**BOLETIN B. MODELO FÍSICO**

He creado la BD “BDBoletinB” con el comando:

“CREATE DATABSE BDBoletinB CHARACTER SET utf8 COLLATE utf8\_general\_ci;”

1. **Escalada**

**Modelo lógico:**

**Asociación** (código, nombre, ubicación)

CP: código

**Alpinistas** (nif, nombre, fecha\_nacimiento, código, fecha\_ingreso)

CP: nif

CAj: código → Asociación (código)

**Teléfonos** (nif, teléfono)

CP: {nif, teléfono}

CAj: nif → Alpinistas (nif)

**Picos** (nombre, altura, coordenadas, país)

CP: nombre

**Escalada** (nif, nombre, fecha\_inicio, tiempo, oxígeno, cara)

CP: {nif, nombre}

CAj: nif → Alpinistas (nif)

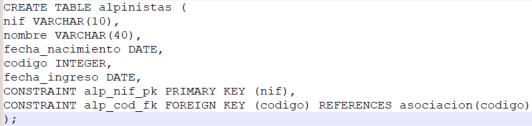
CAj: nombre → Picos (nombre)

* 1. **Traduce el modelo lógico relacional a modelo físico en MySQL**
* Crear la tabla de Asociación:

A close-up of a computer code

Description automatically generated

* Crear la tabla de Alpinistas:



* Crear la tabla de teléfonos:

A close-up of a computer code

Description automatically generated

* Crear la tabla Picos:

A close-up of a white background

Description automatically generated

* Crear la tabla de Escalada:

A close-up of a computer code

Description automatically generated

* 1. **Script**

Escribir las sentencias en Notepad y guardarlo como un archivo SQL. En mi caso, he creado Prueba1.sql y lo he guardado en C:. Para añadirlo a la base de datos, ejecutar:

-> SOURCE C:\prueba1.sql;

* 1. **Añade un campo Observaciones sobre la tabla Picos (decide tú el tipo y tamaño del mismo).**



* 1. **Modifica el campo Codigo de la tabla Asociación para que pueda contener números hasta el 999.999.**

Como este campo es Foreign Key de la tabla Alpinistas, primero hay que borrar esta restricción con:



Luego ya se puede modificar Codigo en Asociación con:



Finalmente, hay que cambiar código en Alpinistas, para que sea del mismo tipo que en asociación, y crear la restricción de Foreign key nuevamente con:

 \*ejecutar por separado.

* 1. **Añade un campo Foto sobre la tabla Alpinistas (decide tú el tipo y tamaño del mismo).**

SQL usa tipos de datos BLOB (objeto binario grande) para almacenar grandes cantidades de datos, y con este tipo de dato se pueden insertar imágenes en MySQL.



También valdría “ALTER TABLE alpinistas ADD ruta\_foto VARCHAR(1024);” 🡪 para guardar la ruta de la foto.

* 1. **Modifica la columna fecha\_ingreso de la tabla Alpinistas para que por defecto tenga la fecha del sistema.**

ALTER TABLE alpinistas MODIFY fecha\_ingreso **TIMESTAMP**;

* 1. **El valor del campo coordenadas no debe repetirse nunca, a pesar de no ser clave primaria. Introduce la correspondiente restricción sobre Picos.**

 Añadir la restricción UNIQUE

* 1. **Si se borra un alpinista de la base de datos, ¿qué debería hacerse con sus correspondientes entradas en la tabla Teléfonos? Indica la sentencia DDL necesaria para aplicar esta restricción.**

No necesitaríamos los teléfonos, así que se deben borrar también. Primero se deberían poner a nulos los valores del nif. Sentencia:

ALTER TABLE telefonos DROP CONSTRAINT tel\_nif\_fk;

ALTER TABLE telefonos ADD CONSTRAINT tel\_nif\_fk FOREIGN KEY (nif) REFERENCES alpinistas(nif) ON DELETE CASCADE;

* 1. **¿Y si se borra una asociación de la base de datos?**

Poner como nulo el valor de código en la tabla de alpinistas. Sentencia:

ALTER TABLE alpinistas DROP CONSTRAINT alp\_cod\_fk;

ALTER TABLE alpinistas ADD CONSTRAINT alp\_cod\_fk FOREIGN KEY (codigo) REFERENCES asociacion(codigo) ON DELETE SET NULL;

1. **Sucursales**

Añadir las claves ajenas pertinentes, alterando las tablas, al siguiente modelo físico:

CREATE TABLE articulos (

id\_art INTEGER,

nombre VARCHAR(30) DEFAULT '',

precio NUMBER DEFAULT 0.0,

codigo VARCHAR(7) DEFAULT '',

id\_gru INTEGER DEFAULT 0 NOT NULL,

CONSTRAINT art\_id\_pk PRIMARY KEY(id\_art)

);

CREATE TABLE clientes (

id\_cli INTEGER NOT NULL ,

nombre VARCHAR(60) DEFAULT '' NOT NULL ,

direccion VARCHAR(80) DEFAULT '',

CONSTRAINT cli\_idc\_pk PRIMARY KEY (id\_cli)

);

CREATE TABLE facturas (

id\_fac INTEGER NOT NULL ,

serie VARCHAR(1) DEFAULT '' NOT NULL,

numero INTEGER DEFAULT 0 NOT NULL,

fecha DATE,

id\_cli INTEGER DEFAULT 0 NOT NULL,

id\_ven INTEGER NOT NULL,

CONSTRAINT fac\_idf\_pk PRIMARY KEY (id\_FAC)

);

CREATE TABLE grupos (

id\_gru INTEGER,

descripcion VARCHAR(15) DEFAULT '' NOT NULL,

CONSTRAINT gru\_idg\_pk PRIMARY KEY (id\_gru)

);

CREATE TABLE lineas\_factura (

id\_lin INTEGER NOT NULL,

id\_fac INTEGER DEFAULT 0 NOT NULL ,

importe FLOAT DEFAULT 0 NOT NULL ,

id\_art INTEGER NOT NULL,

cantidad INTEGER NOT NULL,

fecha DATE DEFAULT '01/01/2000' NOT NULL ,

id\_suc INTEGER DEFAULT 0 NOT NULL ,

CONSTRAINT lin\_pk PRIMARY KEY (id\_lin, factura)

);

CREATE TABLE sucursales (

id\_suc INTEGER NOT NULL,

descripcion VARCHAR(15) DEFAULT '' NOT NULL ,

CONSTRAINT suc\_ids\_pk PRIMARY KEY (id\_suc)

);

CREATE TABLE vendedores (

id\_ven INTEGER NOT NULL,

nombre VARCHAR(50) DEFAULT '' NOT NULL ,

fecha\_ingreso DATE DEFAULT '01-01-2000' NOT NULL ,

salario FLOAT DEFAULT 0 NOT NULL ,

CONSTRAINT ven\_idv\_pk PRIMARY KEY (id\_ven)

);

Claves ajenas:

* Artículos:
  + Id\_gru 🡪 CAj. Viene de la tabla Grupos
* Facturas
  + Id\_cli 🡪 Caj. Viene de la tabla Clientes
  + Id\_ven 🡪 CAj. Viene de la tabla Vendedores.
* Lineas\_factura
  + Id\_fac 🡪 CAj. Viene de la tabla Facturas.
  + Id\_art 🡪 CAj. Viene de la tabla Artículos.
  + Id\_suc 🡪 CAj. Viene de la tabla Sucursales.

Las sentencias para añadir estas claves ajenas serían:

* Tabla Artículos:



* Tabla Facturas:



* Tabla Lineas\_factura:



1. **Ventas**

Modelo lógico:

**Usuario** (dni, nombre, fecha\_nac)

CP: dni

**Cliente** (dni, descuento )

CP: dni

CAj: dni → Usuario (dni)

\* Restricción CAj dni: un usuario no puede borrarse si es cliente.

**Pedido** (código, fecha, dni\_cliente)

CP: código

CAj: dni\_cliente → Cliente (dni)

VNN: dni\_cliente

\* Restricción CAj dni\_cliente: un cliente no puede borrarse si tiene líneas de pedido.

**Línea** (número\_línea, precioventa, cod\_pedido, cod\_producto, cantidad)

CP: {número\_línea, código\_pedido}

CAj: cod\_pedido → Pedido (código)

CAj: cod\_producto → Producto (cod\_producto)

VNN: cod\_producto

\* Restricción CAj cod\_pedido: si se borra o actualiza un pedido se borra todas las líneas

de ese pedido en cascada.

\* Restricción CAj cod\_producto: un producto no podrá borrarse si existe en líneas del

pedido.

**Producto** (cod\_producto, nombre, precioactual)

CP: cod\_producto

* Crear la tabla Usuario:

A close-up of a computer code

Description automatically generated

* Crear la tabla Cliente:

A close-up of a computer code

Description automatically generated ON DELETE ON ACTION también se puede usar para todas las restricciones de no borrar

* Crear la tabla Pedido:

A close-up of a computer screen

Description automatically generated

* Crear la tabla Linea:

A close-up of a computer screen

Description automatically generated

* Crear la tabla Producto:

A close-up of a computer code

Description automatically generated