Bases de datos - UD3. Ejercicios. Diseño Físico y Edición de Datos

Descargar estos ejercicios

Índice

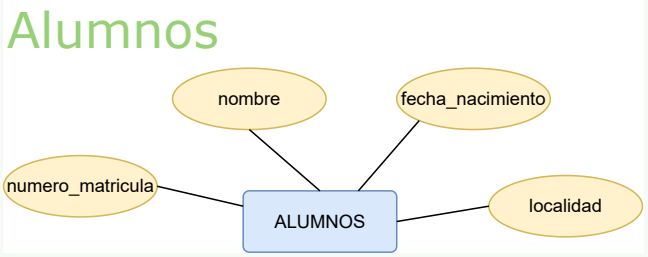
Creación de tablas

Se valorará además de que la solución de los ejercicios sea correcta, las tabulaciones, los comentarios en el código, nombres de elementos (tablas y campos) en minúscula y
la claridad del código en general.

✅ Ejercicio 1

Alumnos

A partir del esquema conceptual (modelo entidad-relación) siguiente



Se obtiene el siguiente esquema lógico (modelo relacional):

ALUMNOS (numero_matricula , nombre , fecha_nacimieno , direccion , localidad)

Escribe la instrucción en SQL para crear la tabla ALUMNOS teniendo en cuenta las siguientes restricciones:

- Restricciones del esquema ER:
  - Clave primaria: numero_matricula
- Restricciones del diccionario de datos:
  - El atributo nombre no puede tomar valores nulos

💡 ***Solución***

```
CREATE DATABASE ejercicios;
USE bd_ejercicios;

CREATE TABLE ALUMNOS (
    numero_matricula INT(6) PRIMARY KEY,
    nombre VARCHAR(15) NOT NULL,
    fecha_nacimiento DATE,
    direccion VARCHAR(30),
    localidad VARCHAR(50)
);
```

✅ Ejercicio 2

Alumnos

Intenta crear de nuevo la tabla “ALUMNOS” de la actividad anterior. ¿Qué ocurre?

💡 ***Solución***

Table 'ALUMNOS' already exists

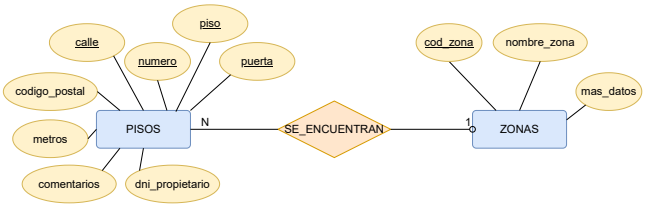
SQL devuelve un error, indicando que la tabla ya existe y no se puede volver a crear.

✅ Ejercicio 3

Pisos y zonas

A partir del esquema conceptual (modelo entidad-relación) siguiente

Pisos y zonas



Escribe las instrucciones en SQL para crear las tablas obtenidas en el esquema lógico (del modelo relacional).

Primero crearemos la tabla ZONAS, ya que PISOS hace referencia a dicha tabla.

Recuerda que otra opción sería:

- Crear primero la tabla PISOS sin clave ajena
- Crear la tabla ZONAS
- Añadir la restricción de clave ajena a la tabla PISOS.

ZONAS (cod_zona <entero 2>, nombre_zona <caracter 20>, mas_datos <caracter 50>)

Restricciones del esquema ER:

- Clave primaria: cod_zona
  - Observa que la clave primaria está formada por el atributo cod_zona, y que aunque no lo especifiquemos como NOT NULL automáticamente al formar parte de la clave el SGBD coloca NOT NULL en este atributo.
  - Como la clave primaria está formada por una única columna podemos poner la restricción a continuación de la definición de dicha columna o al final de la tabla. En este caso la pondremos junto a la definición de la columna.

PISOS (calle <caracter 30>, numero <entero 3>, piso <entero 2>, puerta <caracter 1>, codigo_postal <entero 5>, metros <entero 5>, comentarios <caracter 60>, **cod_zona* <entero 2>**, dni_propietario <caracter 9>)

Restricciones del esquema ER:

- Clave primaria: calle, numero, piso, puerta. (Nombra a esta restricción como pk_viv)
  - Observa que la clave primaria está formada por los atributos calle, número, piso y puerta y que aunque no los especifiquemos como NOT NULL automáticamente al formar parte de la clave coloca NOT NULL en esos atributos.
  - Recuerda que como la clave primaria está formada por un grupo de columnas hemos de definirla al final de la tabla.
- Clave ajena: cod_zona → ZONAS
- De momento no incluimos las restricciones por ON DELETE o por ON UPDATE. ES decir, de momento no se especifica qué debe hacer la base de datos automáticamente cuando se elimine o actualice un registro relacionado en una clave foránea.

💡 ***Solución***

```sql
CREATE TABLE ZONAS (
    cod_zona INT(2) PRIMARY KEY,
    nombre_zona VARCHAR(20),
    mas_datos VARCHAR(50)
);

CREATE TABLE PISOS (
    calle VARCHAR(30),
    numero INT(3),
    piso INT(2),
    puerta CHAR(1),
    codigo_postal INT(5),
    metros INT(5),
    comentarios VARCHAR(60),
    cod_zona INT(2),
    dni_propietario CHAR(9),
    CONSTRAINT pk_viv PRIMARY KEY (calle, numero, piso, puerta),
    FOREIGN KEY (cod_zona) REFERENCES ZONAS(cod_zona)
);
```

✅ Ejercicio 4

Partiendo del esquema lógico, crea la tabla de **proveedores** de nuestra comunidad autónoma. La estructura de la tabla es la siguiente:
proveedores(`cod_prov` , `localidad` , `nombre` , `provincia`)
El diccionario de datos (sólo con restricciones de valor) es:

Tabla proveedores

CAMPO	TIPO	LONGITUD	CARACTERÍSTICAS
cod_prov	Cadena	3	Clave Primaria No Nulo
localidad	Cadena	100	Nulo
nombre	Cadena	100	No nulo Único Clave Alternativa
provincia	Enumerado		'Alicante', 'Valencia', 'Castellón'

💡 ***Solución***

```sql
CREATE TABLE proveedores (
    cod_prov    VARCHAR(3) PRIMARY KEY,
    localidad   VARCHAR(100) DEFAULT NULL,
    nombre      VARCHAR(100) NOT NULL UNIQUE,
    provincia   ENUM('Alicante','Valencia','Castellón')
);
```

✅ Ejercicio 5

En XAMPP-MySQL mediante la utilidad `mysql.exe` ejecuta las instrucciones anteriores.

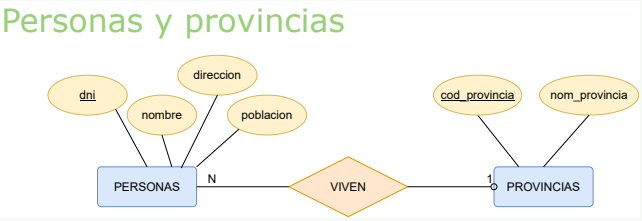
Entra a **phpMyAdmin** y comprueba con el asistente visual la estructura de la tabla y crea otra llamada **proveedores2** con la misma información con la herramienta gráfica.
Pulsando en la pestaña *Insertar*, añade registros a la tabla **proveedores2**.

Comprueba con el botón *Examinar* los datos introducidos. Comprobarás que aparece un botón *Editar* y otro *Borrar* que afecta a cada registro.

✅ Ejercicio 6

Personas y provincias

Del esquema conceptual (del modelo entidad-relación) siguiente:



Escribe las instrucciones en SQL para crear las tablas obtenidas en el esquema lógico (del modelo relacional).

La tabla PERSONAS contiene datos sobre las personas de una comunidad, mientras que la tabla PROVINCIAS contiene el código y el nombre de cada provincia, se relacionan por el atributo codprovin.

En primer lugar crearemos la tabla PROVINCIAS y, después la tabla PERSONAS, ya que PERSONAS referencia a PROVINCIAS. Si creamos primero la tabla PERSONAS y la tabla PROVINCIAS no está creada, MySQL emitirá un mensaje de error.

Recuerda que otra opción sería:

- Crear primero la tabla PERSONAS sin clave ajena
- Crear la tabla PROVINCIAS
- Añadir la restricción de clave ajena a la tabla PERSONAS

Cuando crees las tablas, hazlo sin asignar nombre a las restricciones de clave primaria ni tampoco a las de clave ajena.

PROVINCIAS (cod_provincia <entero 2>, nom_provincia <caracter 25>)

Restricciones del esquema ER:

- Clave primaria: cod_provincia

PERSONAS (dni <entero 8>, nombre <caracter 15>, direccion <caracter 25>, poblacion <caracter 20>, **cod_provin* <entero 2>**)

Restricciones del esquema ER:

- Clave primaria: DNI.
- Clave ajena: cod_provin → PROVINCIAS.

- Como la clave ajena está formada por un único atributo se puede definir tanto a continuación de la definición de la columna cod_provin como al final de la definición de la tabla. En este caso, lo haremos a continuación de la definición de la columna y no especificaremos ninguna política ante borrados para asegurar la integridad referencial.
- De momento no incluimos las restricciones por ON DELETE o por ON UPDATE

💡 ***Solución***

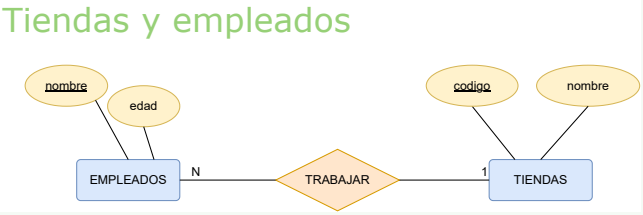
```
CREATE TABLE PROVINCIAS (
    cod_provincia INT(2) PRIMARY KEY,
    nom_provincia VARCHAR(25)
);

CREATE TABLE PERSONAS (
    dni INT(8) PRIMARY KEY,
    nombre VARCHAR(15),
    direccion VARCHAR(25),
    poblacion VARCHAR(20),
    cod_provin INT(2),
    FOREIGN KEY (cod_provin) REFERENCES PROVINCIAS(cod_provincia)
);
```

✅ Ejercicio 7

Tiendas y empleados

Del esquema conceptual (del modelo entidad-relación) siguiente:



Escribe las instrucciones en SQL para crear las tablas obtenidas en el esquema lógico (del modelo relacional).

Primero crearemos la tabla TIENDAS, ya que EMPLEADOS hace referencia a dicha tabla.

TIENDAS (codigo <entero 2>, nombre <caracter 25>)

Restricciones del esquema ER:

- Clave primaria: codigo

EMPLEADOS (nombre <caracter 25>, edad <entero 3>, **cod_tienda* <entero 2>**)

Restricciones del esquema ER:

- Clave primaria: nombre
- Clave ajena: cod_tienda → TIENDAS

Restricciones del diccionario de datos:

- Al borrar la fila asociada a una tienda deseamos que se eliminen automáticamente las filas con claves ajenas que la referencien, es decir, que se eliminen todos los empleados de esa tienda.
- Verificación de condiciones: edad comprendida entre 18 y 65.

💡 ***Solución 1***

```
CREATE TABLE TIENDAS (
    codigo INT(2) PRIMARY KEY,
    nombre VARCHAR(25)
);

CREATE TABLE EMPLEADOS (
    nombre VARCHAR(25) PRIMARY KEY,
    edad INT(3) CHECK (edad BETWEEN 18 AND 65),
    cod_tienda INT(2),
    FOREIGN KEY (cod_tienda) REFERENCES TIENDAS(codigo)
        ON DELETE CASCADE
);
```

Otra opción sería:

- Crear primero la tabla EMPLEADOS sin clave ajena
- Crear la tabla TIENDAS
- Añadir la restricción de clave ajena a la tabla EMPLEADOS.

💡 ***Solución 2***

```
CREATE TABLE EMPLEADOS (
    nombre VARCHAR(25) PRIMARY KEY,
    edad INT(3) CHECK (edad BETWEEN 18 AND 65),
    cod_tienda INT(2)
);

CREATE TABLE TIENDAS (
    codigo INT(2) PRIMARY KEY,
    nombre VARCHAR(25)
);

ALTER TABLE EMPLEADOS
ADD FOREIGN KEY (cod_tienda) REFERENCES TIENDAS(codigo)
ON DELETE CASCADE;
```

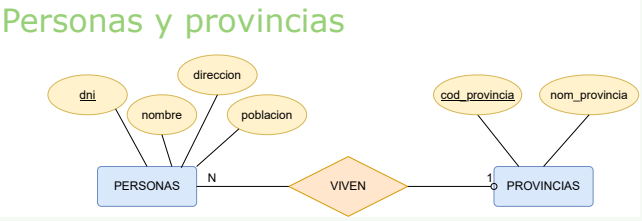
Edición de datos

Para probar las soluciones crearemos bases de datos con phpMyAdmin en XAMPP.

✅ Ejercicio 8

Personas y provincias

Anteriormente hemos creado las tablas PROVINCIAS y PERSONAS.




```
CREATE TABLE PROVINCIAS (
    cod_provincia INT(2) PRIMARY KEY,
    nom_provincia VARCHAR(25)
);

CREATE TABLE PERSONAS (
    dni INT(8) PRIMARY KEY,
    nombre VARCHAR(15),
    direccion VARCHAR(25),
    poblacion VARCHAR(20),
    cod_provin INT(2),
    FOREIGN KEY (cod_provin) REFERENCES PROVINCIAS(cod_provincia)
);
```

Ahora vamos a insertar las filas que aparecen a continuación en las tablas. Para ello, vamos a utilizar la orden INSERT de la siguiente forma:

```
INSERT INTO nombre_Tabla [(columna [, columna]...)]
VALUES (valor [, valor]...);
```

Por ejemplo, para insertar la primera fila de la tabla PROVINCIAS escribiríamos la siguiente sentencia:

```
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia)
VALUES (1, 'Alicante');
```

PROVINCIAS

cod_provincia	nom_provincia
1	Alicante
2	Valencia
3	Castellón
43	Barcelona
55	Huesca

PERSONAS

dni	nombre	direccion	poblacion	cod_provin
56783412	Pepe	Avenida de las Américas	Castellón	3
34556784	Ramón	Calle Poeta Quintana	Alicante	1
12334523	Silvia	Calle Pintor Aparicio	Alicante	1
98564321	Juan	Avenida de la Libertad	Valencia	2
90765678	Bartolo	Calle Rosa Chacel	Alicante	1
45676565	Isabel	Calle Paraguay	Barcelona	43
23454322	Juan	Avenida de Alfonso X el Sabio	Alicante	1
67424563	Bernardo	Plaza Imperial	Huesca	55

💡 ***Solución 1***

Inserciones en la tabla PROVINCIAS

```
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (1, 'Alicante');
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (2, 'Valencia');
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (3, 'Castellón');
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (43, 'Barcelona');
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (55, 'Huesca');
```

Inserciones en la tabla PERSONAS

```
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (56783412, 'Pepe', 'Avenida de las Américas', 'Cas
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (34556784, 'Ramón', 'Calle Poeta Quintana', 'Alica
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (12334523, 'Silvia', 'Calle Pintor Aparicio', 'Ali
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (98564321, 'Juan', 'Avenida de la Libertad', 'Vale
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (90765678, 'Bartolo', 'Calle Rosa Chacel', 'Alicar
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (45676565, 'Isabel', 'Calle Paraguay', 'Barcelona'
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (23454322, 'Juan', 'Avenida de Alfonso X el Sabio'
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (67424563, 'Bernardo', 'Plaza Imperial', 'Huesca',
```

Ahora insertaremos en la tabla PERSONAS una fila dando al campo **cod_provin** un valor que no exista en provincias. Por ejemplo, prueba a insertar una persona que pertenezca a la provincia de Teruel de código 35. Explica qué ocurre.

💡 ***Solución 2***

```
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin)
VALUES (11223344, 'Lucía', 'Calle Mayor', 'Teruel', 35);
```

¿Qué ocurre?

La inserción fallará y el SGBD devolverá un error ya que se produce una violación de la *integridad referencial*: El campo PERSONAS.cod_provin es una clave foránea a la tabla PROVINCIAS (campo PROVINCIAS.cod_provincia), la provincia 35 no tiene un valor asociado en la tabla PROVINCIAS.

Solución

Antes de realizar la inserción en PERSONAS, dar de alta la provincia 35 en PROVINCIAS.

```
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (35, 'Teruel');
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (11223344, 'Lucía', 'Calle Mayor', 'Teruel', 35);
```

✅ Ejercicio 9

Personas y provincias

En este ejercicio partimos de la situación en que las tablas PROVINCIAS y PERSONAS tienen datos, ya que los hemos insertado en el ejercicio anterior.

Vamos a borrar todas las filas de la tabla PROVINCIAS. Recuerda que para ello podemos utilizar la orden DELETE. Observa que se produce un error. ¿Por qué?

> 💡 ***Solución***
>
> Borrado de todas las filas de la tabla PROVINCIAS
>
> ```
> DELETE FROM PROVINCIAS;
> ```
>
> **¿Qué ocurre?**
> El borrado fallará y el SGBD devolverá un error ya que se produce una *violación de la integridad referencial*: Hay campos de la tabla PERSONAS (PERSONAS.cod_provincia) que dependen de PROVINCIAS (PROVINCIAS.cod_provincia), PROVINCIAS.cod_provincia es una clave foránea a PERSONAS.cod_provincia

Para solucionarlo, borra la tabla PERSONAS con la orden DROP y vuelve a crearla con la restricción adecuada (ON DELETE y ON UPDATE) para que se mantenga la integridad referencial.

> 💡 ***Solución***
>
> ```
> -- Eliminar la tabla PERSONAS
> DROP TABLE PERSONAS;
>
> -- Volver a crearla con la restricción ON DELETE CASCADE
> CREATE TABLE PERSONAS (
>     dni INT(8) PRIMARY KEY,
>     nombre VARCHAR(15),
>     direccion VARCHAR(25),
>     poblacion VARCHAR(20),
>     cod_provin INT(2),
>     FOREIGN KEY (cod_provin) REFERENCES PROVINCIAS(cod_provincia) ON DELETE CASCADE
> );
>
> -- También se podría incluir ON UPDATE CASCADE
> ```

Ahora, vuelve a insertar las filas en la tabla PERSONAS.
Borra una fila de la tabla PROVINCIAS ¿Qué ocurre en la tabla PERSONAS?

> 💡 ***Solución***

```
-- Inserciones en  la tabla PERSONAS
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (56783412, 'Pepe', 'Avenida de las Américas', 'Cas
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (34556784, 'Ramón', 'Calle Poeta Quintana', 'Alica
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (12334523, 'Silvia', 'Calle Pintor Aparicio', 'Ali
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (98564321, 'Juan', 'Avenida de la Libertad', 'Vale
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (90765678, 'Bartolo', 'Calle Rosa Chacel', 'Alicar
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (45676565, 'Isabel', 'Calle Paraguay', 'Barcelona'
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (23454322, 'Juan', 'Avenida de Alfonso X el Sabio'
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (67424563, 'Bernardo', 'Plaza Imperial', 'Huesca',
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (11223344, 'Lucía', 'Calle Mayor', 'Teruel', 35);

--- Borrado de una fila de la tabla PROVINCIAS
DELETE FROM PROVINCIAS WHERE cod_provincia = 1;

--- ¿Qué ocurre con la tabla PERSONAS?
SELECT * FROM PERSONAS;
```

dni	nombre	direccion	poblacion	cod_provin
56783412	Pepe	Avenida de las Américas	Castellón	3
98564321	Juan	Avenida de la Libertad	Valencia	2
45676565	Isabel	Calle Paraguay	Barcelona	43
67424563	Bernardo	Plaza Imperial	Huesca	55
11223344	Lucía	Calle Mayor	Teruel	35

¿Qué ocurre en la tabla PERSONAS?
Al eliminar de la tabla PROVINCIAS la provincia con cod_provincia=1, el SGBD ha eliminado de la tabla PERSONAS todos los registros en los que cod_provincia=1 (ON DELETE CASCADE)

Ahora intenta insertar una fila en la tabla PROVINCIAS cuyo código de provincia (clave primaria) ya existe. ¿Qué ocurre?. Observa el nombre que MySQL le da a la restricción que ha sido violada.

💡 ***Solución***

```
INSERT INTO PROVINCIAS (cod_provincia, nombre) VALUES (2, 'Castelló');
```

MySQL devuelve un error indicando que ya existe un registro en la tabla con esea clave primaria.

```
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

A continuación, intenta insertar una fila en la tabla PERSONAS cuyo código de provincia (clave ajena) no existe en la tabla PROVINCIAS. ¿Qué ocurre?. Observa el nombre que MySQL le da a la restricción que ha sido violada.

💡 ***Solución***

```
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (12345678, 'Pepe', 'Paseo de la Castellana', 'Madr
```

MySQL devuelve un error indicando que no existe un registro con cod_provincia=99 en la tabla PROVINCIAS (violación integridad referencial).

```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`nombre_base`.`PERSONAS`, CONSTRAINT `pe
```

✅ Ejercicio 10

Personas y provincias

En esta actividad vas a borrar las tablas PERSONAS y PROVINCIAS; y vas a crearlas de nuevo dando nombre a las restricciones de clave ajena.

💡 ***Solución***

Debemos eliminar las tablas en el orden correcto (primero personas y luego provincias)

```
DROP TABLE IF EXISTS PERSONAS;
DROP TABLE IF EXISTS PROVINCIAS;
```

Y volver a crearlas, dando nombre a las restricciones (CONSTRAINT)

```
CREATE TABLE PROVINCIAS (
    cod_provincia INT(2),
    nombre VARCHAR(20),
    CONSTRAINT pk_provincias PRIMARY KEY (cod_provincia)
);

CREATE TABLE PERSONAS (
    dni INT(8) PRIMARY KEY,
    nombre VARCHAR(15),
    direccion VARCHAR(25),
    poblacion VARCHAR(20),
    cod_provin INT(2),
    CONSTRAINT fk_personas_provincias
        FOREIGN KEY (cod_provin)
        REFERENCES PROVINCIAS(cod_provincia)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

A continuación inserta algunas filas en la tabla PROVINCIAS para hacer que se viole la restricción de clave primaria. Para ello, deberás insertar un código de provincia
existente. ¿Qué instrucción has ejecutado? ¿Qué error devuelve?

💡 ***Solución***

```
-- Inserciones en  la tabla PROVINCIAS
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (1, 'Alicante');
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (2, 'Valencia');
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (3, 'Castellón');
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (43, 'Barcelona');
INSERT INTO PROVINCIAS (cod_provincia, nom_provincia) VALUES (55, 'Huesca');


-- Violación restricción clave primaria
INSERT INTO PROVINCIAS (cod_provincia, nombre) VALUES (2, 'Castelló');

-- Error devuelto
ERROR 1062 (23000): Duplicate entry '2' for key 'pk_provincias'
```

Ahora inserta una fila en la tabla PERSONAS cuyo código de provincia no exista en la tabla PROVINCIAS. Observa que se produce un error en la restricción de clave ajena que habrás nombrado previamente al crear la tabla. ¿Qué instrucción has ejecutado? ¿Qué error devuelve? ¿Cómo has llamado a la restricción?

💡 ***Solución***

```
-- Inserciones en  la tabla PERSONAS
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (56783412, 'Pepe', 'Avenida de las Américas', 'Cas
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (34556784, 'Ramón', 'Calle Poeta Quintana', 'Alica
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (12334523, 'Silvia', 'Calle Pintor Aparicio', 'Ali
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (98564321, 'Juan', 'Avenida de la Libertad', 'Vale
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (90765678, 'Bartolo', 'Calle Rosa Chacel', 'Alicar
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (45676565, 'Isabel', 'Calle Paraguay', 'Barcelona'
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (23454322, 'Juan', 'Avenida de Alfonso X el Sabio'
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (67424563, 'Bernardo', 'Plaza Imperial', 'Huesca',
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (11223344, 'Lucía', 'Calle Mayor', 'Teruel', 35);

-- Violación restricción clave foránea / clave ajena
INSERT INTO PERSONAS (dni, nombre, direccion, poblacion, cod_provin) VALUES (12345678, 'Pepe', 'Paseo de la Castellana', 'Madr

-- Error devuelto
ERROR 1452 (23000): Cannot add or update a child row:
a foreign key constraint fails (`test`.`PERSONAS`,
CONSTRAINT `fk_personas_provincias` FOREIGN KEY (`cod_provin`)
REFERENCES `PROVINCIAS` (`cod_provincia`))
```

✅ Ejercicio 11

Ejemplo 1

Crea la siguiente tabla en una base de datos nueva.

EJEMPLO1 (nif <caracter 10>, nombre <caracter 30>, edad <entero 2>)

Restricciones del esquema ER:

- Clave primaria: nif

Restricciones del diccionario de datos:

- El atributo nombre no pueda tomar valores nulos.

Solución

```
-- Creación base de datos
CREATE DATABASE IF NOT EXISTS bd_ejemplo1;
USE bd_ejemplo1;

-- Creación tabla
CREATE TABLE EJEMPLO1 (
    nif VARCHAR(10) PRIMARY KEY,
    nombre VARCHAR(30) NOT NULL,
    edad INT(2)
);
```

Inserta una fila en la tabla EJEMPLO1 de forma que el campo nombre no tome ningún valor. ¿Qué instrucción has ejecutado? ¿Qué ocurre?.

Solución

```
-- Inserción con campo nombre vacío
INSERT INTO EJEMPLO1 (nif, edad) VALUES ('12345678', 25);

-- Error devuelto
ERROR 1364 (HY000): Field 'nombre' doesn't have a default value
```

Aviso

```
INSERT INTO EJEMPLO1 (nif, nombre, edad) VALUES ('12345678A', '', 25);
```

Esta instrucción no produce ningún error ya que en el campo nombre se inserta una cadena de caracteres vacía (que no es el valor NULL).

Una instrucción similar que sí produciría error sería la siguiente

```
INSERT INTO EJEMPLO1 (nif, nombre, edad) VALUES ('12345678A', NULL, 25);
```

Ya que el campo nombre se ha declarado como `NOT NULL`

✅ Ejercicio 12

Ejemplo 2

Crea la siguiente tabla en una base de datos nueva.

EJEMPLO2 (nif <caracter 10>, nombre <caracter 30>, fecha_modificacion)

Restricciones del esquema ER:

- Clave primaria: nif

Restricciones del diccionario de datos:

- El atributo nombre no pueda tomar valores nulos.
- El atributo fecha tome por defecto la fecha del sistema que viene dado por la variable CURRENT_TIMESTAMP.

💡 ***Solución***

```
-- Creación base de datos
CREATE DATABASE IF NOT EXISTS bd_ejemplo2;
USE bd_ejemplo2;

-- Creación tabla
CREATE TABLE EJEMPLO1 (
    nif VARCHAR(10) PRIMARY KEY,
    nombre VARCHAR(30) NOT NULL,
    fecha_modificacion DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Ahora inserta una fila en la tabla dando valores a todas las columnas, salvo a la columna fecha:

```
INSERT INTO ejemplo2 (nif, nombre) VALUES ('21777888', 'JUANA GÓMEZ');
```

Al visualizar el contenido de la tabla, en la columna fecha se almacenará la fecha del sistema, ya que no se dio valor.
Consulta los valores de la tabla e indica el valor que
ha tomado el campo fecha.

💡 ***Solución***

```
-- Consulta de datos
SELECT * FROM EJEMPLO2;

-- Valores devueltos
| nif      | nombre      | fecha_modificacion |
| -------- | ----------- | ------------------ |
| 21777888 | JUANA GÓMEZ | 2025-07-14 11:07:35 |
```

✅ Ejercicio 13

Ejemplo 3

Crea la siguiente tabla en una base de datos nueva.

EJEMPLO3 (nif <caracter 10>, nombre <caracter 30>, edad <entero 3>, curso <caracter 4>)

Restricciones del esquema ER:

- Clave primaria: nif

Restricciones del diccionario de datos:

- El atributo nombre no pueda tomar valores nulos.
- El atributo nombre debes estar en mayúsculas.
- La edad ha de estar comprendida entre 5 y 20 años.
- El atributo curso sólo puede tomar los valores '1ESO','2ESO','3ESO','4ESO','1BAC','2BAC'.

Recuerda que con CONSTRAINT se le puede dar nombre a las restricciones

💡 ***Solución***

```
-- Creación base de datos
CREATE DATABASE IF NOT EXISTS bd_ejemplo3;
USE bd_ejemplo3;

-- Creación tabla
CREATE TABLE EJEMPLO3 (
    nif VARCHAR(10) PRIMARY KEY,
    nombre VARCHAR(30) NOT NULL,
    edad INT(3),
   curso VARCHAR(30),
    CONSTRAINT edad_valida CHECK (edad>=5 AND edad<=20),
    CONSTRAINT curso_valido CHECK (curso IN ('1ESO','2ESO','3ESO','4ESO','1BAC','2BAC')),
    CONSTRAINT nombre_mayusculas CHECK (nombre = UPPER(nombre))

);
```

Prueba ahora a realizar inserciones en la tabla que hagan fallar las restricciones

💡 ***Solución***

```
-- Registro correcto
INSERT INTO EJEMPLO3 (nif, nombre, edad, curso) VALUES ('12345678', 'PACO PÉREZ', 16, '2ESO');

-- Error: Clave primaria duplicada
INSERT INTO EJEMPLO3 (nif, nombre, edad, curso) VALUES ('12345678', 'ANA MARTÍNEZ', 17, '1BAC');
ERROR 1062 (23000): Duplicate entry '12345678' for key 'PRIMARY'

-- Error: Nombre NULL
INSERT INTO EJEMPLO3 (nif, nombre, edad, curso) VALUES ('23456789', NULL, 15, '3ESO');
ERROR 1048 (23000): Column 'nombre' cannot be null

-- Error: Edad fuera de rango
INSERT INTO EJEMPLO3 (nif, nombre, edad, curso) VALUES ('34567890', 'MARTA SÁNCHEZ', 3, '1ESO');
ERROR 3819 (HY000): Check constraint 'edad_valida' is violated.

-- Error: Curso no válido
INSERT INTO EJEMPLO3 (nif, nombre, edad, curso) VALUES ('45678901', 'CARLOS PÉREZ', 17, '1DAW');
ERROR 3819 (HY000): Check constraint 'curso_valido' is violated.

-- Error: Curso no válido, edad fuera de rango y nombre en minúsculas
INSERT INTO EJEMPLO3 (nif, nombre, edad, curso) VALUES ('56789012', 'luis garcía', 3, '1DAM');
ERROR 3819 (HY000): Check constraint 'edad_valida' is violated.
```

> ⚠ **Aviso**
>
> MySQL detiene la validación en la primera restricción que falla.

✅ Ejercicio 14

Ejemplo 4

Crea la siguiente tabla en una base de datos nueva.

EJEMPLO4 (nif <caracter 10>, nombre <caracter 30>, edad <entero 3>)

Restricciones del esquema ER:

- Clave primaria: nif

Restricciones del diccionario de datos:

- El atributo nombre no pueda tomar valores repetidos.

💡 ***Solución***

```sql
-- Creación base de datos
CREATE DATABASE IF NOT EXISTS bd_ejemplo4;
USE bd_ejemplo4;

-- Creación tabla
CREATE TABLE EJEMPLO4 (
    nif VARCHAR(10) PRIMARY KEY,
    nombre VARCHAR(30) UNIQUE,
    edad INT(3)
);
```

Inserta filas en la tabla haciendo fallar la restricción de nombre repetido. ¿Qué instrucción has ejecutado? ¿Qué error aparece?

💡 ***Solución***

```sql
-- Registro correcto
INSERT INTO EJEMPLO4 (nif, nombre, edad) VALUES ('12345678A', 'María López', 17);

-- Error: Nombre duplicado
INSERT INTO EJEMPLO4 (nif, nombre, edad) VALUES ('87654321B', 'María López', 18);
ERROR 1062 (23000): Duplicate entry 'MARÍA LÓPEZ' for key 'EJEMPLO4.nombre'
```

✅ Ejercicio 15

Fabricantes y artículos

Crea las siguientes tablas con las restricciones definidas.

Tabla FABRICANTES

cod_fabricante	entero (3)
nombre	carácter (15)
pais	carácter (15)

Tabla ARTICULOS

articulo	carácter (20)
cod_fabricante	entero (3)
peso	entero (3)
categoria	carácter (10)
precio_venta	decimal (6,2)
precio_costo	decimal (6,2)
existencias	entero (5)

Restricciones para la tabla FABRICANTES:

- La clave primaria es cod_fabricante
- Las columnas nombre y pais han de almacenarse en mayúscula.

Restricciones para la tabla ARTICULOS:

- la clave primaria está formada por las columnas: articulo y cod_fabricante.
- cod_fabricante es clave ajena que referencia a la tabla FABRICANTES.
- precio_venta, precio_costo y peso han de ser >0
- categoria debe tomar los valores 'Primera', 'Segunda' o 'Tercera'.

💡 ***Solución***

```
-- Creación base de datos
CREATE DATABASE IF NOT EXISTS bd_fabricantes;
USE bd_fabricantes;

-- Creación tabla FABRICANTES
CREATE TABLE FABRICANTES (
    cod_fabricante INT(3) PRIMARY KEY,
    nombre VARCHAR(15) NOT NULL CHECK (nombre = UPPER(nombre)),
    pais VARCHAR(15) NOT NULL CHECK (pais = UPPER(pais))
);

-- Creación tabla ARTICULOS
CREATE TABLE ARTICULOS (
    articulo VARCHAR(20),
    cod_fabricante INT(3),
    peso INT(3) CHECK (peso > 0),
    categoria VARCHAR(10) CHECK (categoria IN ('Primera', 'Segunda', 'Tercera')),
    precio_venta DECIMAL(6,2) CHECK (precio_venta > 0),
    precio_costo DECIMAL(6,2) CHECK (precio_costo > 0),
    existencias INT(5),

    PRIMARY KEY (articulo, cod_fabricante),
    FOREIGN KEY (cod_fabricante) REFERENCES FABRICANTES(cod_fabricante)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

✅ Ejercicio 16

Ejemplo 5

Ahora vamos a crear la tabla EJEMPLO5 a partir de los datos recuperados en una consulta sobre otra tabla, en este caso realizaremos una consulta sobre la tabla EJEMPLO que hemos creado en el Ejercicio 8:

```
USE bd_ejercicios;
CREATE TABLE EJEMPLO5 AS SELECT * FROM personas;
```

La tabla se crea con los mismos nombres de columnas e idéntico contenido de filas que la tabla PERSONAS.

✅ Ejercicio 17

Recuerda que en la actividad Ejercicio 6 creamos la tabla PROVINCIAS y la tabla PERSONAS. Recuerda que la tabla PROVINCIAS tiene definida como clave primaria la columna cod_provincia, y la tabla PERSONAS, tiene definida una

clave
ajena cod_provin referenciando a la tabla PROVINCIAS.

Al no haber indicado ninguna política de borrado, si intentamos borrar los registros de la tabla PROVINCIAS que tienen registros referenciados en la tabla PERSONAS.

¿Qué ocurre? ¿Cómo puedes solucionar esta situación, de manera que al borrar estos registros no muestre este mensaje?

Ejercicios finales

✅ Ejercicio ??

Atletismo

Haz un script de creación de la base de datos ATLETISMO (atletismo.sql) a partir del modelo lógico-relacional.
Los tipos de datos de los distintos campos serán los que consideres convenientes, siempre que la solución adoptada sea coherente.

CLUBS(codigo, nombre, ciudad)

ATLETAS(nif, nombre, apellidos)

COMPETICIONES(codigo, descripcion, ciudad, fechaEvento)

ACTIVIDADES_DEPORTIVAS(codigo, descripcion)

FECHAS(fechaAlta)

PARTICIPAR(club, competicion)
CF: club → CLUBS
CF: competicion → COMPETICIONES

PRACTICAR(atleta, deporte, plusmarca)
CF: atleta → ATLETAS
CF: deporte → ACTIVIDADES_DEPORTIVAS

PERTENECER(atleta, fecha, club, fechaBaja)
CF: atleta → ATLETAS
CF: fecha → FECHAS
CF: club → CLUBS

COMPETIR(competicion, atleta, deporte, marca)
CF: competicion → COMPETICIONES
CF: atleta → ATLETAS
CF: deporte → ACTIVIDADES_DEPORTIVAS

atletismo.sql

✅ Ejercicio XXXX

StarTrek

Haz un script de creación de la base de datos STARTREK (startrek.sql) a partir del modelo lógico-relacional.
Los tipos de datos de los distintos campos serán los que consideres convenientes, siempre que la solución adoptada sea coherente.

ACTORES(idActor, nombre, nacionalidad, fechaNacimiento)

PERSONAJES(nombre, raza, graduacionMilitar, actor, capMilitar)
CF: actor → ACTORES
CF: capMilitar → PERSONAJES

PELÍCULAS(titulo, director, añoLanzamiento, protagonista)
CF: protagonista → PERSONAJES

CAPÍTULOS(titulo, temporada, ordenRodaje, fechaEmision1)

PLANETAS(codigo, nombre, galaxia)

NAVES(codigo, nombre, tripulacion)

APARECER(personaje, pelicula)
CF: personaje → PERSONAJES
CF: pelicula → PELÍCULAS

PARTICIPAR(capitulo, personaje)
CF: capitulo → CAPÍTULOS
CF: personaje → PERSONAJES

VISITAR(capitulo, planeta, nave, problema)
CF: capitulo → CAPÍTULOS
CF: planeta → PLANETAS
CF: nave → NAVES

startrek.sql