2. SQL/DDL

Vamos a aprender cómo utilizar las sentencias `CREATE`, `ALTER`, `DROP`, `RENAME`, `TRUNCATE`, `COMMENT`, `SHOW` y `DESCRIBE` del lenguaje de definición de datos (DDL, Data Definition Language).

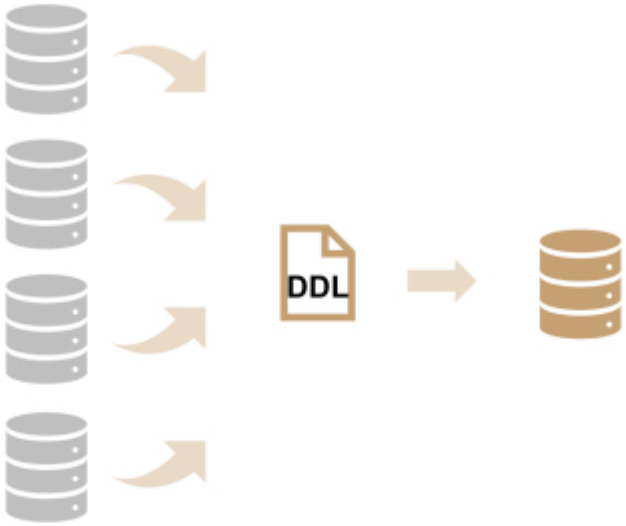

Fig. 6.2. Las tablas de nuestras bases de datos serán creadas con DDL sobre el SGBDR.

Vamos a ir viendo la descripción, su sintaxis y algún ejemplo con las bases de datos de *MCGRAWCINES*, *MCGRAWPELICULAS*, *MCGRAWINSTITUTO* y *MCGRAWHOTELES* que hemos ido diseñando y normalizando. Puedes descargar estas bases de datos en la sección **Ejemplos para trabajar y analizar,** al inicio del curso**.** Ahora es el momento de ir creándolas y utilizándolas desde SQL.

A. CREATE

Vamos a ir viendo la descripción, su sintaxis y algún ejemplo con las bases de datos de MCGRAWCINES, MCGRAWPELICULAS, MCGRAWINSTITUTO y MCGRAWHOTELES que hemos ido diseñando y normalizando. **Todas ellas las tienes en la documentación complementaria de la unidad.** Ahora es el momento de ir creándolas y utilizándolas desde SQL.

La instrucción `CREATE` es una potente instrucción que se utiliza para crear desde la base de datos hasta los siguientes objetos de esta, como tablas (tables), índices (index), funciones (functions), vistas (views), procedimientos de almacenamiento (storage procedures) y desencadenadores (triggers). En esta unidad nos vamos a centrar en la BBDD y en las tablas, así como en alguno de los índices.

A1. CREATE (USE) DATABASE

Empecemos creando la base de datos, que es muy simple:

Descripción
La instrucción `CREATE` **`DATABASE`** se utiliza para crear la base de datos. La instrucción `USE` sirve para indicar con qué base de datos vamos a trabajar.

Sintaxis básica

```
CREATE DATABASE nombre_base_datos;
USE nombre_base_datos;
```

Ejemplo

```
CREATE DATABASE mcgrawpeliculas;
CREATE DATABASE mcgrawinstituto;
USE mcgrawpeliculas;
```

Con cada una de estas instrucciones hemos creado una base de datos en el SGBDR con el nombre que le hemos indicado. Si queremos indicar sobre qué base de datos vamos a trabajar, en el caso de tener más de una, tendremos que indicarlo con la instrucción USE. Si utilizamos una herramienta gráfica o web como MySQL Workbench o phpMyAdmin simplemente tendremos que elegir la base de datos de la lista; lo veremos en los casos prácticos ampliados.

Tabla 6.9. Sintaxis de la sentencia `CREATE (USE) DATABASE`.

A2. CREATE TABLE

Cuando ya disponemos de la base de datos y la tenemos elegida, podemos empezar a crear sus tablas.

IMPORTANTE

Aunque hoy en día con el formato UNICODE se pueden utilizar acentos o letras especiales como la Ñ, que no están en el alfabeto inglés, y por tanto en el ASCII de 7 bits, se recomienda no poner acentos en los nombres de las tablas o campos para no cometer errores luego a la hora de utilizarlas. Si no ponemos nunca sabremos siempre cómo está escrito.

Descripción

La instrucción **`CREATE TABLE`** se utiliza para **crear tablas** de una base de datos. Al crear una tabla debemos especificar los nombres de los atributos, los tipos y todas las demás características que tengamos de nuestro modelo relacional normalizado.

Sintaxis básica

```
CREATE TABLE nombre_tabla (
 <nombre_campo1> <tipo_de_dominio1>[ (tamaño) ]    [NOT NULL]
[índice1],
```

```
[<nombre_campo2> <tipo_de_dominio2>[ (tamaño) ] ]  [NOT NULL]
[índice2], ]
[ , ...]  [, CONSTRAINT <restricciones> [, ...] ] ) [
<opciones_de_la_base_datos> ];
```

Ejemplo: vamos a crear la tabla ACTOR de la base de datos MCGRAWPELICULAS. Recordemos que su estructura es: actor (#idActor, nombreActor, nacionalidadActor)

```
CREATE TABLE actor (
  idActor int(11) PRIMARY KEY,
  nombreActor varchar(30) COLLATE utf8_spanish2_ci NOT NULL,
  nacionalidadActor int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish2_ci;
```

El argumento `PRIMARY KEY` nos creará la clave primaria de la tabla. Si estuviera compuesta de varios campos se pondrá al final `PRIMARY KEY` (idActor, idNacionalidad). Si en vez de ser un campo la nacionalidad necesitáramos tener más datos en otra tabla nacionalidad, en este caso, para crear una clave ajena sobre esta tabla nacionalidad tendríamos que definirla así:

idNacionalidad smallint(4) unsigned references nacionalidad(idNacionalidad).

Si ese campo, en nuestro caso idNacionalidad, no fuera clave primaria en su tabla nos haría falta crear un índice sobre ese campo de la siguiente manera:

index(idNacionalidad)

Por último, para crear clave ajena compuesta por más de un campo lo haremos así:

foreign key (idNacionalidad, idPaís) references nacionalidad (idNacionalidad, idPaís));

El argumento `NOT NULL` nos indica que ese campo no admite valores nulos.

El argumento `COLLATE` nos indica en el formato que se va a almacenar ese texto.

Ejemplo: vamos a crear la tabla ALUMNADO de la base de datos MCGRAWINSTITUTO. Recordemos que su estructura es: alumnado (#NIA, nombreAlumnado, apellidosAlumnado)

```
CREATE TABLE alumnado (
  NIA int(11) PRIMARY KEY,
  nombreAlumnado varchar(30) COLLATE utf8_spanish2_ci NOT NULL,
  apellidosAlumnado varchar(30) COLLATE utf8_spanish2_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish2_ci;
```

Tabla 6.10. Sintaxis de la sentencia `CREATE TABLE`.

IMPORTANTE

En las opciones_base_datos vemos que utilizamos el motor InnoDB del SGBDR de Oracle MySQL, de MariaDB o de Amazon Aurora, que son proyectos derivados del primero y, por tanto, compatibles entre ellos, indicándole que el juego de caracteres está codificado con el formato de caracteres UNICODE dinámico de 8 bits denominado UTF8.

Para simplificar la notación hemos eliminado algunos símbolos que se utilizan para acotar los nombres de tablas o campos/columnas, etcétera. Cuando exportemos las sentencias a un script .sql nos encontraremos que cada SGBDR utilizará el símbolo del acento abierto ` u otro para delimitarlos.

```
CREATE TABLE `alumnado` (
  `NIA` int(11) NOT NULL,
  `nombreAlumnado` varchar(30) COLLATE utf8_spanish2_ci NOT NULL,
  `apellidosAlumnado` varchar(30) COLLATE utf8_spanish2_ci NOT
NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish2_ci;
```

Para averiguar la sintaxis exacta de una instrucción y, si tenemos que poner comillas (y qué tipo) en nombres de tabla, columnas, etcétera, siempre es mejor ejecutar la instrucción correspondiente en phpmyadmin y ver qué código SQL nos genera.

A partir de este momento, vamos a omitir las instrucciones `ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish2_ci` para simplificar los ejemplos. Igual que `COLLATE utf8_spanish2_ci` en todos los campos varchar().

A3. CREATE INDEX

Cuando ya disponemos de nuestras tablas podemos crearles índices:

Descripción
La instrucción **`CREATE INDEX`** se utiliza para **crear índices en nuestras tablas**.
Sintaxis básica

```
CREATE [UNIQUE] INDEX nombre_índice
ON nombre_de_la_tabla(
<nombre_campo1>[DESC][, <nombre_campo2> [DESC][, ...]])
[WITH {PRIMARY | DISALLOW NULL | IGNORE NULL}];
```

- El argumento `UNIQUE` impide que el campo restringido acepte valores duplicados.
- El argumento `DESC` crea el índice en orden descendente. Si lo omitimos de forma predeterminada, un índice se crea en orden ascendente.
- El argumento `WITH PRIMARY` establece qué campos indexados formarán la clave principal de la tabla. Lo más habitual es que sea un único campo.
- El argumento `WITH DISALLOW NULL` no permite valores nulos en el índice.

Ejemplo: vamos a crear un índice de clave primaria sobre el campo idActor de la tabla actor de la base de datos MCGRAWPELICULAS.

```
CREATE INDEX idActorPK
ON actor (
  idActor
) WITH PRIMARY;
```

Ejemplo: vamos a crear un índice de unicidad sobre el campo NIA de la tabla ALUMNADO de la base de datos MCGRAWINSTITUTO.

```
CREATE UNIQUE INDEX índice_valor_único_NIA
ON alumnado (
  NIA
) WITH DISALLOW NULL;
```

Con el argumento `UNIQUE` nos aseguramos de que no habrá valores duplicados, y con `WITH DISALLOW NULL` también nos aseguraremos de que no exista ningún valor nulo.

Tabla 6.11. Sintaxis de la sentencia `CREATE INDEX`.