Conceptos Basicos De Base De Datos

Elaborado Por:

Daniel Pardo Cuenca.

Universidad Nacional Abierta y a Distancia

2017

Qué es una base de datos

Una base de datos es un conjunto de datos **organizados** e **interrelacionados** que se organizan y relacionan entre sí de manera **sistemática**, esto es, siguiendo unas determinadas reglas. En muchos sitios veremos que se refieren a una base de datos con la abreviatura **BD** o **DB** (del inglés ***d**ata**b**ase*).

Ejemplos de bases de datos:

- La base de datos de una **tienda online**, con los datos de sus clientes, productos, métodos de pago, etc.
- La base de datos de un **foro online**, almacenando las conversaciones, usuarios, temas, etc.
- La base de datos de un **blog,** con los artículos, categorías, etiquetas, etc.

Objetos de una base de datos

Las bases de datos normalmente presentan 6 tipos de objetos:

- Tablas
- Vistas
- Funciones
- Índices
- Procesos almacenados
- Triggers o disparadores

Veamos cada uno de estos objetos con más detalle.

Tablas

Las tablas son los principales objetos de una base de datos. Representan la estructura física donde se almacenan los datos. Las tablas contienen registros y cada registro contiene campos. Un **registro** es cada una de las filas de la tabla,mientras que el **campo** es cada una de las columnas de la tabla.

Vistas

Son tablas que se forman a partir de otras tablas como resultado de una consulta SQL. Se pueden realizar sobre ellas las mismas operaciones que sobre las tablas, pero es importante recordar que los cambios afectan a las tablas originales, pues una vista es sólo eso, un modo de visualizar los datos de otras tablas.

Funciones

Son operaciones que el sistema gestor de base de datos realiza sobre las mismas. Estas operaciones son necesarias para poder interactuar con la base de datos.

Ejemplo: Operaciones para crear los objetos de la base de datos: tablas, vistas, etc.

Índices

Los índices permiten acceder a los elementos con mayor rapidez a los registros de una tabla de una base de datos. Normalmente se utilizan en aquellos campos que son más frecuentes en las búsquedas.

Ejemplo: Utilizar el número de pasaporte para localizar a un ciudadano europeo en una base de datos interestatal. De todos los datos de la tabla “ciudadano europeo”, escogeríamos el pasaporte como índice.

Procesos almacenados

Se trata de un programa que se almacena en la base de datos y que se ejecuta directamente en el sistema gestor de base de datos.

Ejemplo: buscar en la base de datos todos los usuarios cuya fecha de nacimiento sea hoy y enviarles una felicitación de cumpleaños

Triggers o disparadores

Es un proceso que se ejecuta únicamente cuando se cumple una condición preestablecida. Los triggers o disparadores pueden crear, editar o borrar tablas en una base de datos.

Ejemplo: Envío de un correo electrónico de bienvenida a un usuario que se acaba de registrar.

Qué es un sistema gestor de bases de datos

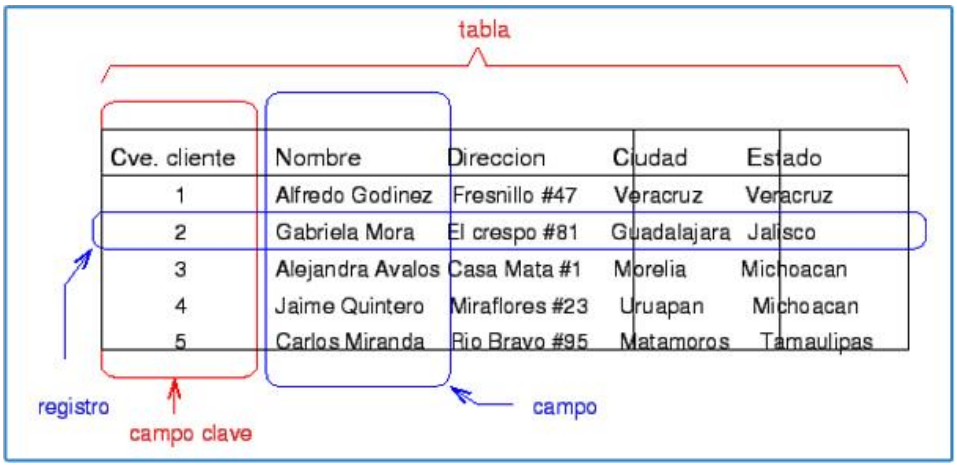
Un sistema gestor de bases de datos es una **aplicación informática** que permite al usuario interactuar con las bases de datos. Es una interfaz que permite al usuario acceder a los datos almacenados en las bases de datos que lo integran. Nos acostumbramos a referir a estos sistemas con las siglas **DBMS**, que provienen del nombre en inglés: ***D**ata**b**ase **M**anagement **S**ystem*.

Modelos de bases de datos

Tenemos diferentes modelos de bases de datos, entre ellos cabe destacar:

Modelo tabla.

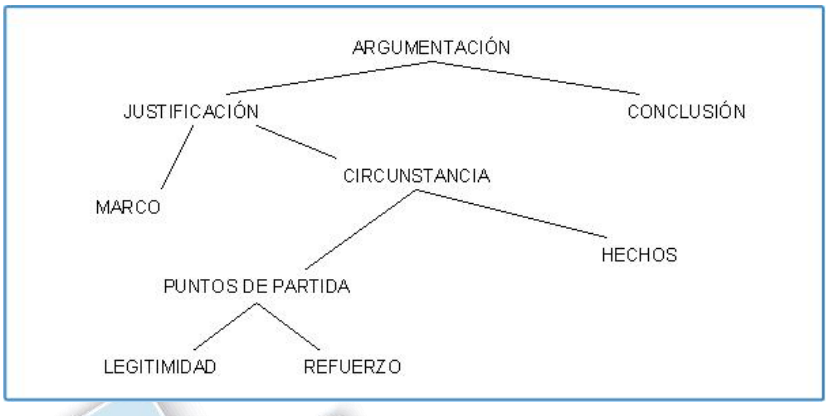
Se trata de una serie formada por una tabla bidimensional compuesta por *registros* y por *campos* en la que se recogen los datos.



Modelo jerárquico.

Se basa en registros organizados en forma de **árbol** jerárquico inverso.

- Ventajas: Facilita las relaciones 1:N unidireccionales o padre-hijo, en el que el padre es el elemento superior y el hijo el que queda justo debajo (igual que sucede con CSS). **1:N** significa que un padre puede tener muchos hijos, pero un hijo sólo puede tener un padre.
- Inconvenientes: Es un modelo que implica la **duplicidad** de registros, lo que dificulta mucho su gestión en casos de bases de datos grandes.



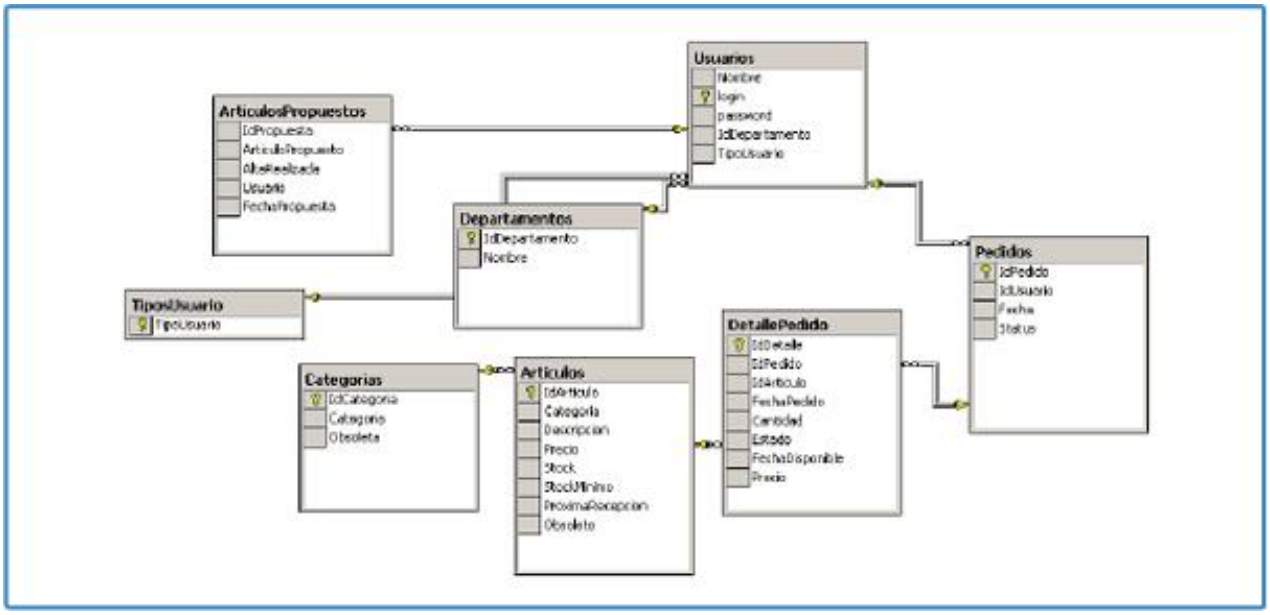
Modelo relacional.

Este es el modelo que se acabó imponiendo y **el más popular** actualmente. Está basado en el modelo de tablas, pero permitiendo la relación entre las diferentes tablas en base a unas reglas. MySQL se basa en este modelo, por lo que lo vamos a ver con más detalle. Ahora simplemente especificar estos puntos:

- Los elementos de una base de datos relacional son capaces de relacionarse sin necesidad de duplicar la información.

- Utilizan **SQL** (**S**tructured **Q**uery **L**anguage) para obtener la información de varias fuentes en una única consulta.

- Utilizan claves para establecer estas relaciones (claves primarias y externas, que veremos enseguida).



Conceptos básicos de las bases de datos relacionales

Los principales conceptos de las bases de datos relacionales son:

- Datos
- Entidades
- Claves primarias
- Claves externas
- Relaciones
- Restricciones de integridad referencia
- Metadatos

A continuación veremos cada uno de ellos con detalle y ejemplos que sirvan para entender los conceptos.

Datos.

Cogiendo la definición de Ramez Elmasri, los datos son hechos conocidos que se pueden registrar y que tienen un significado implícito.

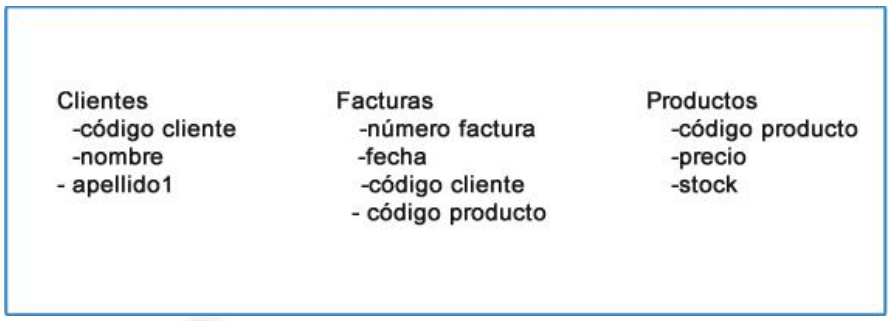
Ejemplos**:** nombre, apellido, dirección, teléfono.

Entidades.

Una entidad es todo aquello de lo que nos interesa tener unos datos guardados.

Ejemplos: Clientes, personas, productos, trabajadores.

En un modelo de entidad-relación, las relaciones las conforman los atributos y los campos del tema que nos interesa guardar.



Claves primarias

Cada entidad tiene que tener una clave primaria que identifique únicamente al conjunto de datos. Siguiendo con el ejemplo, la entidad Clientes tendrá como clave primaria el código cliente, la entidad facturas el número facturas y la entidad producto el código producto. Es muy importante que la clave primaria sea **única** y que jamás se pueda repetir. Así, no puede haber dos facturas con un mismo número.

Claves externas

Las claves externas son las claves externas de otra entidad que forman parte de la tabla de la entidad actual. Así, en nuestro ejemplo, en la entidad Factura, el código cliente y el código producto son entidades externas.

Relaciones

Las relaciones son lo que dice el término, como se relacionan las entidades entre sí a través de sus campos y atributos. Así, en nuestro ejemplo, la entidad Facturas se relaciona con las entidades Clientes y Productos a través del código cliente y el código producto de cada uno de sus campos respectivamente.

Restricciones de integridad referencial

Las restricciones de integridad referencia son las condiciones que se han de cumplir para que el modelo tenga sentido y sea coherente.

Siguiendo con nuestro ejemplo, para que en las Facturas se pueda poner el código cliente, antes ha de existir el cliente.

Metadatos

Los metadatos son datos que informan sobre los datos presentes en una base de datos.

Ejemplos: longitud de un campo (número de caracteres), tipo de campo (texto, número..), información del campo, etc.

SQL y MySQL

Qué es SQL

SQL (Structured Query Language) es un lenguaje declarativo estándar para la gestión de bases de datos relacionales, que permite recuperar y modificar fácilmente información de interés de las bases de datos mediante consultas.

Actualmente, la mayoría de sistemas gestores de contenidos utilizan SQL, por lo que si se conoce SQL, se puede trabajar con diferentes DBMS como MySQL, Oracle, SQL Server, PostgreSQL, etc.

Características de SQL:

- Es un lenguaje declarativo de alto nivel que permite una alta productividad, ya que está basado en la gestión de un conjunto de registros.
- Explota tanto la flexibilidad como la potencia de los sistemas relacionales
- Se divide en dos sublenguajes:

- *Data Definition Language (DDL)*: lenguaje para la creación de objetos de una base de datos
- *Data Manipulation Language (DML)*: lenguaje para la manipulación y consulta de los datos de una base de datos.

Introducción a MySQL

Qué es MySQL

MySQL es un sistema de base de datos *basado en el modelo relacional*, multihilo y multiusuario. *Multihilo* significa que el sistema distribuye automáticamente las tareas a realizar entre los procesadores disponibles, optimizando el rendimiento. El nombre proviene de la unión de My con SQL. My era la hija del cofundador de la empresa originaria de la idea.

Por qué utilizar MySQL:

- Es **código abierto**, lo que significa que es gratis de utilizar y que se puede modificar.
- Su uso está **muy extendido**: desde sistemas gestores de contenidos como WordPress y Drupal, a grupos de empresas como Prisa.
- Es muy **fácil de aprender y utilizar**, al ser muy intuitivo.
- Funciona muy bien **junto con PHP**, lo que permite crean páginas web dinámicas con facilidad.

Operaciones básicas con MySQL

Con MySQL se pueden realizar muchos tipos de operaciones. Las operaciones básicas son:

- Crear bases de datos
- Crear usuarios, grupos de usuarios y contraseñas.
- Crear y modificar tablas
- Eliminar tablas
- Cargar datos
- Añadir y modifcar registros en las tablas
- Consultar y actualizar los registros
- Eliminar registros

- Hacer consultas conjuntas a varias tablas en modo vista
- Operaciones de agrupación, orden, comparación, etc.
- Operaciones avanzadas, como programación de triggers, procesos almacenados, etc.

Primeros Pasos en MySQL

Ya tenemos instalado MySQL y hemos iniciado la consola. Ahora vamos a aprender los comandos más importantes para empezar a trabajar con MySQL.

Ver las bases de datos instaladas

Para ver las bases de datos que tenemos instaladas, ejecutamos este comando en el terminal de mysql:

```
1 show databases;
```

como justo acabamos de instalar MySQL y no hemos creado todavía ninguna base de datos, te aparecerán las que viene por defecto, como `information_schema`, `test` o `mysql`.

Seleccionar una base de datos

Para seleccionar una base de datos, ejecutamos este comando:

```
1 use test;
```

Ahora tenemos seleccionada la base de datos test y todo lo que hagamos será sobre esta base de datos. Para seleccionar otra, usamos el mismo comando, cambiando el nombre de la base de datos.

Crear una base de datos

```
1 create database biblioteca;
```

Si todo va como esperado, aparecerá una línea de código similar a ésta: `Query OK, 1 row affected (0.05 sec)`. Acabamos de crear una base de datos llamada “biblioteca”. Para comprobarlo, es tan fácil como utilizar el comando que justo acabamos de aprender, `show databases;` Veremos que ahora en el listado aparece biblioteca.

Cerrar la consola

```
1 exit;
```

Para salir de la consola tecleamos simplemente exit y damos a enter. Es el único caso en que da igual poner o no el punto y coma final.

Conceptos Básicos de MySQL

Ya hemos visto como crear una base de datos y como crear una nueva. Ahora es el momento de poner manos a la obra y meternos de lleno en el mundo de MySQL. Para hacer fácil lo difícil, aprenderemos los comandos con un ejemplo. Si os surge cualquier duda o problema, sólo tenéis que dejar un comentario y haremos todo lo posible por ayudaros.

Para empezar, seleccionemos ahora la base de datos “biblioteca:

```
1 use biblioteca;
```

Crear una tabla

Partimos de la base de datos que acabamos de crear llamada “biblioteca”. En ella queremos crear una tabla llamada “libros” que recoja el fondo bibliográfico. Una **tabla** está formada por campos (columnas) y registros (filas) donde podemos almacenar la información deseada. En nuestra tabla de ejemplo crearemos los campos “titulo”, “autor”, “fecha_publicacion” y “cantidad”: No ponemos el campo “editorial” porque, como mañana veremos, es mejor separar algunos datos en diferentes tablas y luego utilizar un comando para enlazarlas.

```sql
create table libros(
       libro_id int unsigned auto_increment,
       titulo varchar(50) not null,
       autor varchar(30) not null default 'Desconocido',
       cantidad smallint unsigned default 0,
      primary key (libro_id)
 );
```

No nos asustemos, aquí hemos introducido muchas cosas de golpe, vayamos paso por paso. Hemos creado varios campos con una serie de parámetros. Los campos han de ir separados por coma y dentro del paréntesis. Aquí el punto y coma va al final, después del paréntesis. Es muy fácil olvidárselo, por lo que prestad especial atención en este punto. Veamos ahora que hemos hecho: `create table libros` crea una tabla llamada libros. Podríamos haber hecho lo siguiente:

```sql
create table libros;
```

Lo que habría pasado, es que habríamos creado una tabla vacía, esto es, sin ningún campo. Como hemos dicho, una tabla está compuesta por **campos** y por **registros**. Para poder introducir un registro en la misma, primero es necesario crear los campos, que es lo que hemos hecho. Así en nuestro ejemplo hemos creado cuatro campos:l*ibro_id, titulo, autor y cantidad*. ¿por qué crear un libro_id? Porque, como explicábamos anteriormente toda tabla tiene que tener definida una **clave primaria** que tiene que ser única para cada registro.

Veamos ahora uno por uno cada uno de los campos creados:

```sql
libro_id int unsigned auto_increment,
```

Lo primero que indicamos siempre es el nombre del campo que estamos creando. `int` significa integer, lo que quiere decir que tiene que ser un número entero. `unsigned` significa que no puede ser negativo, por lo que la cuenta empezará con el 1. `auto_increment` significa que el propio gestor de bases de datos, en este caso MySQL, incrementará de manera automática este valor cada vez que introduzcamos un registro en esta tabla.

```sql
titulo varchar(50) not null,
);
```

Ahora estamos creando un campo llamado título. `varchar` significa caracteres y `varchar(50)` indica que este campo está formado por una cadena de como máximo 50 caracteres. `significa que este campo es obligatorio y que no podemos dejarlo en blanco.`

```
1 autor varchar(30) not null default 'Desconocido',
```

Efectivamente, esta sentencia crea un campo llamado autor de como mucho 30 caracteres que tampoco puede dejarse en blanco. Aquí introducimos una diferencia, y es que hemos añadido `default 'Desconocido'`. Significa que, si no introducimos nada, el sistema automáticamente pondrá que el autor es Desconocido. Así, un libro siempre tiene título, pero no siempre aparece un autor, pensemos en muchos libros de texto o en libros de autor anónimo.

```
1 cantidad smallint unsigned default 0,
```

`smallint` es un tipo de número entero que no admite valores grandes, esto nos sirve para evitar errores de tecleado. Aquí hemos creado un campo llamado cantidad que es un número entero pequeño positivo y que en caso de dejarlo en blanco el sistema pondrá que no hay ninguno. El motivo por el que el default es 0 y no 1, es porque un libro se puede haber extraviado.

```
1 primary key (libro_id)
```

Aquí definimos que la clave primaria de esta tabla sera el id del libro.

Imaginemos ahora que ya existía una tabla llamada libros que no nos interesa. Entonces, habríamos escrito este comando:

```
1 drop table if exists libros;
2 create table libros(
3        libro_id int unsigned auto_increment,
4        titulo varchar(50) not null,
5        autor varchar(30) not null default 'Desconocido',
6        cantidad smallint unsigned default 0,
7       primary key (libro_id)
8  );
```

Lo que hace `drop table if exists libros;` es mirar si existe alguna tabla llamada "libros" dentro de la base de datos "biblioteca" y si así es, eliminarla.

Mostrar las tablas de una base de datos

Para ver la tabla que acabamos de crear, utilizamos este comando:

```
1 show tables;
```

Veremos ahora que aparece "libros" en el resultado.

Mostrar los campos de una tabla

Para inspeccionar los campos de la tabla "libros", usamos el comando explain:

```
1 explain libros;
```

Modificar una tabla

Imaginemos ahora que queremos añadir el campo "editorial" a nuestra tabla "libros". Mañana veremos que es mejor crear una tabla llamada "editorial" y enlazar ambas tablas, pero a título de ejemplo haríamos esto:

```
1 alter table libros add editorial varchar(50) not null;
```

Si ahora ejecutamos el comando `explain libros;` veremos que ha creado un campo llamado editorial.

Imaginemos que en lugar de 50 caracteres sólo admita 20. Lo podemos modificar así

```
1 alter table libros modify column editorial varchar(20) not null;
```

Ya hemos creado nuestra tabla libros con cuatro campos. Ahora vamos a proceder a incluir registros en la tabla.

Insertar registros en la tabla

```
1 insert into libros (titulo, autor, cantidad)
2     values ('Un mundo cualquiera', 'Pepito Frito', 2);
```

Con esta simple sentencia hemos creado nuestro primer registro en MySQL. Lo que le decimos es que inserte dentro de libros "Un mundo cualquiera" en el campo título, "Pepito Frito" en el campo autor y 2 en el campo cantidad.

Imaginemos ahora que hemos comprado un libro más y que queremos actualizar la cantidad a 3, veamos ahora como lo podemos modificar.

Modificar el nombre de una tabla

Si quisiéramos cambiar el nombre de la tabla libros al de fondo_bibliotecario, haríamos lo siguiente:

```
1 rename table libros to fondo_bibliotecario;
```

Modificar registros de una tabla

```
1 update libros set  cantidad=3 where
2       titulo like 'Un mundo cualquiera';
```

Le decimos que actualiza la tabla libros, poniendo en cantidad 3 y que la condición es que el título sea "Un mundo cualquiera". Con `where` estamos estableciendo una condición, si no la ponemos, lo que hará es poner 3 en TODOS los registros, por lo que es muy importante no olvidarse el `where`.

Imaginemos que hay dos libros llamados "Un mundo cualquiera" escritos por diferentes autores. En este caso la sentencia sería:

```
1 update libros set  cantidad=3 where (titulo like 'Un
2                   mundo cualquiera' and autor like &quot;Pepito
Frito&quot;);
```

Aquí hemos hecho servir una cláusula AND, en la que indicamos que se han de cumplir las dos condiciones. El motivo del paréntesis es que si no lo ponemos no tendrá en cuenta la segunda condición.

El comando SELECT

Hay muchas más cosas que decir sobre `select`, pero este es un tutorial básico, por lo que no nos podemos extender. Si quieres saber más, te recomendamos leer la documentación oficial sobre la sintaxis de SELECT.

Para poder ver el potencial de `select`, necesitamos crear más registros:

```
insert into libros (titulo, autor, cantidad) values ('Logra un vientre plano', 'Dolores Barriga', 1);
insert into libros (titulo, autor, cantidad) values ('Aprende a bailar', 'Manolo Descalzo', 1);
```

Veamos ahora nuestro primer comando de `select`:

```
select * from books;
```

Veremos que aparecen todos los libros.

```
select * from libros where cantidad like 1;
```

Nos devuelve dos resultados, el libro de "Dolores Barriga" y el de "Manolo Descalzo".

Para que sólo nos muestre el título y el autor, haremos

```
select titulo, autor from libros;
```

Ahora vamos a ver una tabla de los libros en que sólo se muestra el título y el autor. Veamos ahora como hacemos que sólo muestre los libros de los que sólo tenemos 1:

```
select titulo, autor from libros where cantidad=1;
```

Ahora no aparece el libro de Pepito Frito, ya que de él tenemos 3 en total.

Si queremos seleccionar los libros cuyo autor empiece con "D", haremos lo siguiente:

```
select * from libros where autor like &quot;D% ;
```

El signo % actúa como comodín y significa que da igual lo que venga después de la D. También podemos ponerlo antes. Pongamos que queremos escoger todos los autores cuyo primer apellido sea Descalzo. Haremos esto:

```
select * from libros where autor like &quot;%Descalzo% ;
```

Evitar resultados repetidos

Imaginemos que tenemos varios autores repetidos, pues cada autor ha escrito varios libros. Si queremos mostrar los autores y no ponemos ninguna condición, nos mostrará los resultados repetidos. Para evitarlo, utilizamos `distinct`.

```
select distinct autor from libros;
```

Borrar registros, tablas y bases de datos

Borrar un registro de una tabla

Para borrar un registro, utilizamos el comando `delete from`:

```
1 delete from libros where libro_id=2;
```

Aquí vemos que en lugar de `like` hemos utilizado el signo igual. La diferencia es que el signo igual es mucho más restrictivo.

Borrar todos los registros de una tabla

En el caso de querer borrar todos los registros de una tabla, no utilizaremos el condicional `where`:

```
1 delete from libros;
```

Borrar una tabla

Tal y como hemos visto antes, para borrar una tabla utilizamos el comando `drop table` seguido del nombre de la tabla a borrar:

```
1 drop table libros;
```

Borrar una base de datos

Para borrar una base de datos utilizamos el comando `drop database` seguido del nombre de la base de datos:

```
1 drop database biblioteca;
```

(Toda la Información Anterior, conceptos, imágenes, código, fue obtenido de la pagina: eSandra senior consultant)
Link: http://www.esandra.com/curso-de-mysqlii-instalacion-y-conceptos-basicos-de-mysql/