UD.4: MODELO FÍSICO DDL

Prácticas no evaluables
Boletín B (Solucionado)

Bases de Datos (BD)
CFGS DAM/DAW

Abelardo Martínez y Pau Miñana.
Basado y modificado de Sergio Badal y Raquel Torres.
Curso 2023-2024

Aspectos a tener en cuenta

Estas actividades son opcionales y no evaluables pero es recomendable hacerlas para un mejor aprendizaje de la asignatura.

> 🚫 **Si buscas las soluciones por Internet o preguntas al oráculo de ChatGPT, te estarás engañando a ti mismo**. Ten en cuenta que ChatGPT no es infalible ni todopoderoso. Es una gran herramienta para agilizar el trabajo una vez se domina una materia, pero usarlo como atajo en el momento de adquirir habilidades y conocimientos básicos perjudica gravemente tu aprendizaje.
>
> Si lo utilizas para obtener soluciones o asesoramiento respecto a las tuyas, revisa cuidadosamente las soluciones propuestas igualmente. Intenta resolver las actividades utilizando los recursos que hemos visto y la documentación extendida que encontrarás en el "Aula Virtual".

ÍNDICE

1. Soluciones

Para empezar creamos la base de datos y nos conectamos a ella:

```
CREATE DATABASE IF NOT EXISTS bdu4_b
CHARACTER SET utf8mb4 COLLATE utf8mb4_es_0900_as_cs;
USE bdu4_b
```

1.1. Enunciado 1. Escalada

```
Asociación (código, nombre, ubicación)
CP:  código

Alpinistas (nif, nombre, f_nacimiento, cod_aso, f_ingreso)
CP:  nif
CAj: cod_aso → Asociación {código}

Teléfonos (nif, teléfono)
CP:  {nif, teléfono}
CAj: nif → Alpinistas {nif}

Picos (nombre, altura, coordenadas, país)
CP:  nombre

Escalada (nif, nombre, f_inicio, tiempo, oxígeno, cara)
CP:  nif
CAj: nif → Alpinistas {nif}
CAj: nombre → Picos {nombre}
```

1.1.1. Crea el modelo físico

Traduce el Modelo Lógico Relacional anterior a Modelo Físico en MySQL.

Solución

```
CREATE TABLE asociacion (
codigo        VARCHAR(4),
nombre        VARCHAR(30),
ubicacion     VARCHAR(30),
CONSTRAINT aso_cod_pk PRIMARY KEY (codigo)
);

CREATE TABLE alpinistas (
nif           CHAR(9),
nombre        VARCHAR(30),
f_nacimiento  DATE,
cod_aso       VARCHAR(4),
f_ingreso     DATE,
CONSTRAINT alp_nif_pk PRIMARY KEY (nif),
CONSTRAINT alp_codaso_fk FOREIGN KEY (cod_aso) REFERENCES
asociacion(codigo)
);

CREATE TABLE telefonos (
nif           CHAR(9),
telefono      CHAR(9),
CONSTRAINT tel_ntf_pk PRIMARY KEY (nif, telefono),
CONSTRAINT tel_nif_fk FOREIGN KEY (nif) REFERENCES alpinistas(nif)
);

CREATE TABLE picos (
nombre        VARCHAR(30),
altura        INTEGER,
coordenadas   VARCHAR(10),
pais          VARCHAR(20),
CONSTRAINT pic_nom_pk PRIMARY KEY (nombre)
);

CREATE TABLE escalada (
nif           CHAR(9),
nombre        VARCHAR(30),
f_inicio      DATE,
tiempo        VARCHAR(8),
oxigeno       DECIMAL(3,2),
cara          VARCHAR(20),
CONSTRAINT esc_nim_pk PRIMARY KEY (nif, nombre),
CONSTRAINT esc_nif_fk FOREIGN KEY (nif) REFERENCES
alpinistas(nif),
CONSTRAINT esc_nom_fk FOREIGN KEY (nombre) REFERENCES
picos(nombre)
);
```

Unos comentarios sobre el tipo de dato escogido para el tiempo de la escalada:

- Se considera que una escalada puede durar varios días.
- Para mayor compatibilidad entre SGBD usaremos un texto for ejemplo con formato dd:hh:mm y en todo caso lo transformamos en los tipos y formatos adecuados en cada uno.
- Algunos SGBD tienen tipos de datos específicos para intervalos de tiempo, pero no es exactamente el caso de MySQL, que usa TIME para este cometido.
- En la mayoría de SGBD, TIME tiene valor máximo 24:00:00 y su cometido es guardar un tiempo específico de un día. En MySQL se usa también como intervalo de tiempo, y su rango es ±838:59:59.000000 (unos ±34 días). Sería una opción en MySQL, pero no se considera adecuada.

1.1.2. Script

Escribe todas las sentencias de creación de las tablas en un archivo de texto plano con un editor de texto. Llama al script "alpinista.sql" y ejecútalo en MySQL.

Solución

Uniendo la primera parte de creación de la base de datos (*asegúrate de añadir un ; al final del comando USE para el script*) con las tablas creadas del apartado anterior en el archivo "alpinista.sql", solo faltaría ejecutarlo: *SOURCE ~/aplinista.sql* si está en el home de un sistema Linux, por ejemplo.

1.1.3. Alteración de las tablas

Realiza las siguientes modificaciones sobre la base de datos una vez creadas las tablas:

Solución

- Añade un campo observaciones sobre la tabla Picos.

```
ALTER TABLE picos ADD observaciones VARCHAR(100);
```

- Modifica el campo código de la tabla Asociación para que pueda contener números hasta el 999.999.

Se debe cambiar también el tipo de la clave ajena asociada a este campo. Puesto que se ha usado un tipo textual, si se mantiene se puede aumentar, aunque no reducir (incluso sin datos) por culpa de esa clave

```
ALTER TABLE asociacion MODIFY codigo VARCHAR(6);
ALTER TABLE alpinistas MODIFY cod_aso VARCHAR(6);
```

En caso de querer cambiar a tipos numéricos el SGBD hace la transformación siempre que el campo textual solo contenga números, pero se debe eliminar primero la clave ajena, ya que un tipo numérico no puede referenciar a uno textual y viceversa. Se vuelve a crear una vez los tipos vuelven a ser compatibles.

```
ALTER TABLE alpinistas DROP CONSTRAINT alp_codaso_fk;
ALTER TABLE asociacion MODIFY codigo DECIMAL(6);
-- Si se omite s, es 0, ambos DECIMAL son equivalentes.
ALTER TABLE alpinistas MODIFY cod_aso DECIMAL(6,0);
ALTER TABLE alpinistas ADD CONSTRAINT alp_codaso_fk FOREIGN KEY
(cod_aso) REFERENCES asociacion(codigo);
-- Valen INTEGER o MEDIUMINT, pero necesitaríamos otra restricción
-- CONSTRAINT x_ck CHECK (x=<999999) para cada campo.
```

Ten en cuenta que en MySQL *no serviría* para este caso desactivar las restricciones con *SET FOREIGN_KEY_CHECKS = 0;*. Aunque deja de comprobarlas al introducir datos las restricciones existen, con lo que sigue saltando el mismo error por los tipos de datos incompatibles en la clave ajena al modificar los campos.

- Añade un campo foto sobre la tabla Alpinistas.

Aunque es posible, no es recomendable añadir elementos multimedia a una BD relacional:

```
ALTER TABLE alpinistas ADD foto BLOB; -- Mejor NO usar esta.
```

La práctica más común es guardar la ruta completa de acceso al archivo si es local o la url si está disponible en internet en un campo de texto:

```
ALTER TABLE alpinistas ADD foto VARCHAR(1024);
```

- Modifica la columna f_ingreso de la tabla Alpinistas para que por defecto tenga la fecha del sistema.

Puesto que cada SGBD usa sus propias funciones para determinar la fecha del sistema, esta respuesta es exclusiva de MySQL.

En MySQL existen diversas funciones para devolver la fecha del sistema que además tienen varios sinónimos:

CURRENT_DATE/*CURRENT_DATE()*/*CURDATE()* Devuelve una fecha formato DATE.
CURRENT_TIME/*CURRENT_TIME()*/*CURTIME()*. Devuelve una hora formato TIME.
CURRENT_TIMESTAMP/*CURRENT_TIMESTAMP()*/*NOW()*. Devuelve el TIMESTAMP de cuando se lanzó la sentencia.
SYSDATE() Devuelve el TIMESTAMP de cuando realmente se ejecuta la función.

Para usarlas en el DEFAULT se debe poner la función dentro de un paréntesis, aunque en algunos casos concretos es opcional, mejor usarlo siempre para asegurar.

Puesto que para este caso solo se pide el día lo ideal es usar alguna de las variantes de CURDATE().

```
ALTER TABLE alpinistas MODIFY f_ingreso DATE DEFAULT (CURDATE());
```

- El valor del campo coordenadas no debe repetirse nunca.

Se puede modificar la columna coordenadas o añadir una restricción. Recuerda que no deben implementarse ambas.

```
-- Por columna
ALTER TABLE picos MODIFY coordenadas VARCHAR(10) UNIQUE;
--Por restricción
ALTER TABLE picos ADD CONSTRAINT pic_coo_uk UNIQUE(coordenadas);
```

- Los Alpinistas deben tener al menos 18 años cuando se registran.

Para poder hacer este cálculo existen varios métodos, algunas opciones:

Directamente restar las fechas nos da una fecha con la diferencia entre ambas. Se puede usar la función *YEAR* para ver el año de esa respuesta y debe ser >=2018 (los años en 2 cifras de la respuesta, asumiendo que no hay alpinistas registrados hace 100 años, se interpretan como año 20XX).

```
ALTER TABLE alpinistas ADD CONSTRAINT alp_ingnac_ck
CHECK(YEAR(f_ingreso-f_nacimiento)>=2018);
```

DATEDIFF(f1,f2) nos da la diferencia entre 2 fechas en días, pero pasarlo a años es más complejo que simplemente usar la siguiente opción.

TIMESTAMPDIFF(unidad,f1,f2) permite especificar la unidad en la que nos devuelve la diferencia entre TIMESTAMPS, pero es compatible con otros tipos de datos, así que se puede usar directamente con nuestros DATE.

```
ALTER TABLE alpinistas ADD CONSTRAINT alp_ingnac_ck
CHECK(TIMESTAMPDIFF(YEAR,f_nacimiento,f_ingreso)>=18);
```

- Modifica la condición para que no se puedan registrar hasta los 21 años.

En MySQL se necesita eliminar primero la restricción y crearla de nuevo.

```
ALTER TABLE alpinistas DROP CONSTRAINT alp_ingnac_ck;
ALTER TABLE alpinistas ADD CONSTRAINT alp_ingnac_ck
CHECK(TIMESTAMPDIFF(YEAR,f_nacimiento,f_ingreso)>=21);
```

- 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.

Puesto que no tiene sentido almacenar teléfonos de alpinistas dados de baja se deberían borrar en cascada. Esto se consigue modificando la clave ajena del nif en la tabla telefonos.

```
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;
```

- ¿Y si se borra una asociación de la base de datos?

En este caso parece factible que queramos conservar a los alpinistas para mantener sus datos y marcas o porque la asociación simplemente ha dejado de existir y probablemente se unan a otra y no sea buena idea eliminarlos. En este caso podemos establecer la asociación del alpinista a NULL o crear una asociación genérica (no real) donde almacenarlos, esta segunda opción sería necesaria si el diseño de la BD no permite alpinistas sin asociación (si cod_aso fuese NOT NULL).

```sql
ALTER TABLE alpinistas DROP CONSTRAINT alp_codaso_fk;
ALTER TABLE alpinistas ADD CONSTRAINT alp_codaso_fk FOREIGN KEY
(cod_aso) REFERENCES asociacion(codigo) ON DELETE SET NULL;

-- Suponiendo que el código de la asociación genérica sea 000000
ALTER TABLE alpinistas DROP CONSTRAINT alp_codaso_fk;
ALTER TABLE alpinistas MODIFY cod_aso DECIMAL(6) DEFAULT 000000;
ALTER TABLE alpinistas ADD CONSTRAINT alp_codaso_fk FOREIGN KEY
(cod_aso) REFERENCES asociacion(codigo) ON DELETE SET DEFAULT;
```

1.2. Enunciado 2. Ventas

Traduce el siguiente Modelo Lógico Relacional a Modelo Físico en MySQL.

```
Usuario (dni, nombre, f_nacimiento)
CP:  dni

Cliente (dni, descuento)
CP:  dni
CAj: dni → Usuario {dni}
Restricción CAj dni: un usuario no puede borrarse si es cliente.
Restricción Adicional:
descuento es un número entre 0'00 y 1'00.

Pedido (código, fecha, dni_cliente*)
CP:  código
CAj: dni_cliente → Cliente {dni}
VNN: dni_cliente
Restricción CAj dni: un usuario no puede borrarse si tiene pedidos.

Línea (cod_ped, num_línea, cod_prod*, cantidad, precio_venta)
CP:  {cod_ped, num_línea}
ÚNICO: {cod_ped, cod_prod}
CAj: cod_ped → Pedido {código} B:C M:C
CAj: cod_prod → Producto {cod_producto} B:R
VNN: cod_prod

Producto (cod_producto, nombre, precio_actual)
CP:  cod_producto
```

Solución

Las restricciones de integridad referidas a evitar borrados o actualizaciones si están afectados por una clave ajena (NO ACTION/RESTRICT) no se necesitan implementar, puesto que es el funcionamiento por defecto de los SGBD.

```sql
-- Tabla Usuario
CREATE TABLE usuario (
  dni             CHAR(9),
  nombre          VARCHAR(45),
  fecha_nac       DATE,
  CONSTRAINT usu_dni_pk PRIMARY KEY (dni)
);

-- Tabla Cliente
CREATE TABLE cliente (
  dni             CHAR(9),
  descuento       DECIMAL(3,2),
  CONSTRAINT cli_dni_pk PRIMARY KEY (dni),
  CONSTRAINT cli_dni_fk FOREIGN KEY (dni) REFERENCES usuario
(dni),
  CONSTRAINT cli_des_ck CHECK(descuento>=0.00 AND descuento<=1.00)
);

-- Tabla Producto
CREATE TABLE producto (
  codigo_producto VARCHAR(10),
  nombre          VARCHAR(45),
  precio_actual   DECIMAL(5,2),
  CONSTRAINT pro_cod_pk PRIMARY KEY (codigo_producto)
);

-- Tabla Pedido
CREATE TABLE pedido (
  codigo          VARCHAR(10),
  fecha           DATE,
  dni_cliente     CHAR(9) NOT NULL,
  CONSTRAINT ped_cod_pk PRIMARY KEY (codigo),
  CONSTRAINT ped_dni_fk FOREIGN KEY (dni_cliente) REFERENCES
cliente(dni)
);

-- Tabla Línea
CREATE TABLE linea (
  cod_ped   VARCHAR(10),
  numero_linea    INTEGER,
  cod_prod VARCHAR(10) NOT NULL,
  cantidad        INTEGER,
  precio_venta    DECIMAL(6,2),
  CONSTRAINT lin_lincod_pk PRIMARY KEY (numero_linea, cod_ped),
  CONSTRAINT lin_ped_fk FOREIGN KEY (cod_ped) REFERENCES pedido
(codigo) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT lin_pro_fk FOREIGN KEY (cod_prod) REFERENCES producto
(codigo_producto),
  CONSTRAINT lin_pedpro_uk UNIQUE (cod_ped, cod_prod)
);
```

1.3. Enunciado 3. Sucursales

Adapta las instrucciones para que el script pueda ser ejecutado sobre una base de datos de MySQL.

Por otro lado, fíjate que no se ha creado ninguna clave ajena. Debes analizar cuáles son las claves ajenas necesarias para el correcto funcionamiento del sistema y añadirlas a posteriori a la base de datos por medio de modificaciones de tablas.

Solución

Simplemente se cambia el NUMBER a DECIMAL y se usa el formato de MySQL para los DEFAULT en los DATE. Puesto que las claves ajenas se añaden después no es necesario reordenar las tablas. Las otras tablas se quedan como estaban. En caso de incorporarlas a la propia creación de cada tabla se deben mover delante las tablas referenciadas antes de las que las tengan en su clave ajena.

Las tablas cambiadas son:

```
CREATE TABLE articulos (
id_art      INTEGER,
nombre      VARCHAR(30) DEFAULT '',
precio      DECIMAL(2,1) DEFAULT 0.0,
codigo      VARCHAR(7) DEFAULT '',
id_gru      INTEGER DEFAULT 0 NOT NULL,
CONSTRAINT art_id_pk PRIMARY KEY(id_art) );

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 '2000-01-01' NOT NULL,
id_suc      INTEGER DEFAULT 0 NOT NULL,
CONSTRAINT lin_linfac_pk PRIMARY KEY (id_lin, id_fac) );

CREATE TABLE vendedores (
id_ven      INTEGER NOT NULL,
nombre      VARCHAR(50) DEFAULT '' NOT NULL,
f_ingreso   DATE DEFAULT '2000-01-01' NOT NULL,
salario     FLOAT DEFAULT 0 NOT NULL,
CONSTRAINT ven_idv_pk PRIMARY KEY (id_ven) );
```

- Claves Ajenas

Se usa un único ALTER para cada tabla, añadiendo todas sus restricciones de golpe, aunque se puede separar cada restricción en su propio ALTER.

Para las claves ajenas se considera la actualización en cascada y el borrado se pone al valor por defecto cuando éste existe y se considera oportuno. En los otros casos, si se considera oportuno un borrado en cascada se añade y si no se deja restringido.

- Borrar las facturas en que haya participado un vendedor si éste se elimina no parece una buena idea, B:R.
- Si se borra una factura, se considera que se ha anulado una compra y por tanto B:C para sus líneas aunque exista un valor de factura por defecto.
- Aunque se borre un artículo no tiene sentido borrar las líneas de sus ventas, al no poder ser nulo ni tener valor por defecto, se deja restringido B:R.

```
-- Tabla articulos
ALTER TABLE articulos ADD CONSTRAINT art_idg_fk
FOREIGN KEY (id_gru) REFERENCES grupos (id_gru)
ON DELETE SET DEFAULT ON UPDATE CASCADE;

-- Tabla facturas
ALTER TABLE facturas
ADD CONSTRAINT fac_idv_fk
FOREIGN KEY (id_ven) REFERENCES vendedores (id_ven)
ON UPDATE CASCADE,
ADD CONSTRAINT fac_idc_fk
FOREIGN KEY (id_cli) REFERENCES clientes (id_cli)
ON DELETE SET DEFAULT ON UPDATE CASCADE;

-- Tabla lineas_factura
ALTER TABLE lineas_factura
ADD CONSTRAINT lin_ids_fk
FOREIGN KEY (id_suc) REFERENCES sucursales (id_suc)
ON DELETE SET DEFAULT ON UPDATE CASCADE,
ADD CONSTRAINT lin_idf_fk
FOREIGN KEY (id_fac) REFERENCES facturas (id_fac)
ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT lin_ida_fk
FOREIGN KEY (id_art) REFERENCES articulos (id_art)
ON UPDATE CASCADE;
```

2. Bibliografía

- MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/refman/8.0/en/
- Oracle Database Documentation https://docs.oracle.com/en/database/oracle/oracle-database/index.html
- JavaTPoint. Difference between MySQL and Oracle. https://www.javatpoint.com/mysql-vs-oracle
- W3Schools. MySQL Tutorial. https://www.w3schools.com/mysql/
- GURU99. Tutorial de MySQL para principiantes Aprende en 7 días. https://guru99.es/sql/
- Adam McGurk.How to change a foreign key constraint in MySQL https://dev.to/mcgurkadam/how-to-change-a-foreign-key-constraint-in-mysql-1cma
- Sqlines. MySQL - SET FOREIGN_KEY_CHECKS. http://www.sqlines.com/mysql/set_foreign_key_checks