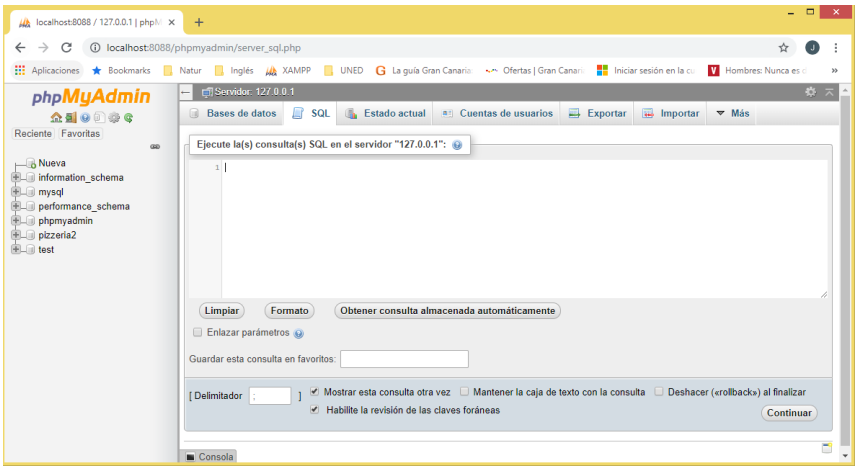


Figura 2: Pantalla para introducción de órdenes SQL.

- Usar la línea de comandos de MySQL. Para ello, accederemos a la línea de comandos haciendo clic en Inicio – Todos los programas – Accesorios – Símbolo del sistema. Nos dirigiremos a la carpeta c:\xampp\mysql\bin para acceder a la línea de comandos de MySQL. Para ello escribiremos cd c:\xampp\mysql\bin. Al instalar XAMPP se ha creado un usuario root sin contraseña y para acceder a la línea de comandos de MySQL escribiremos el comando mysql –u root –p (figura 3). Con esto indicamos que deseamos acceder a la línea de comandos MySQL con el usuario root y la contraseña indicada. Como este usuario no tiene asignada contraseña después de *Enter password:* teclearemos Intro y saldrá el *prompt* `MariaDB [(none)]>` para poder introducir comandos MySQL.

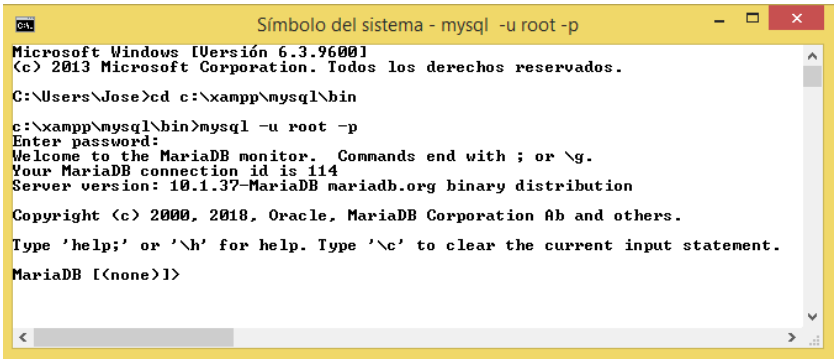


Figura 3: Línea de comandos para MySQL.

Para ejecutar órdenes SQL simplemente habrá que escribirlas tras el prompt `MariaDB [(none)]>` y tras escribir la orden, escribir el símbolo punto y coma (;) y pulsar Intro. Una misma orden puede abarcar varias líneas. Para pasar de una línea a la siguiente dentro de la misma orden se deberá pulsar la tecla Intro, y para indicar que la orden ha finalizado se deberá escribir punto y coma (;) y pulsar Intro.

Para crear bases de datos se hace uso de la sentencia CREATE DATABASE o CREATE SCHEMA. En los formatos de las instrucciones que se van a mostrar a partir de ahora, lo que se especifica entre corchetes ([]) es opcional y la barra vertical (|) indica que se debe escribir una de las opciones que se indican.

El formato de la instrucción CREATE DATABASE es el siguiente:

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] nombre _BD
[especificación_creación]…

Especificación_creación:

[DEFAULT] CHARACTER SET [=] nombre_conjunto_caracteres
| [DEFAULT] COLLATE [=] nombre_cotejamiento

Como se puede observar, se debe indicar después de la palabra CREATE, la palabra DATABASE o bien SCHEMA y después el nombre que se desea dar a la base de datos. Este nombre puede contener cualquier carácter válido para crear una carpeta excepto "/", "\" o ".". No obstante, no es aconsejable escribir caracteres especiales ni espacios en blanco.

Si ya existe en el sistema una base de datos con el nombre indicado, se mostrará un mensaje de error. Para evitar esto, se puede incluir antes del nombre de la base de datos la cláusula IF NOT EXISTS, de tal forma que en este caso si no existe ninguna base de datos con el nombre indicado, se crea; en caso contrario, no se crea, pero no se muestra ningún mensaje de error.

Además, se puede indicar opcionalmente el conjunto de caracteres por defecto que se desea para la base de datos y el cotejamiento correspondiente.

El conjunto de caracteres hace referencia a todos los símbolos que va a ser posible almacenar en atributos de las tablas de la base de datos. Para visualizar los conjuntos de caracteres disponibles, se puede ejecutar el siguiente comando, cuyo resultado también se muestra:

```
MariaDB [(none)]> show character set;
+----------+-----------------------------+---------------------+--------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+--------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |      2 |
| dec8     | DEC West European           | dec8_swedish_ci     |      1 |
| cp850    | DOS West European           | cp850_general_ci    |      1 |
| hp8      | HP West European            | hp8_english_ci      |      1 |
| koi8r    | KOI8-R Relcom Russian       | koi8r_general_ci    |      1 |
| latin1   | cp1252 West European        | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |      1 |
| swe7     | 7bit Swedish                | swe7_swedish_ci     |      1 |
```

```
| ascii    | US ASCII                    | ascii_general_ci    |      1 |
| ujis     | EUC-JP Japanese             | ujis_japanese_ci    |      3 |
| sjis     | Shift-JIS Japanese          | sjis_japanese_ci    |      2 |
| hebrew   | ISO 8859-8 Hebrew           | hebrew_general_ci   |      1 |
| tis620   | TIS620 Thai                 | tis620_thai_ci      |      1 |
| euckr    | EUC-KR Korean               | euckr_korean_ci     |      2 |
| koi8u    | KOI8-U Ukrainian            | koi8u_general_ci    |      1 |
| gb2312   | GB2312 Simplified Chinese   | gb2312_chinese_ci   |      2 |
| greek    | ISO 8859-7 Greek            | greek_general_ci    |      1 |
| cp1250   | Windows Central European    | cp1250_general_ci   |      1 |
| gbk      | GBK Simplified Chinese      | gbk_chinese_ci      |      2 |
| latin5   | ISO 8859-9 Turkish          | latin5_turkish_ci   |      1 |
| armscii8 | ARMSCII-8 Armenian          | armscii8_general_ci |      1 |
| utf8     | UTF-8 Unicode               | utf8_general_ci     |      3 |
| ucs2     | UCS-2 Unicode               | ucs2_general_ci     |      2 |
| cp866    | DOS Russian                 | cp866_general_ci    |      1 |
| keybcs2  | DOS Kamenicky Czech-Slovak  | keybcs2_general_ci  |      1 |
| macce    | Mac Central European        | macce_general_ci    |      1 |
| macroman | Mac West European           | macroman_general_ci |      1 |
| cp852    | DOS Central European        | cp852_general_ci    |      1 |
| latin7   | ISO 8859-13 Baltic          | latin7_general_ci   |      1 |
| utf8mb4  | UTF-8 Unicode               | utf8mb4_general_ci  |      4 |
| cp1251   | Windows Cyrillic            | cp1251_general_ci   |      1 |
| utf16    | UTF-16 Unicode              | utf16_general_ci    |      4 |
| utf16le  | UTF-16LE Unicode            | utf16le_general_ci  |      4 |
| cp1256   | Windows Arabic              | cp1256_general_ci   |      1 |
| cp1257   | Windows Baltic              | cp1257_general_ci   |      1 |
| utf32    | UTF-32 Unicode              | utf32_general_ci    |      4 |
| binary   | Binary pseudo charset       | binary              |      1 |
| geostd8  | GEOSTD8 Georgian            | geostd8_general_ci  |      1 |
| cp932    | SJIS for Windows Japanese   | cp932_japanese_ci   |      2 |
| eucjpms  | UJIS for Windows Japanese   | eucjpms_japanese_ci |      3 |
+----------+-----------------------------+---------------------+--------+
40 rows in set (0.00 sec)
```

Cualquier conjunto de caracteres tiene como mínimo un cotejamiento, aunque puede tener varios. Un cotejamiento es un conjunto de reglas que permite comparar caracteres incluidos dentro de un conjunto de caracteres. Tengamos en cuenta que en un conjunto de caracteres a cada carácter se le asigna un código, como por ejemplo el código ASCII correspondiente. A la hora de comparar dos caracteres y saber cuál es menor, por ejemplo, se comparan sus códigos. Supongamos que por ejemplo la letra “A” tiene asignado el código 0 y la letra “B” el código 1. De esta manera podremos decir que la letra A es “menor” que la letra B porque su código lo es.

Para listar los cotejamientos para un conjunto de caracteres, se puede usar el comando SHOW COLLATION. Por ejemplo, de la siguiente manera se obtienen los cotejamientos para el conjunto de caracteres *utf8mb4*.

```
MariaDB [(none)]> show collation like 'utf8mb4%';
+-----------------------------+---------+-----+---------+----------+---------+
| Collation                   | Charset | Id  | Default | Compiled | Sortlen |
+-----------------------------+---------+-----+---------+----------+---------+
| utf8mb4_general_ci          | utf8mb4 |  45 | Yes     | Yes      |       1 |
| utf8mb4_bin                 | utf8mb4 |  46 |         | Yes      |       1 |
```

```
| utf8mb4_unicode_ci           | utf8mb4 | 224 |         | Yes      |       8 |
| utf8mb4_icelandic_ci         | utf8mb4 | 225 |         | Yes      |       8 |
| utf8mb4_latvian_ci           | utf8mb4 | 226 |         | Yes      |       8 |
| utf8mb4_romanian_ci          | utf8mb4 | 227 |         | Yes      |       8 |
| utf8mb4_slovenian_ci         | utf8mb4 | 228 |         | Yes      |       8 |
| utf8mb4_polish_ci            | utf8mb4 | 229 |         | Yes      |       8 |
| utf8mb4_estonian_ci          | utf8mb4 | 230 |         | Yes      |       8 |
| utf8mb4_spanish_ci           | utf8mb4 | 231 |         | Yes      |       8 |
| utf8mb4_swedish_ci           | utf8mb4 | 232 |         | Yes      |       8 |
| utf8mb4_turkish_ci           | utf8mb4 | 233 |         | Yes      |       8 |
| utf8mb4_czech_ci             | utf8mb4 | 234 |         | Yes      |       8 |
| utf8mb4_danish_ci            | utf8mb4 | 235 |         | Yes      |       8 |
| utf8mb4_lithuanian_ci        | utf8mb4 | 236 |         | Yes      |       8 |
| utf8mb4_slovak_ci            | utf8mb4 | 237 |         | Yes      |       8 |
| utf8mb4_spanish2_ci          | utf8mb4 | 238 |         | Yes      |       8 |
| utf8mb4_roman_ci             | utf8mb4 | 239 |         | Yes      |       8 |
| utf8mb4_persian_ci           | utf8mb4 | 240 |         | Yes      |       8 |
| utf8mb4_esperanto_ci         | utf8mb4 | 241 |         | Yes      |       8 |
| utf8mb4_hungarian_ci         | utf8mb4 | 242 |         | Yes      |       8 |
| utf8mb4_sinhala_ci           | utf8mb4 | 243 |         | Yes      |       8 |
| utf8mb4_german2_ci           | utf8mb4 | 244 |         | Yes      |       8 |
| utf8mb4_croatian_mysql561_ci | utf8mb4 | 245 |         | Yes      |       8 |
| utf8mb4_unicode_520_ci       | utf8mb4 | 246 |         | Yes      |       8 |
| utf8mb4_vietnamese_ci        | utf8mb4 | 247 |         | Yes      |       8 |
| utf8mb4_croatian_ci          | utf8mb4 | 608 |         | Yes      |       8 |
| utf8mb4_myanmar_ci           | utf8mb4 | 609 |         | Yes      |       8 |
| utf8mb4_thai_520_w2          | utf8mb4 | 610 |         | Yes      |       4 |
+------------------------------+---------+-----+---------+----------+---------+
29 rows in set (0.00 sec)
```

A modo de ejemplo, vamos a crear una base de datos para una empresa en la que se va a almacenar información sobre los pedidos realizados a la misma. A la base de datos la vamos a llamar *Pedidos* y vamos a crearla asignándole el cotejamiento *utf8mb4_spanish_ci*:

```
mysql> create database pedidos
    -> collate utf8mb4_spanish_ci;
Query OK, 1 row affected (0.00 sec)
```

Si ejecutamos a continuación el comando SHOW DATABASES observamos que una de las que aparece es la base de datos *Pedidos* que acabamos de crear.

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| pedidos            |
| performance_schema |
| phpmyadmin         |
| test               |
+--------------------+
6 rows in set (0.00 sec)
```

Para realizar alguna modificación en una base de datos previamente creada se usa la sentencia ALTER DATABASE o ALTER SCHEMA, cuyo formato es el siguiente:

```
ALTER {DATABASE | SCHEMA}  nombre _BD
especificación_alteración …

Especificación_alteración:
      [DEFAULT] CHARACTER SET [=] nombre_conjunto_caracteres
      | [DEFAULT] COLLATE [=] nombre_cotejamiento
```

Como se puede observar, se debe indicar después de ALTER DATABASE o ALTER SCHEMA el nombre de la base de datos. Sobre la base de datos, como se puede observar, se puede modificar el conjunto de caracteres permitidos y/o el cotejamiento.

Por ejemplo, para modificar el conjunto de caracteres permitidos para la base de datos *P*, podemos poner:

```
mysql> alter database p
    -> character set latin1;
Query OK, 1 row affected (0.02 sec)
```

Para eliminar una base de datos y, por tanto, todo su contenido, se usará la orden DROP DATABASE con el siguiente formato:

```
DROP {DATABASE | SCHEMA} [IF EXISTS] nombre _BD
```

Como se puede observar, después de DROP se debe escribir DATABASE o SCHEMA y después se debe indicar el nombre de la base de datos que se desea borrar. Esta orden, en el caso de que no exista una base de datos con el nombre indicado, devolverá un mensaje de error. Se puede usar la cláusula IF EXISTS si se quiere omitir el mensaje de error en este caso.

La orden DROP DATABASE se debe emplear con cuidado pues no se muestra un mensaje de confirmación antes de proceder al borrado de toda la base de datos.

A modo de ejemplo, se va a borrar la base de datos llamada *prueba*:

```
mysql> drop database prueba;
Query OK, 1 row affected (0.22 sec)
```

4. Creación, modificación y borrado de tablas.

La orden para crear una tabla es la orden CREATE TABLE, cuyo formato genérico es el siguiente:

CREATE TABLE [IF NOT EXISTS] nombre_tabla
(columna$_1$ definición_columna$_1$ [restricciones_columna$_1$],
columna$_2$ definición_columna$_2$ [restricciónes_columna$_2$],
….
columna$_n$ definición_columna$_n$ [restricciones_columna$_n$],
[restricción_tabla$_1$],
[restricción_tabla$_2$], …) [opciones_tabla];

Si ya existe en la base de datos una tabla con el nombre indicado, se mostrará un mensaje de error. Para evitar esto, se puede incluir antes del nombre de la tabla la cláusula IF NOT EXISTS, de tal forma que, en este caso, si no existe ninguna tabla con el nombre indicado en la base de datos, se crea; en caso contrario, no se crea, pero no se muestra ningún mensaje de error.

Una tabla consta de varios campos, atributos o columnas. Pues bien, por cada uno de estos atributos habrá de indicarse su nombre y a continuación se debe definir. Esta definición puede presentar el siguiente formato:

Definición_columna:
Tipodato [NOT NULL] [DEFAULT valor] [AUTO_INCREMENT] [COMMENT 'comentarios']

Se explican a continuación cada uno de los elementos indicados:

- Tipodato hace referencia a uno de los tipos de datos explicados anteriormente. Al tipo de dato indicado se le pueden añadir a continuación los siguientes modificadores:
  - UNSIGNED: Quiere decir que no se admiten signos, lo que conlleva que solo se puedan almacenar números positivos y el cero. Solo es aplicable a campos de tipo numérico.
  - ZEROFILL: Los espacios en blanco a la izquierda del número se rellenan con ceros. Solo es aplicable a campos de tipo numérico.
  - BINARY: Es aplicable a los tipos de datos char, varchar, tinytext, text, mediumtext y longtext y sirve para indicar a MySQL que distinga en los datos almacenados entre letras mayúsculas y minúsculas, ya que por defecto no lo hace.
- NOT NULL: Indica que el atributo correspondiente es requerido u obligatorio, es decir, que se debe rellenar obligatoriamente.
- DEFAULT valor: Permite especificar el valor por defecto que toma un atributo, de manera que para todas las filas que se añadan a la tabla, ese atributo tomará ese valor por defecto a no ser que se especifique un valor distinto.
  - Si se trata de un valor numérico se especifica sin más, poniendo después de la palabra DEFAULT el número en cuestión. Si se trata de un número real, se emplea el punto para separar la parte entera de la parte decimal. Ejemplos: edad tinyint(3) default 25, salario float (7,2) default 1123.45.
  - Si se trata de una cadena de caracteres, se especifica el valor por defecto entre comillas simples. Ejemplo: provincia varchar(20) default 'Ávila'.

- Si se trata de una fecha o una hora, se especifica el valor por defecto entre comillas simples siguiendo la fecha el formato ‘aaaa-mm-dd’ y la hora el formato ‘hhh:mm:ss’. Ejemplo: fecnac date default ‘1960-01-01’.
- Si se trata de un campo del tipo enumerado (enum), para especificar el valor por defecto hay que indicar el valor concreto por defecto entre comillas simples. Ejemplo: Si tenemos el campo Idioma enum (‘Inglés’, ‘Francés’, ‘Alemán’, ’Italiano’), para indicar que el valor por defecto es Italiano, pondríamos:

  Idioma enum (‘Inglés’, ‘Francés’, ‘Alemán’, ’Italiano’) default ‘Italiano’

- AUTO_INCREMENT: Sirve para indicar que un campo es autonumérico, lo que quiere decir que su valor será asignado directamente por MySQL a medida que se vayan añadiendo filas a la tabla. Los campos así definidos tomarán el valor 1 para la primera fila añadida a la tabla y valores sucesivos: 2, 3, 4 …
- COMMENT ‘comentarios’: Permite especificar un comentario para un atributo de una tabla.

Además, se pueden incluir por cada atributo de la tabla una o varias restricciones de columna, que son aquellas que afectan a una sola columna o atributo. Las restricciones de columna permitidas son las siguientes:

- PRIMARY KEY: Sirve para indicar que ese atributo será la clave primaria de la tabla, lo que quiere decir que no podrá contener valores nulos ni podrá haber valores duplicados.
- UNIQUE: Sirve para indicar que ese campo debe tomar un valor único, es decir, que no podrá haber dos filas con el mismo valor en ese campo. Sirve para definir claves alternativas.
- CHECK (expresión): Sirve para crear restricciones de rechazo, de manera que si al actualizar el valor del atributo indicado no se cumpliese la condición especificada entre paréntesis, la operación en cuestión sería rechazada. En la expresión se incluirá el nombre del atributo en cuestión y muy probablemente algún valor constante. En la expresión se pueden emplear operadores aritméticos (+, - *, /), relacionales (>, >=, <, <=, =, <>, between … and …, in (…,…,…)) y lógicos (and, or, not). Ejemplo: check (edad between 16 and 65).

Una vez definidos todos los atributos y todas las restricciones de columna y de tabla y después del paréntesis de cierre, se pueden incluir varias opciones de tabla. De estas opciones las más interesantes son las siguientes:

ENGINE = motor_de_almacenamiento

[DEFAULT] CHARACTER SET [=] nombre_conjunto_caracteres

[DEFAULT] COLLATE [=] nombre_cotejamiento

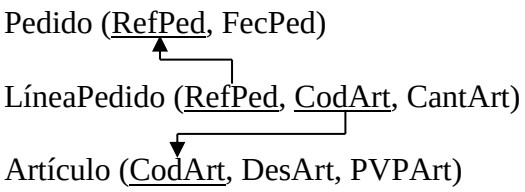
COMMENT = ‘comentarios’

MAX_ROWS = valor

Se explican a continuación cada una de estas opciones:

- Con ENGINE se indica el motor de almacenamiento deseado para la tabla. El motor de almacenamiento por defecto es InnoDB.
- Con CHARACTER SET se indica el conjunto de caracteres deseado para la tabla.
- Con COLLATE se puede indicar el cotejamiento preferido para la tabla.
- Se puede incluir con COMMENT un comentario para la tabla.
- Se puede indicar con MAX_ROWS el número máximo de filas que se van a poder almacenar en la tabla.

A modo de ejemplo, se van a incluir en la base de datos *Pedidos* creada anteriormente varias tablas, concretamente, las que aparecen en el siguiente esquema relacional. Este esquema contiene información sobre los artículos que vende una librería (tabla *Artículo*), los pedidos realizados a la librería por parte de sus clientes (tabla *Pedido*) e información sobre los artículos solicitados en cada uno de los pedidos (tabla *LineaPedido*).



Algunos datos que podrían contener estas tablas se muestran a continuación:

PEDIDO

RefPed	FecPed
P0001	16/02/2018
P0002	18/02/2018
P0003	23/02/2018
P0004	25/02/2018

LÍNEAPEDIDO

RefPed	CodArt	CantArt
P0001	A0043	10
P0001	A0078	12
P0002	A0043	5
P0003	A0075	20
P0004	A0012	15
P0004	A0043	5
P0004	A0089	50

ARTÍCULO

CodArt	DesArt	PVPArt
A0043	Bolígrafo azul fino	0,78
A0078	Bolígrafo rojo normal	1,05
A0075	Lápiz 2B	0,55
A0012	Goma de borrar	0,15
A0089	Sacapuntas	0,25

Figura 4: Contenido de la base de datos *Pedidos*.

Pues bien, para crear estas tablas, en primer lugar, nos deberemos situar en la base de datos en la que deseamos crear las tablas, que es la base de datos *Pedidos*. Para situarnos en una base de datos en concreto deberemos escribir la orden USE y a continuación el nombre de la base de datos:

```
MariaDB [(none)]> use Pedidos
Database changed
```

Comencemos creando la tabla *Pedido*:

- Al atributo *RefPed* le asignamos el tipo char(5), ya que las referencias de los pedidos son cadenas de caracteres de longitud fija 5. Además, este atributo es la clave primaria de la tabla, por lo que pondremos la restricción *primary key*.
- Al atributo *FecPed* le asignamos el tipo date y, dado que es obligatorio, le pondremos la restricción *not null*.

La orden CREATE TABLE quedará así sin asignar nombre a las restricciones:

```
MariaDB [Pedidos]> create table pedido
    -> (RefPed char(5) primary key,
    -> FecPed date not null);
Query OK, 0 rows affected (0.19 sec)
```

Tras crear una tabla, podemos ver que efectivamente está incluida dentro de la base de datos activa, escribiendo el comando SHOW TABLES, como se puede observar a continuación:

```
MariaDB [Pedidos]> show tables;
+-------------------+
| Tables_in_pedidos |
+-------------------+
| pedido            |
+-------------------+
1 row in set (0.00 sec)
```

Además, se puede observar la estructura o el diseño de una tabla en concreto escribiendo la orden DESC o DESCRIBE y a continuación el nombre de la tabla:

```
MariaDB [Pedidos]> desc pedido;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| RefPed | char(5) | NO   | PRI | NULL    |       |
| FecPed | date    | NO   |     | NULL    |       |
+--------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Creemos a continuación la tabla *Articulo*:

- Al atributo *CodArt* le asignaremos el tipo char(5) y la restricción *primary key* por ser este atributo la clave primaria de la tabla.
- Las descripciones de los artículos son cadenas alfanuméricas sin longitud fija, por lo que al atributo *DesArt* le asignaremos el tipo varchar(30) (se ha considerado 30 como

longitud máxima). Además, le asignaremos la restricción *not null* por ser un atributo obligatorio.

- Al atributo *PVPArt* le asignaremos el tipo float (6,2) para indicar que es un número que admite decimales con 4 posiciones antes de la coma y 2 después de ella. Como no se deben permitir números negativos, pondremos *unsigned*. Además, incluiremos la restricción not null por ser obligatorio.

La instrucción quedará como sigue:

```
MariaDB [Pedidos]> create table articulo
    -> (CodArt char(5) primary key,
    -> DesArt varchar(30) not null,
    -> PVPArt float(6,2) unsigned not null);
Query OK, 0 rows affected (0.06 sec)

MariaDB [Pedidos]> desc articulo;
+--------+---------------------+------+-----+---------+-------+
| Field  | Type                | Null | Key | Default | Extra |
+--------+---------------------+------+-----+---------+-------+
| CodArt | char(5)             | NO   | PRI | NULL    |       |
| DesArt | varchar(30)         | NO   |     | NULL    |       |
| PVPArt | float(6,2) unsigned | NO   |     | NULL    |       |
+--------+---------------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Además, se pueden incluir, normalmente tras la definición de todos los atributos, una o varias restricciones de tabla, que son aquellas que afectan a varias columnas o atributos. A algunas de estas restricciones se les puede asignar un nombre, en cuyo caso antes de la restricción habrá que escribir la palabra CONSTRAINT y a continuación el nombre que se le desea dar a la restricción. Las restricciones de tabla permitidas son las siguientes:

- [CONSTRAINT nombre_restricción] PRIMARY KEY ($columna_1$, $columna_2$, …): Sirve para indicar que los atributos indicados entre paréntesis y separados por comas constituirán la clave primaria de la tabla, lo que quiere decir que ninguno de ellos podrá contener valor nulo ni podrá haber valores duplicados para el grupo de atributos indicado.
- [CONSTRAINT nombre_restricción] UNIQUE ($columna_1$, $columna_2$, …): Sirve para indicar que el conjunto de atributos especificados entre paréntesis debe tomar un valor único, es decir, que no podrá haber dos filas con la misma combinación de valores para el grupo de atributos . Sirve para definir claves alternativas.
- INDEX [nombre_índice] ($columna_1$, $columna_2$, …): Permite crear un índice para los atributos indicados.
- [CONSTRAINT nombre_restricción] FOREIGN KEY ($columna_{11}$, $columna_{12}$, …) REFERENCES tabla [($columna_{21}$, $columna_{22}$, …)] [ON DELETE opción_referencia] [ON UPDATE opción_referencia]: Sirve para indicar que el/los atributos $columna_{11}$,

columna$_{12}$, … de la tabla que se está definiendo constituyen una clave ajena que hace referencia a la tabla indicada después de la palabra REFERENCES y al / a los atributos indicados a continuación entre paréntesis. Si se omiten este/os último/s atributo/s, se supone que la clave ajena apunta a la clave primaria de la tabla referenciada. Para que las restricciones FOREIGN KEY funcionen es necesario que la dos tablas sean del tipo InnoDB.

Se pueden indicar después de la cláusula REFERENCES las opciones seleccionadas para el borrado y modificación de filas que contienen la clave referenciada.

opción_referencia:

{RESTRICT | CASCADE | SET NULL | NO ACTION}

Las opciones disponibles en MySQL son las siguientes:

- RESTRICT: No se va a permitir el borrado o modificación de filas de la relación referenciada si hay alguna fila en la otra relación que contiene el mismo valor en la clave ajena. Es la opción por defecto.
- CASCADE: El borrado o modificación de filas de la relación que contiene la clave referenciada implica el borrado o modificación en cascada de las tuplas correspondientes en la tabla que contiene la clave ajena.
- SET NULL: El borrado o modificación de filas de la relación que contiene la clave referenciada lleva consigo poner a valor nulo el atributo que constituye la clave ajena.
- NO ACTION: Es equivalente a la opción RESTRICT.

- CHECK (expresión): Permite crear restricciones de rechazo en cuya expresión aparecen varios atributos de la tabla.

Para nuestra base de datos *Pedidos* nos falta por crear la tabla *LineaPedido*. Pues bien, con respecto a esta tabla:

- Al atributo *RefPed* le asignaremos el tipo char(5), tipo exactamente igual que la clave primaria de la tabla *Pedido*, atributo al que referencia. Se trata de un atributo obligatorio, pero no es necesario que incluyamos en este caso la restricción not null. Tengamos en cuenta que este atributo forma parte de la clave primaria de la tabla (aspecto que indicaremos más abajo en la definición de la tabla) y por la restricción de integridad de la entidad, ningún atributo que forme parte de la clave primaria de una tabla puede tomar valor nulo, lo que se encuentra implementado en el SGBD MySQL.
- Al atributo *CodArt* le asignaremos el tipo char(5), tipo exactamente igual que la clave primaria de la tabla *Articulo*, atributo al que referencia. Por el mismo motivo aducido

para el atributo *RefPed*, no es necesario incluir la restricción not null, a pesar de tratarse de un atributo obligatorio.

- Al atributo *CantArt* le asignaremos el tipo int (4) pues es un número sin decimales y no se consideran necesarios más de 4 dígitos. Como no se deben permitir números negativos, pondremos *unsigned*. Además, le asignaremos como valor por defecto un 1. Por considerar que es obligatorio, pondremos la restricción NOT NULL.
- A continuación especificaremos una restricción de clave ajena para el atributo *RefPed*. Recordemos que apunta al atributo homónimo de la tabla *Pedido*. Si deseamos que al modificar la referencia de un pedido en la tabla *Pedido*, se modifique el atributo *RefPed* para todas sus líneas de pedido, entonces es conveniente que añadamos la cláusula ON UPDATE CASCADE. Para el caso del borrado, es aconsejable ser más cauteloso y no realizar borrados en cascada. Por ello, optamos por la opción por defecto (RESTRICT). Al tratarse de la opción por defecto, no es necesario escribir nada.
- A continuación especificaremos una restricción de clave ajena para el atributo *CodArt*. Recordemos que apunta al atributo homónimo de la tabla *Artículo*. Si deseamos que al modificar el código de un artículo en la tabla *Articulo*, se modifique el atributo *CodArt* para todas las líneas de pedido en que aparece, entonces es conveniente que añadamos la cláusula ON UPDATE CASCADE. Para el caso del borrado, es aconsejable ser más precavido y no realizar borrados en cascada.
- Debemos indicar mediante una restricción de tabla que la clave primaria de esta tabla está formada por la pareja de atributos (RefPed, CodArt).

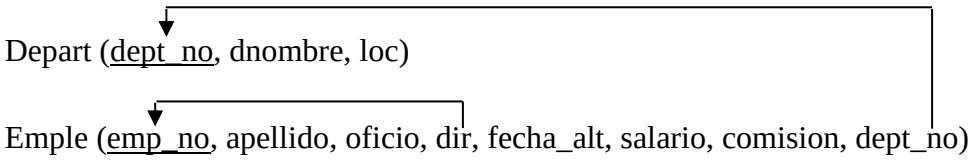
La instrucción quedará como sigue:

```
MariaDB [Pedidos]> create table LineaPedido
    -> (RefPed char(5),
    -> CodArt char(5),
    -> CantArt int(4) unsigned not null default 1,
    -> foreign key (RefPed) references Pedido(RefPed) on update cascade,
    -> foreign key (CodArt) references Articulo(CodArt) on update cascade,
    -> primary key (RefPed, CodArt));
Query OK, 0 rows affected (0.28 sec)

MariaDB [Pedidos]> desc LineaPedido;
+---------+-----------------+------+-----+---------+-------+
| Field   | Type            | Null | Key | Default | Extra |
+---------+-----------------+------+-----+---------+-------+
| RefPed  | char(5)         | NO   | PRI | NULL    |       |
| CodArt  | char(5)         | NO   | PRI | NULL    |       |
| CantArt | int(4) unsigned | NO   |     | 1       |       |
+---------+-----------------+------+-----+---------+-------+
3 rows in set (0.06 sec)
```

Si bien se ha distinguido entre restricciones de columna (aquellas que afectan a un solo atributo) y restricciones de tabla, todas ellas (tanto las de columna como las de tabla) se pueden especificar como si fuesen restricciones de tabla aunque afecten a un solo atributo.

A modo de un segundo ejemplo, se va a crear una nueva base de datos llamada *Empresa* en la que se van a crear dos tablas, concretamente, las que aparecen en el siguiente esquema relacional. Este esquema contiene información sobre los departamentos en que se divide una empresa (tabla *Depart*) y los empleados que trabajan en los mismos (tabla *Emple*):



Por cada departamento se almacena en la tabla *Depart* un número identificativo (*dept_no*), su nombre (*dnombre*) y la localidad en la que está ubicado (*loc*). Por otro lado, por cada empleado se almacena un número que lo identifica (*emp_no*), su apellido (*apellido*), el oficio o puesto que desempeña en la empresa (*oficio*), el número del empleado director (*dir*), su fecha de ingreso en la empresa (*fecha_alt*), el salario que cobra (*salario*), su comisión (*comision*) y el número identificativo del departamento en el que trabaja (*dept_no*). Los datos que se van a almacenar en esta base de datos se muestran en la figura 5:

Tabla Depart

dept_no	dnombre	loc
10	CONTABILIDAD	SEVILLA
20	INVESTIGACIÓN	MADRID
30	VENTAS	BARCELONA

Tabla Emple

emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7369	SÁNCHEZ	EMPLEADO	7900	2016-12-12	600.00	0.00	20
7499	ARROYO	VENDEDOR	7698	2013-02-20	1200.00	240.00	30
7521	SALA	VENDEDOR	7698	2014-02-22	960.00	390.00	30
7566	JIMÉNEZ	DIRECTOR	7839	2014-02-04	2300.00	0.00	20
7654	MARTÍN	VENDEDOR	7698	2014-09-29	965.00	1000.00	30
7698	NEGRO	DIRECTOR	7839	2014-05-01	2200.00	0.00	30
7738	CEREZO	DIRECTOR	7839	2014-09-06	2210.00	0.00	10
7788	GIL	ANALISTA	7566	2017-04-23	2350.00	0.00	20
7839	REY	PRESIDENTE	NULL	2014-11-17	3900.00	0.00	10
7844	TOVAR	VENDEDOR	7698	2014-09-08	1100.00	0.00	30
7876	ALONSO	EMPLEADO	7788	2017-08-09	860.00	0.00	20
7900	JIMENO	EMPLEADO	7698	2014-12-03	725.00	0.00	30

Figura 5: Contenido de la base de datos *Empresa*.

En primer lugar, crearemos la base de datos *Empresa* y accederemos a ella:

```
MariaDB [Pedidos]> create database empresa collate utf8mb4_spanish_ci;
Query OK, 1 row affected (0.00 sec)

MariaDB [Pedidos]> use empresa
Database changed
```

A continuación pasaremos a crear las tablas. Comenzaremos creando la tabla *Depart* porque *Emple* tiene una clave ajena a *Depart*:

- En cuanto al atributo *dept_no*, si suponemos que los números de departamento pueden tener como máximo 3 cifras y son números positivos, le asignaremos el tipo int(3) unsigned. Indicaremos que este atributo es la clave primaria de la tabla.
- Al atributo *dnombre* le asignaremos el tipo varchar por ser una cadena de caracteres de longitud variable. Además, para indicar que se trata de un campo obligatorio pondremos not null y si sabemos además que los nombres de los departamentos no se pueden repetir, le asignaremos también la restricción unique.
- El atributo *loc* lo definiremos también como cadena de caracteres de longitud variable e indicaremos que es obligatorio.

La orden CREATE TABLE sin asignar nombre a las restricciones nos quedará como sigue:

```
MariaDB [empresa]> Create table depart
    -> (dept_no int(3) unsigned primary key,
    -> dnombre varchar(40) unique not null,
    -> loc varchar(40) not null);
Query OK, 0 rows affected (0.32 sec)
```

En cuanto a la table *Emple*:

- Como vemos en la figura 5, el atributo *emp_no* es un campo numérico entero de 4 cifras y positivo (int(4) unsigned). Además, se debe indicar que es clave primaria.
- Al atributo *apellido* le asignaremos el tipo varchar y longitud máxima 40. Además, indicaremos que es obligatorio con la restricción not null.
- El oficio de los empleados solo puede tomar los valores EMPLEADO, VENDEDOR, DIRECTOR, ANALISTA, PROGRAMADOR y PRESIDENTE, por lo que lo definiremos con el tipo enum. Además, obligaremos a introducir siempre uno de estos valores poniendo la restricción not null.
- El atributo *dir* es una clave ajena al atributo *emp_no* de la misma tabla *Emple*, por lo que su tipo debe ser igual (int(4) unsigned). Como este atributo es una clave ajena, deberemos especificar una restricción de clave ajena indicando que apunta al atributo *emp_no* de la tabla *Emple* (cláusula references). Si deseamos que al cambiar el atributo *emp_no* de un

empleado, se modifique automáticamente el atributo *dir* para aquellos empleados que lo tienen como director, pondremos la cláusula ON UPDATE CASCADE.

- El atributo *fecha_alt* es obvio que es de tipo date. Lo pondremos como un atributo obligatorio (not null).
- El *salario* debe ser un atributo de tipo float por ser numérico con decimales. Permitiremos 5 dígitos antes de la coma y 2 después de ella (por los céntimos de euro), por lo que su tipo será float (7,2). Pondremos que solo se admitirán salarios positivos (unsigned) y que es obligatorio.
- El atributo *comision* será del mismo tipo que el atributo *salario*, pero en este caso, no es obligatorio.
- El atributo *dept_no* es clave ajena al atributo homónimo de la tabla *Depart*, por lo que su tipo debe ser igual (int(3) unsigned). Le asignaremos por defecto el valor 10 y lo pondremos como obligatorio. Por tratarse de una clave ajena, especificaremos una restricción foreign key apuntando al atributo *dept_no* de la tabla *Depart*. Nos interesa que al modificarse el número de un departamento en la tabla *Depart* se modifique automáticamente el atributo *dept_no* para los empleados que trabajen en ese departamento, por lo que añadimos ON UPDATE CASCADE.

La orden CREATE TABLE sin asignar nombre a las restricciones nos quedará como sigue:

```
MariaDB [empresa]> create table emple
    -> (emp_no int(4) unsigned primary key,
    -> apellido varchar(40) not null,
    ->  oficio  enum  ('EMPLEADO',  'VENDEDOR',  'DIRECTOR',  'ANALISTA',
                      'PROGRAMADOR', 'PRESIDENTE') not null,
    -> dir int(4) unsigned,
    -> fecha_alt date not null,
    -> salario float(7,2) unsigned not null,
    -> comision float(7,2) unsigned,
    -> dept_no int(3) unsigned default 10 not null,
    -> foreign key(dir) references emple(emp_no) on update cascade,
    -> foreign key(dept_no) references depart(dept_no) on update cascade);
Query OK, 0 rows affected (0.21 sec)
```

Sobre una tabla también se puede modificar su estructura o diseño haciendo uso de la sentencia ALTER TABLE.

Para practicar la sentencia ALTER TABLE, vamos a crear una nueva base de datos llamada *Instituto* que almacenará información sobre los alumnos que cursan en un instituto ciclos formativos de Formación Profesional. En esta base de datos crearemos dos tablas: *Alumnos* y *Estudios*, con la siguiente estructura:

Alumnos (NIF, Nombre, Apellidos, Ciclo, Curso, Idioma, Dirección)

Estudios (CodCiclo, NomCiclo, Nivel, Duración)

Comenzaremos creando la base de datos de la siguiente manera:

```
MariaDB [(none)]> create database instituto collate utf8mb4_spanish_ci;
Query OK, 1 row affected (0.05 sec)
```

Antes de crear las tablas, tendremos que indicar que vamos a utilizar la base de datos recién creada:

```
MariaDB [(none)]> use instituto;
Database changed
```

Comenzaremos creando la tabla *Estudios* con la siguiente orden SQL:

```
MariaDB [instituto]> create table Estudios
    -> (CodCiclo char(3) primary key,
    -> NomCiclo varchar(40) not null unique,
    -> Nivel enum ('M', 'S') not null,
    -> Duracion int(4) unsigned not null);
Query OK, 0 rows affected (0.17 sec)
```

A continuación se va a crear la tabla *Alumnos*, pero sin incluir varios aspectos que añadiremos después a la tabla mediante sentencias ALTER TABLE, como, por ejemplo, la indicación de la clave primaria de la tabla, de la clave ajena, etc. La orden SQL con la que crearemos por ahora la tabla *Alumnos* es la siguiente:

```
MariaDB [instituto]> create table Alumnos
    -> (NIF char(9),
    -> Nombre varchar(20) not null,
    -> Apellidos varchar(40) not null,
    -> Ciclo char(3) not null,
    -> Curso enum('1', '2') not null,
    -> Idioma enum ('Inglés', 'Francés'),
    -> Dirección varchar(40) not null);
Query OK, 0 rows affected (0.09 sec)
```

Veamos a continuación las opciones para la sentencia ALTER TABLE. Pues bien, la modificación del diseño de una tabla puede implicar:

- Añadir un nuevo atributo a la tabla, lo que se llevará a cabo de acuerdo con la siguiente sintaxis:

  ```
  ALTER TABLE nombre_tabla
  ADD [COLUMN]  columna definición_columna [restricciones_columna];
  ```

  Como se puede observar, se debe indicar después de ADD el nombre del atributo que se desea añadir, su definición y opcionalmente la o las restricciones de columna que se deseen para él.

  A modo de ejemplo, añadamos un atributo llamado *NumMatrícula* a la tabla *Alumnos* como un número entero de como máximo 6 cifras, con relleno de ceros por la izquierda, no negativo y obligatorio:

```
MariaDB [instituto]> alter table Alumnos
    -> add NumMatrícula int(6) unsigned zerofill not null;
Query OK, 0 rows affected (0.24 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- Añadir un nuevo índice a la tabla, para lo que se empleará la siguiente sintaxis:

```
ALTER TABLE nombre_tabla
ADD INDEX [nombre_índice] (columna1, columna2, …);
```

Como se puede observar, se puede indicar después de ADD INDEX el nombre del índice opcionalmente y a continuación se debe indicar entre paréntesis el/los atributo/s para el/los que se desea establecer el índice.

A modo de ejemplo, añadamos un índice para el atributo *Ciclo* de la tabla *Alumnos*:

```
MariaDB [instituto]> alter table Alumnos
    -> add index(Ciclo);
Query OK, 0 rows affected (0.23 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- Añadir una clave primaria para la tabla, para lo que se empleará la sintaxis:

```
ALTER TABLE nombre_tabla
ADD [CONSTRAINT nombre_restricción] PRIMARY KEY (columna1, columna2, …);
```

Se puede asignar un nombre a la restricción y se debe indicar después de PRIMARY KEY entre paréntesis el/los atributo/s que forma/n parte de la clave primaria de la tabla.

A modo de ejemplo, indiquemos para la tabla *Alumnos* que su clave primaria es el NIF:

```
MariaDB [instituto]> alter table Alumnos
    -> add primary key(NIF);
Query OK, 0 rows affected (0.16 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- Añadir una clave ajena, para lo que se empleará la sintaxis:

```
ALTER TABLE nombre_tabla
ADD [CONSTRAINT nombre_restricción] FOREIGN KEY (columna11, columna21, …)
REFERENCES tabla [(columna21, columna22, …)] [ON DELETE
opción_referencia] [ON UPDATE opción_referencia]
```

Se puede asignar un nombre a la restricción y luego se añade la restricción de clave ajena, como se explicó anteriormente.

A modo de ejemplo, se va a indicar que el atributo *Ciclo* de la tabla *Alumnos* es una clave ajena al atributo *CodCiclo* de la tabla *Estudios*. Vamos a indicar que al modificar el código de un ciclo se modifique dicho código en todas las filas de la tabla *Alumnos* para todos los alumnos que cursen dicho ciclo:

```
MariaDB [instituto]> alter table Alumnos
    -> add foreign key (Ciclo) references Estudios(CodCiclo) on update cascade;
Query OK, 0 rows affected (1.23 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- Añadir una restricción de unicidad para uno o varios atributos, para lo que se seguirá la sintaxis:

```
ALTER TABLE nombre_tabla
ADD [CONSTRAINT nombre_restricción] UNIQUE (columna1, columna2, …)
```

Como se puede observar, se puede asignar un nombre a la restricción y luego habrá que indicar tras la palabra UNIQUE y entre paréntesis el/los atributo/s único/s.

A modo de ejemplo, se va a asignar una restricción de unicidad al atributo *NumMatrícula* de la tabla *Alumnos*:

```
MariaDB [instituto]> alter table Alumnos
    -> add unique (NumMatrícula);
Query OK, 0 rows affected (0.95 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- Asignar un valor por defecto a un atributo, para lo que se usa la siguiente sintaxis:

```
ALTER TABLE nombre_tabla
ALTER [COLUMN] columna SET DEFAULT literal;
```

Como se puede observar, se debe indicar el atributo al que se desea asignar el valor por defecto y dicho valor después de SET DEFAULT.

A modo de ejemplo, se va a asignar al atributo *Duracion* de la tabla *Estudios* el valor 2000, ya que la mayoría de los ciclos formativos tienen una duración de 2000 horas:

```
MariaDB [instituto]> alter table Estudios
    -> alter Duracion set default 2000;
Query OK, 0 rows affected (0.16 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- Cambiar el nombre de un atributo y su definición, de acuerdo con la siguiente sintaxis:

```
ALTER TABLE nombre_tabla
CHANGE [COLUMN] nombre_anterior_columna
nuevo_nombre_columna definición_columna [restricciones_columna];
```

Como se puede observar, se puede asignar un nuevo nombre al atributo y a continuación se deberá definir el atributo y se le pueden asignar restricciones.

- Cambiar la definición de un atributo sin cambiar su nombre, para lo que se usa la sintaxis:

```
ALTER TABLE nombre_tabla
MODIFY [COLUMN] columna definición_columna [restricciones_columna];
```

A modo de ejemplo, se va a modificar el campo *Idioma* de la tabla *Alumnos* para hacer que además del inglés y el francés, los alumnos puedan seleccionar el alemán y el italiano. Además vamos a poner como valor por defecto el idioma inglés.

```
MariaDB [instituto]> alter table Alumnos
    ->   modify   Idioma   enum('Inglés','Francés','Alemán','Italiano')
default 'Inglés';
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- Eliminar un atributo, para lo que se usa la sintaxis:

```
ALTER TABLE nombre_tabla
DROP [COLUMN] columna;
```

A modo de ejemplo, se va a eliminar el atributo *Duracion* de la tabla *Estudios*:

```
MariaDB [instituto]> alter table Estudios
    -> drop Duracion;
Query OK, 0 rows affected (0.19 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- Eliminar la restricción de clave primaria, para lo que se usa la sintaxis:

```
ALTER TABLE nombre_tabla
DROP PRIMARY KEY;
```

- Eliminar un índice, para lo que se emplea la sintaxis:

```
ALTER TABLE nombre_tabla
DROP INDEX nombre_índice;
```

- Eliminar una clave ajena, para lo que se debe escribir:

```
ALTER TABLE nombre_tabla
DROP FOREIGN KEY nombre_restricción;
```

- Renombrar una tabla, para lo que se utiliza la sintaxis:

```
ALTER TABLE nombre_tabla
RENAME [TO] nuevo_nombre_tabla;
```

Para este cometido también se puede emplear el comando RENAME TABLE, que presenta la sintaxis:

```
RENAME TABLE nombre_tabla1 TO nuevo_nombre_tabla1,
             nombre_tabla2 TO nuevo_nombre_tabla2,…;
```

Como se puede observar, con un solo comando RENAME TABLE se pueden renombrar varias tablas.

- Modificar las opciones de una tabla, para lo que se emplea la sintaxis:

```
ALTER TABLE nombre_tabla
opciones_tabla;
```

La orden que se emplea para la eliminación completa de una tabla es la orden DROP TABLE. Su sintaxis es la siguiente:

```
DROP TABLE [IF EXISTS] tabla1, tabla2, …;
```

Esta orden no solo borra los datos contenidos en la tabla, sino también toda la tabla de la base de datos en la que está definida. Se pueden borrar con la misma orden varias tablas simplemente poniendo los diversos nombres de las tablas separados por comas. La opción IF EXISTS sirve para que si no existe alguna de las tablas, no se muestre el mensaje de error que aparece por defecto.

A modo de ejemplo, se van a borrar con la misma orden SQL las tablas *t1* y *t2*:

```
MariaDB [instituto]> drop table t1, t2;
Query OK, 0 rows affected (0.06 sec)
```

La orden TRUNCATE TABLE permite vaciar una tabla completamente, es decir, eliminar la totalidad de las filas almacenadas en la misma. Su sintaxis es la siguiente:

```
TRUNCATE TABLE nombre_tabla;
```

Esta orden tiene el mismo efecto que una orden DELETE sin cláusula WHERE, es decir, es equivalente en cuanto a efecto a la orden:

```
DELETE FROM nombre_tabla;
```

No obstante, hay algunas diferencias entre ambas órdenes. Las diferencias más relevantes son las siguientes:

- TRUNCATE es una operación que se registra en el log de transacciones como un todo, mientras que DELETE registra la eliminación de cada fila en dicho log.
- TRUNCATE se ejecuta en menor tiempo que DELETE.
- TRUNCATE no puede desencadenar la ejecución de disparadores, mientras que DELETE sí.
- TRUNCATE reiniciará el contador para una columna de tipo AUTO_INCREMENT, mientras que DELETE no.