Tema 3

Bases de datos

Este material ha sido desarrollado en su totalidad por Román Ginés Martínez Ferrández (rgmf@riseup.net) salvo referencias al pie de página.

Sumario

1. Base de datos

Una **base de datos** (BD) es un conjunto de datos relacionados que forman parte de un mismo contexto y que se mantienen para su posterior explotación.

Con el desarrollo de la informática surgieron los denominados **Sistemas Gestores de Bases de Datos** (SGBD) que son programas que permiten añadir, eliminar, modificar y extraer información de una BD.

Estos SGBD proporcionan métodos, entre otras cosas, para:

- Garantizar la integridad de los datos.

- Administrar el acceso de usuarios a los datos.

- Ofrecer la información en diferentes formatos.

Los SGBD utilizan algún modelo de datos para el almacenamiento de la información. Nosotros nos vamos a centrar en los **SGBD Relacionales** que emplean el **Modelo Relacional** en el que la información se almacena en forma de **tabla**.

Las **características del Modelo Relacional** se resumen en el siguiente listado:

- Una BD se compone de varias tablas o relaciones.
- Cada tabla tiene un nombre y no pueden haber dos tablas con el mismo nombre.
- Cada tabla consta de un conjunto de columnas que se denominan campos y filas que denominadas registros.
- Las tablas se relacionan mediante claves primarias y claves ajenas.
- Cada registro consta de una clave primaria que lo identifica del resto.

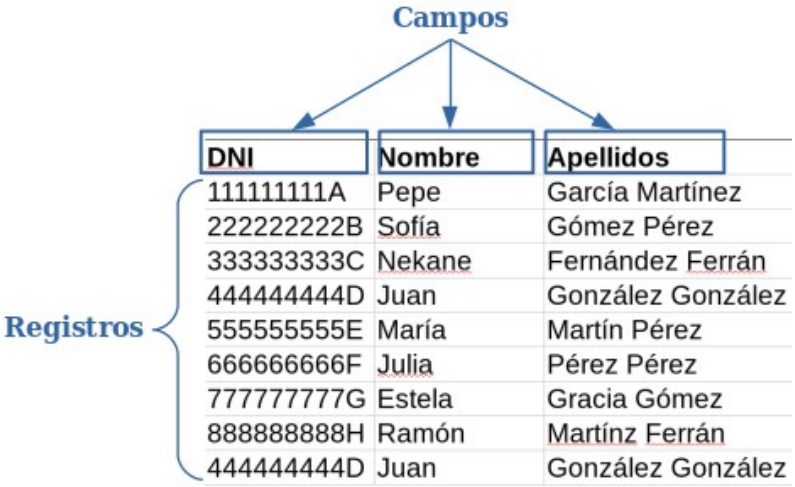

Ilustración 1: tabla con tres campos y nueve registros.

2. Diseño de BBDD Relacionales

Para construir una BD se parte de la descripción de la misma y, a partir de ella, se llevan a cabo los siguientes pasos:

1. Diseño conceptual (Modelo Entidad Relación – ER).
2. Diseño lógico (Modelo Relacional).
3. Implementar en un SGBD mediante SQL.

2.1. Diseño conceptual: Modelo Entidad Relación (ER)

El Modelo ER es una herramienta de diagrama para representar las entidades relevantes de un sistema de información así como sus relaciones y propiedades.

En la Ilustración 3 puedes ver un ejemplo del lenguaje de diagrama que se emplea en el Modelo ER. En él se tienen:

- **Entidades**: en forma de rectángulos.
- **Relaciónes**: mediante un rombo.
- **Propiedades**: dentro de óvalos. Además, la clave primaria se subraya como se puede ver en el ejemplo.
- **Cardinalidad**: indicada mediante el “1” y la “N” (que se lee muchos) al lado de las entidades.

En este ejemplo se tienen dos entidades: Artista y Disco, que se relacionan de la siguiente manera:

- Un artista puede crear muchos discos.
- Un disco puede haber sido creado por un artista y solo uno.

Además, el artista tiene las siguiente propiedades:

- ID: que es la clave primaria de la entidad (por eso se subraya).
- nombre: el nombre del artista.
- apellidos: los apellidos del artista.
- nom_art: el nombre artístico que usa el artista.

Y el disco presenta las siguiente propiedades:

- cod_bar: que es la clave primaria (por eso se subraya).
- titulo: el título del disco.
- fec_pub: la fecha en que se publico el disco.

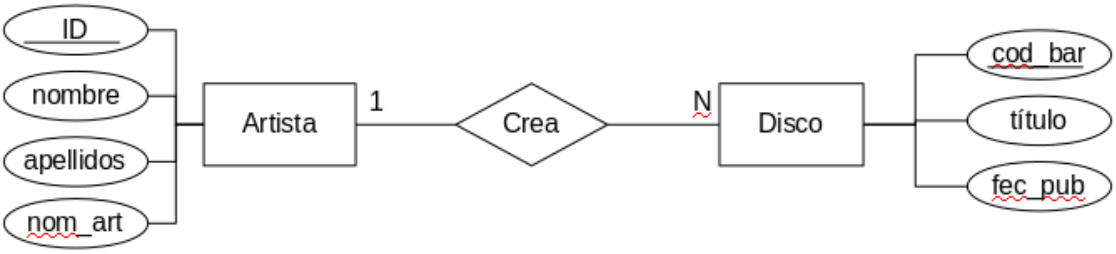

Ilustración 2: ejemplo de modelado mediante un diagrama ER.

Veamos un ejemplo más complejo, con más de dos entidades:

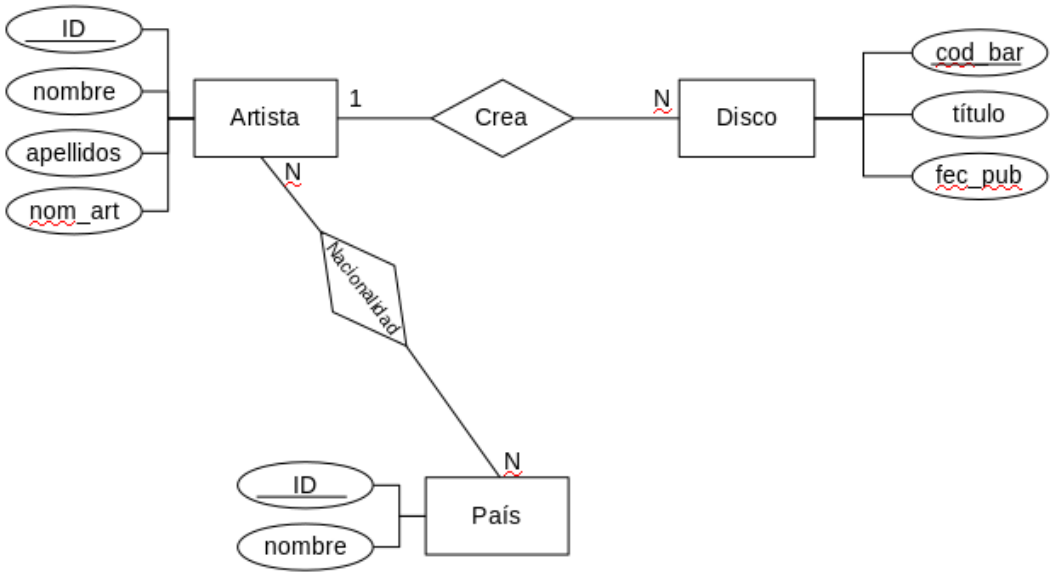

Ilustración 3: diagrama ER con tres entidades y dos relaciones.

En la Ilustración 3 se puede ver el mismo diagrama que en la Ilustración 3 con una entidad más: País, que tiene dos propiedades:

- ID: que es la clave primaria (fíjate cómo se ha subrayado).
- nombre: el nombre del país.

Además, se añade una relación entre Artista y País:

- Un artista puede tener muchas nacionalidades.
- En un país pueden haber muchos artistas.

2.2. Diseño lógico: Modelo Relacional

Se trata del modelo de datos que emplean los SGBD Relacionales y, por tanto, una vez hecho el diseño lógico estamos a un paso de poder implementar la BD.

En el diseño lógico partimos del diseño conceptual para construir el esquema de la base de datos final. En él se indican el nombre de las tablas (relaciones), los campos así como las claves primarias y ajenas.

Veamos cómo convertir de Modelo ER a Modelo Relacional según las cardinalidades de las relaciones de las entidades.

a) Relación 1:N

Veamos con un ejemplo cómo convertir de Modelo ER (conceptual) a Modelo Relacional (lógico) cuando tenemos una relación 1:N.

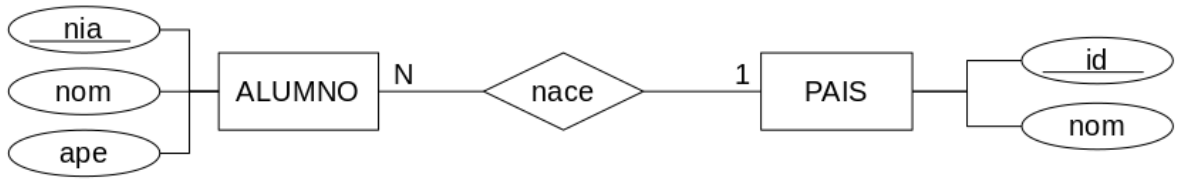

Ilustración 4: Modelo ER con relación 1:N entre dos entidades.

En la Ilustración 4 podemos ver una base de datos diseñada mediante el Modelo ER en el que tenemos dos entidades: ALUMNO y PAIS, que se relacionan mediante un cardinalidad 1:N. Un alumno nace en un país y en un país pueden haber nacido muchos alumnos.

Fíjate en la sintaxis utilizada y cómo:

- Cada entidad del Modelo ER se transforma en una relación en la que, entre paréntesis, se especifican los campos que tiene.
- La entidad con la "N" toma un campo al convertirla a relación que es la clave ajena que apunta a la otra relación.
- Se indica debajo de la definición de la relación qué campo es la clave primaria y qué otros campos son clave ajena. Todas las relaciones tienen una clave primaria pero no necesariamente tienen claves ajenas.

```
ALUMNO (nia, nom, ape, nace)
  CP: nia
  CAj: nace → PAIS

PAIS (id, nom)
  CP: id
```

b) Relación N:N

Veamos qué sucede si la relación entre dos entidades es muchos a muchos (N:N) con el siguiente ejemplo:

Ilustración 5: Modelo ER con relación N:N entre dos entidades.

En este caso tenemos también dos entidades: ALUMNO y ASIGNATURA, en la que se tiene una relación N:N, donde cada alumno puede estar matriculado a muchas asignaturas y en una asignatura puede haber matriculados muchos alumnos.

Al transformar a Modelo Relaciones este caso surgen tres relaciones.

Fíjate en la sintaxis utilizada y cómo:

- Ahora se tienen tres relaciones porque aparece una relación llamada MATRICULADO que en el Modelo ER era la relación.
- La relación que surge toma dos campos que forman la clave primaria y, además, cada una de ellas por separado es una clave ajena a cada una de las tablas de la relación en el Modelo ER.

Claves primarias
Como puedes ver en este ejemplo, las claves primarias pueden ser compuestas. Las claves ajenas también pueden ser compuestas.

```
ALUMNO (nia, nom, ape)
  CP: nia

ASIGNATURA (id, nom)
  CP: id

MATRICULADO (alu, asignatura)
  CP: alu, asignatura
  CAj: alu → ALUMNO
  CAj: asignatura → ASIGNATURA
```

Hay otros casos, relaciones 1:1, relaciones ternarias, etc, pero para este curso es suficiente con los casos 1:N y N:N que, además, son los casos habituales.

c) Implementar en un SGBD mediante SQL

Por último, solo queda implementar la BD. Nos serviremos del diseño lógico (Modelo Relacional) para ello.

Antes tenemos una decisión a tomar: el SGBD, porque depende de esta elección el "dialecto" SQL que hay que usar puede ser diferente.

SQL (Structured Query Language)

> Se trata de un lenguaje usado en los SGBD para crear esquemas de BBDD, añadir registros, actualizarlos, eliminarlos y extraer información.

Nosotros vamos a utilizar MariaDB como Sistema Gestor de Base de Datos. Es software libre, está libre de cargo por licencia y, además, es muy popular y utilizada.

MariaDB es un *fork* de MySQL, así que es muy probable que hagamos alusión al término MariaDB e, incluso, como verás los programas que vamos a utilizar se llaman *mysql* de alguna u otra manera.

3. Structured Query Language (SQL)

Como el SGBD que vamos a utilizar es MariaDB, de aquí en adelante toda la sintaxis SQL que vamos a estudiar funciona en MariaDB y, seguramente, en muchos otros SGBD con pequeñas o sin ninguna diferencia.

3.1. Entrar al intérprete de MariaDB (mysql)

Imagina que tenemos un usuario llamado **`miusuario`** y contraseña **`1234`**. Para entrar al intérprete de MariaDB, abrimos una terminal y ejecutamos la siguiente orden:

```
mysql -u miusuario -p
```

Tras ello, te pedirá una contraseña, introduce **`1234`** y ya estás dentro de MariaDB. Verás que el *prompt* ha cambiado.

3.2. Usar una base de datos

MariaDB, como cualquier otro SGBD, puede gestionar multitud de bases de datos. Así pues, cuando entramos al intérprete de MariaDB debemos indicarle la base de datos sobre la que queremos trabajar. Imagina que deseas trabajar sobre una base de datos llamada **`mibd`**, la orden a ejecutar para usar esta base de datos sería:

```
use mibd;
```

3.3. Crear una base de datos

Por ejemplo, imagina que queremos crear una base de datos llamada **`mibd`**. La orden a ejecutar sería:

```
create database mibd;
```

3.4. Crear un usuario para una base de datos

Al instalar MariaDB se crea un usuario root que no deberíamos usar más que para tareas administrativas. Lo ideal es crear un usuario por cada base de datos.

Imagina que queremos crear un usuario llamado **`miusuario`** con contraseñ **`1234`** con permisos para llevar a cabo todo tipo de tareas sobre la base de datos **`mibd`** creada con anterioridad. Hay que ejecutar dos instrucciones: una para crear el usuario y otra para otorgarle permisos:

```
create user ‘miusuario’@’localhost’ identified by ‘1234’;
```

```
grant all privileges on mibd.* to 'miusuario'@'localhost';
```

3.5. Crear una tabla (relación)

A partir del diseño lógico creamos las relaciones dentro de una base de datos. Lo único que tenemos que considerar antes son los tipos de datos.

a) Tipos de datos

Cada campo que declaramos en una relación tiene que tener un tipo de dato apropiado. Los principales tipos de datos en MariaDB son (hay decenas de tipos de datos, nosotros nos centramos en los que vamos a necesitar):

- integer → para campos que vayan a contener números enteros,
- float → para campos que vayan a contener números reales,
- date → para campos que vayan a contener fechas,

> **Formato de las fechas.**
> Se usa el formato americano: cuatro dígitos para el año seguido de un guión, seguido de dos dígitos para el mes, un guión y dos dígitos para el día. Por ejemplo: **`2017-01-01`**.

- datetime → para campos que vayan a contener fecha y hora,

> **Formato de las horas.**
> Las horas se almacenan de la siguiente manera: dos dígitos para la hora, otros dos para los minutos y otros dos para los segundos, separados por el carácter dos puntos. Por ejemplo: **`17:30:45`**.
> Así, un datetime sería, por ejemplo: **`2017-01-01 17:30:45`**.

- varchar → para campos que vayan a contener cadenas de caracteres. Hay que especificar su longitud entre paréntesis. Por ejemplo, un **`varchar(20)`** podría almacenar hasta 20 caracteres y un **`varchar(100)`** hasta 100 caracteres.

b) Create table a través de ejemplos

Imagina que queremos crear la siguiente relación:

PAIS (id, nombre)

CP: id

A partir de la información que nos da el Modelo Lógico, la crearíamos de la siguiente manera:

```
create table pais (
  id integer primary key,
  nombre varchar(40)
) engine=InnoDB, charset=utf8;
```

Como puedes observar he usado minúsculas tanto para el nombre de la tabla como para el nombre de los campos. Acostúmbrate a seguir siempre un mismo criterio.

Vamos a crear ahora la relación ALUMNO cuyo Modelo Lógico es:

ALUMNO (nia, nombre, apellidos, fec_nac, nacimiento)

CP: nia

A partir de la información que nos da el Modelo Lógico, la crearíamos de la siguiente manera:

```
create table alumno (
  nia integer primary key,
  nombre varchar(10),
  apellidos varchar(30),
  fec_nac date,
  nacimiento integer,
  foreign key (nacimiento) references pais(id)
) engine=InnoDB, charset=utf8;
```

Fíjate muy bien cómo al definir una clave ajena se indica el nombre de la tabla con la que se relaciona y entre paréntesis la clave primaria a la que apunta.

Tras analizar estos ejemplos, puedes concluir la sintaxis del *create table*:

```
create table <nombre de la tabla> (
  <definición de campos>
) engine=InnoDB, charset=utf8;
```

3.6. Insertar registros

Para insertar registros en las relaciones de una BD se utiliza la siguiente sintaxis:

```
insert into <nombre de la tabla> (<lista campos>)
values (<lista valores>);
```

Aprendemos a usar el *insert* a través de ejemplos guiados en clase.

3.7. Eliminar registros

Para eliminar registros de una relación de una BD se utiliza el *delete* como se indica a continuación:

```
delete from <nombre de la tabla>
where <condición>;
```

La cláusula *where* es opcional, si no se incluye se eliminan todos los registros de la relación. Si se incluye el *where* se eliminan los registros que cumplen la condición.

Aprenderemos a usar el *delete* mediante ejercicios guiados que haremos en clase.

3.8. Actualizar registros

Para modificar el valor de un registro usamos el *update* como se indica:

```
update <nombre de la tabla>
set <nuevos valores de campos>
where <condición>;
```

De nuevo, aquí la cláusula *where* es opcional, si no se incluye se actualiza el valor de todos los registros de la relación. Aprenderemos a usar esta sentencia a través de ejercicios guiados en clase.

3.9. Extraer información de las tablas de una base de datos

La instrucción para extraer información de las tablas de una base de datos se denomina **`select`** y consta de tres partes básicas de las cuales dos son obligatorias:

```
select <campos>
from <lista de tablas>
where <condiciones>
order by <campos>
```

El **`where`** es opcional y si no lo ponemos se extrae toda la información de las tablas del **`from`**.

La cláusula **`order by`** también es opcional y se usa para ordenar los resultados que se muestran mediante la lista de campos que se indican.

Vamos a aprender a usar el **`select`** mediante ejemplos guiados que haremos en clase.

4. Miscelánea

En los siguientes apartados se indican algunos comandos útiles del intérprete de MariaDB.

4.1. Mostrar las bases de datos disponibles

```
show databases;
```

4.2. Activar / seleccionar una base de datos llamada mibd

```
use mibd;
```

4.3. Mostrar las tablas tras seleccionar una base de datos

```
show tables;
```

4.4. Ver la descripción (el esquema) de una tabla llamada mitab

```
desc mitab;
```

4.5. Exportar una base de datos a un fichero SQL

Este comando se ejecuta desde la terminal del sistema (NO DESDE LA TERMINAL DE MySQL). En el ejemplo que se ve debajo:

- **2bach** es el nombre de usuario.
- **mibd** es el nombre de la base de datos que deseo exportar.
- **esquema.sql** es el nombre del fichero SQL donde se exporta la base de datos. Esta orden crea un fichero llamado, en este ejemplo, **esquema.sql** en el directorio donde estoy ejecutando la orden **mysqldump**.

```
mysqldump -u 2bach -p mibd > esquema.sql
```

La opción **-p** se introduce para que **mysqldump** nos pida la contraseña del usuario **2bach**. Es obligatoria esta opción si dicho usuario tiene una contraseña.

Por otro lado, el usuario **2bach** debe tener permisos suficientes sobre la base de datos **mibd**.

4.6. Importar una base de datos desde un fichero SQL

En este apartado vamos a mostrar cómo hacer la operación inversa a la del apartado anterior: tenemos un fichero SQL y queremos volcarlo a una base de datos.

Este fichero SQL puede tener todo tipo de sentencias SQL: create tables, inserts, etc.

En el ejemplo que se ve debajo:

- **2bach** es el nombre de usuario.
- **mibd** es el nombre de la base de datos sobre la que quier volcar el fichero **esquema.sql** (ejecutar las sentencias SQL que hay en el fichero).
- **esquema.sql** es el nombre del fichero SQL donde están las sentencias SQL.

```
mysql -u 2bach -p mibd < esquema.sql
```

PRESTA ATENCIÓN A DOS DETALLES MUY IMPORTANTES:

- La orden no es mysqldump. En este caso es mysql.
- El signo que se usa es el de “menor que”. En el caso anterior se empleaba el de “mayor que”.