# Diseño e implementación de la base de datos de una universidad

Icono

El contenido generado por IA puede ser incorrecto.

**Roberto Buendia Espada**

**Fundamentos de bases de datos**

**ASIR**

Contenido

[Diseño e implementación de la base de datos de una universidad 1](#_Toc205806168)

[Descripción general del sistema 3](#_Toc205806169)

[Entidades y atributos 3](#_Toc205806170)

[Relaciones entre entidades 6](#_Toc205806171)

[Extensiones del modelo (MERE) 9](#_Toc205806172)

[Decisiones de diseño tomadas 10](#_Toc205806173)

[Listado de tablas y descripción 11](#_Toc205806174)

[Diagrama MERE final 14](#_Toc205806175)

[Código SQL completo 15](#_Toc205806176)

## Descripción general del sistema

En este proyecto he diseñado y creado una base de datos para gestionar todo lo relacionado con una universidad.  
La idea es que sirva para guardar y organizar información de las personas que forman parte de ella (profesores, alumnos, administrativos, auditores y visitas) y de todo lo que pasa a nivel académico: asignaturas, aulas, exámenes, matrículas, auditorías y premios.

Para hacerlo, primero he creado un **diagrama MERE** (Modelo Entidad-Relación Extendido) donde he definido las entidades, sus atributos y cómo se relacionan. Usando cosas como:

* Jerarquías **ISA** para diferenciar tipos de personal.
* Relaciones reflexivas, por ejemplo, para alumnos que son representantes de otros.
* Relaciones de agregación, como las auditorías docentes y los premios.
* Relaciones ternarias, como la matrícula, que relaciona alumno, asignatura y convocatoria.

Después, ese modelo lo he pasado a un **modelo relacional** en MySQL, creando las tablas con sus columnas, claves y relaciones. También he añadido un trigger para controlar que el precio de una matrícula sea siempre mayor que 0 y he dejado documentados otros triggers que se necesitarían, pero que no puedo implementar aquí por las limitaciones del enunciado.

## Entidades y atributos

En el diagrama MERE he identificado las siguientes entidades y para cada una he anotado qué información guarda:

**1. Persona**

Es la base para todos los tipos de personal de la universidad.  
Guarda datos comunes como:

* DNI (identificador único)
* Nombre
* Apellidos
* Edad
* Teléfono (al menos uno)
* Sexo (opcional: Masculino, Femenino o Prefiero no indicar)

**2. Alumno**

Es un tipo de persona que estudia en la universidad.  
Además de lo anterior, guardo:

* Especialidad (ingenierías, filologías, biologías, medicinas o humanidades)

**3. Profesor**

Es un tipo de persona que imparte asignaturas.  
Además de los datos comunes, guardo:

* Salario anual (más de 0 y menos de 60.000 €)
* Departamento (o varios)

**4. Administrativo**

Tipo de persona que realiza tareas de gestión en la universidad.  
Guarda:

* Especialización (por ejemplo: recursos humanos, secretariado, legal, pedagógico…)

**5. Auditor**

Tipo de persona que evalúa a profesores y asignaturas.  
Guarda:

* Años de experiencia
* Si trabaja fuera de la universidad (sí/no)

**6. Visita**

Tipo de persona que entra en la universidad, pero no forma parte del personal o alumnado.  
Solo tiene los datos básicos de persona.

**7. Departamento**

Agrupa a los profesores y asignaturas por área de conocimiento.  
Guarda:

* Nombre (único)
* Años de existencia
* Primer profesor adscrito

**8. Curso de orientación**

Curso especial para que un auditor pueda auditar asignaturas de un departamento.  
Guarda:

* Nombre del curso
* Departamento al que pertenece
* Número de horas

**9. Aula**

Lugar físico donde se imparten las clases.  
Guarda:

* Número de aula
* Planta
* Edificio
* Capacidad
* Si tiene tablón de anuncios (sí/no)

**10. Asignatura**

Materia que imparte un profesor en un aula.  
Guarda:

* Nombre
* Profesor que la imparte
* Aula donde se da
* Posible asignatura prerrequisito (opcional)

**11. Convocatoria**

Convocatorias de examen que pueden ser compartidas por varias asignaturas.  
Guarda:

* Nombre (por ejemplo: diciembre, febrero…)

**12. Matrícula**

Relación ternaria entre alumno, asignatura y convocatoria.  
Guarda:

* Precio (mayor que 0)
* Nota final (opcional)

**13. Representa**

Relación reflexiva en la que un alumno puede ser representante (vocal) de otros.  
No tiene atributos extra.

**14. Auditoría docente**

Agregación que guarda una evaluación de un profesor en una asignatura hecha por un auditor, junto con el curso de orientación que habilita esa auditoría.

**15. Premio**

Premios otorgados a una impartición (profesor + asignatura).  
Guarda:

* Nombre
* Descripción

## Relaciones entre entidades

En este sistema, las entidades se conectan de distintas maneras para representar la realidad de la universidad. Estas son las principales relaciones y lo que significan:

**1. ISA: Tipos de persona**

PERSONA se divide en ALUMNO, PROFESOR, ADMINISTRATIVO, AUDITOR y VISITA.  
Cada persona pertenece a un solo tipo.

**2. Teléfono**

Un PERSONA puede tener uno o varios teléfonos.  
La relación es 1:N.

**3. Imparte**

Un PROFESOR imparte una o más ASIGNATURAS y  
cada asignatura es impartida por un único profesor, 1:N.

**4. Adscripción a departamento**

Un PROFESOR puede pertenecer a varios DEPARTAMENTOS y un DEPARTAMENTO puede tener varios profesores.  
Relación N:N.

**5. Aula asignada**

Cada ASIGNATURA se imparte en un AULA, pero un AULA puede tener varias asignaturas a lo largo del curso.  
Relación 1:N.

**6. Prerrequisito**

Una ASIGNATURA puede tener como prerrequisito a otra asignatura.  
Es una relación reflexiva y opcional.

**7. Convocatorias**

Una CONVOCATORIA puede estar asociada a varias ASIGNATURAS, y cada asignatura puede estar en varias convocatorias.  
Relación N:N.

**8. Matrícula**

Es una relación ternaria que une:

* Un ALUMNO
* Una ASIGNATURA
* Una CONVOCATORIA  
  y además guarda el precio y la nota.

**9. Representa**

Relación reflexiva entre ALUMNO y ALUMNO para indicar que uno es representante (vocal) de otros.  
Relación 1:N

**10. Auditoría docente**

Es una agregación que combina:

* Un AUDITOR
* Un PROFESOR
* Una ASIGNATURA
* Un CURSO\_ORIENTACION  
  y representa una evaluación del trabajo docente.

**11. Premios**

Un PREMIO puede darse a varias imparticiones (profesor + asignatura) y una impartición puede recibir varios premios.  
Relación N:N con tabla intermedia.

## Extensiones del modelo (MERE)

**1. ISA (Jerarquía de generalización/especialización)**

La entidad PERSONA se divide en 5 tipos diferentes:  
ALUMNO, PROFESOR, ADMINISTRATIVO, AUDITOR y VISITA.  
Esto se hace para no repetir los mismos atributos comunes (nombre, apellidos, DNI, etc.) en cada tabla y para separar claramente la información que es específica de cada tipo.

**2. Relaciones reflexivas**

* ASIGNATURA tiene una relación consigo misma para representar prerrequisitos.
* ALUMNO tiene una relación consigo mismo en REPRESENTA para indicar quién es representante de quién.

**3. Relaciones ternarias**

La tabla MATRICULA une a ALUMNO, ASIGNATURA y CONVOCATORIA.  
No se podría representar igual de bien con relaciones 1 a 1 o 1 a N, porque aquí se necesita tener las tres entidades juntas para saber:

* qué alumno
* en qué asignatura
* en qué convocatoria

y además guardar datos extra como precio y nota.

**4. Agregaciones**

He usado agregaciones para casos donde una relación en sí misma tiene más relaciones:

* AUDITORIA\_DOCENTE agrupa a un AUDITOR, un PROFESOR, una ASIGNATURA y el CURSO\_ORIENTACION necesario para que se pueda hacer esa auditoría.
* IMPARTICION\_PREMIO sirve para asociar un premio a una impartición concreta (profesor + asignatura).

## Decisiones de diseño tomadas

Durante el diseño de la base de datos he tenido que tomar varias decisiones para que el modelo sea más claro, funcional y fácil de mantener. Estas son las más importantes:

**1. Uso de una entidad general PERSONA**

En lugar de repetir los datos básicos (DNI, nombre, apellidos, edad, teléfono, sexo) en cada tipo de persona, he creado la entidad PERSONA y de ella se derivan ALUMNO, PROFESOR, ADMINISTRATIVO, AUDITOR y VISITA.  
Esto evita duplicar información y hace el modelo más ordenado.

**2. Claves artificiales en algunas tablas**

En entidades como AULA, ASIGNATURA, DEPARTAMENTO, CURSO\_ORIENTACION, PREMIO he usado un id numérico auto incremental como clave primaria, en vez de usar campos como el nombre o número de aula.  
Esto facilita la gestión, evita problemas con nombres repetidos y mejora el rendimiento en consultas.

**3. Atributos añadidos en AULA**

En el enunciado se hablaba de numeración, pero para tenerlo más claro he añadido también planta y edificio, así es más fácil identificar un aula y diferenciarla de otras con el mismo número.

**4. Datos opcionales**

* En PERSONA, el campo sexo es opcional para respetar la preferencia del usuario.
* En MATRICULA, el campo nota también es opcional porque no siempre estará disponible al momento de la inscripción.

**5. Relación ternaria en MATRICULA**

Podría haberse modelado con relaciones intermedias, pero decidí hacerla ternaria para que en una sola tabla se guarde todo lo que tiene sentido junto: alumno, asignatura, convocatoria, precio y nota.

**6. Uso de agregaciones**

En vez de mezclar todo en una sola relación compleja, he usado agregaciones (AUDITORIA\_DOCENTE y IMPARTICION\_PREMIO) para mantener el modelo más legible y organizado.

**7. Restricciones con triggers**

Solo he implementado el trigger que valida que el precio de una matrícula sea mayor que 0, ya que el enunciado no permite usar SELECT en triggers para este apartado.  
El resto los he dejado comentados con su explicación de funcionamiento.

## Listado de tablas y descripción

**1. PERSONA**

Guarda la información básica de todas las personas que entran en la universidad:  
dni, nombre, apellidos, edad, sexo (opcional).

**2. TELEFONO**

Tabla separada para almacenar los teléfonos de cada persona.  
Un mismo DNI puede tener varios teléfonos.

**3. ALUMNO**

Subtipo de PERSONA. Guarda la especialidad que estudia.

**4. PROFESOR**

Subtipo de PERSONA. Guarda el salario anual y el departamento o departamentos donde trabaja.

**5. ADMINISTRATIVO**

Subtipo de PERSONA. Guarda la especialización de su trabajo.

**6. AUDITOR**

Subtipo de PERSONA. Guarda años de experiencia y si trabaja fuera de la universidad.

**7. VISITA**

Subtipo de PERSONA. No tiene datos extra.

**8. DEPARTAMENTO**

Nombre único, años de existencia y primer profesor adscrito.

**9. CURSO\_ORIENTACION**

Curso especial que debe realizar un auditor para poder auditar asignaturas de un departamento.

**10. AULA**

Número, planta, edificio, capacidad y si tiene tablón de anuncios.

**11. ASIGNATURA**

Nombre, profesor que la imparte, aula donde se imparte y posible asignatura prerequisito.

**12. CONVOCATORIA**

Nombre de la convocatoria (ej. “Febrero”, “Junio”…).

**13. MATRÍCULA**

Relación ternaria entre alumno, asignatura y convocatoria, con precio y nota.

**14. REPRESENTA**

Relación reflexiva donde un alumno es representante de otro.

**15. AUDITORIA\_DOCENTE**

Agregación que registra la evaluación de un profesor en una asignatura por un auditor, vinculando también el curso de orientación.

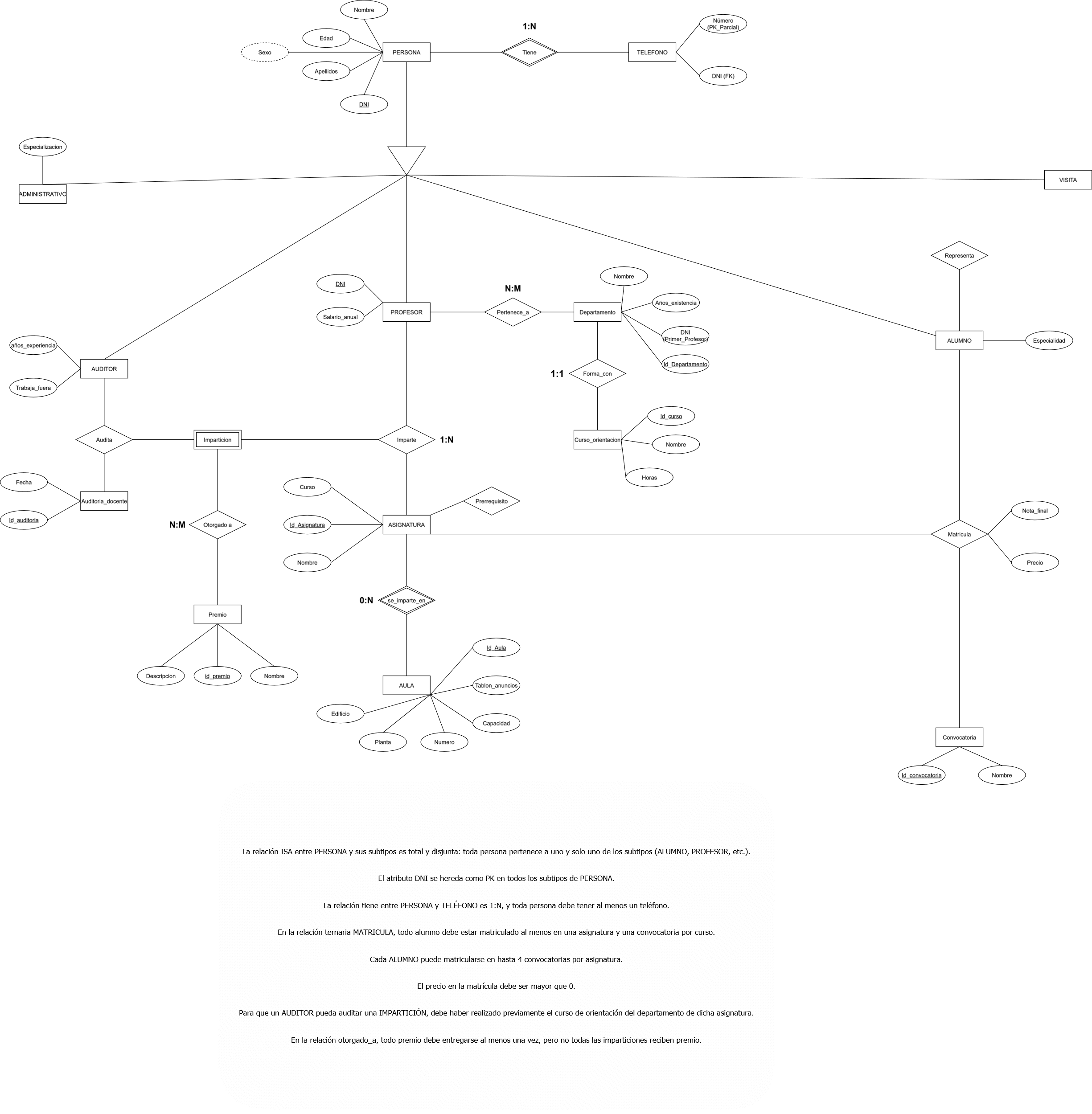
**16. PREMIO**

Premios otorgados a una impartición.

**17. IMPARTICION\_PREMIO**

Relación N:N que asocia premios con imparticiones (profesor + asignatura).

## Diagrama MERE final



## Código SQL completo

Link al proyecto en GitHub 🡪 <https://github.com/rbespada/proyecto_sql_universidad>

En este anexo se incluye el código SQL que crea toda la estructura de la base de datos, incluyendo:

* Creación de la base de datos
* Creación de todas las tablas con sus atributos, claves primarias y foráneas
* Restricciones
* Trigger implementado
* Triggers comentados

*-- ===========================================*

*-- CREACIÓN DE LA BASE DE DATOS*

*-- ===========================================*

*-- Si no existe, se crea una base de datos llamada "Universidad"*

CREATE DATABASE IF NOT EXISTS Universidad;

*-- Activamos la base de datos "Universidad" para usarla a partir de ahora.*

USE Universidad;

*-- ===========================================*

*--  TABLA PERSONA*

*-- ===========================================*

CREATE TABLE PERSONA (

    dni CHAR(9) PRIMARY KEY,  *-- DNI español 9 caracteres, Clave única*

    nombre VARCHAR(50) NOT NULL,

    apellidos VARCHAR(100) NOT NULL,

    edad TINYINT UNSIGNED NOT NULL,  *-- Solo permite valores de 0 a 255*

    sexo ENUM('Masculino', 'Femenino', 'Prefiero no indicar nada') DEFAULT NULL

);

*-- ===========================================*

*--  TABLA TELEFONO (Entidad débil de PERSONA)*

*-- ===========================================*

CREATE TABLE TELEFONO (

    dni CHAR(9),  *-- Clave foránea de la tabla PERSONA*

    numero VARCHAR(20),  *-- Número de teléfono*

    PRIMARY KEY (dni, numero),  *-- Clave compuesta*

    FOREIGN KEY (dni) REFERENCES PERSONA(dni)

        ON DELETE CASCADE  *--Si se elimina una persona de la tabla PERSONA se eliminan tambien el telefono asociado*

        ON UPDATE CASCADE  *--Si se modifica el DNI de la tabla persona se modifica tambien el DNI (Foráneo) en la tabla TELEFONO*

);

*-- ===========================================*

*--  TABLA ALUMNO (subtipo de PERSONA)*

*-- ===========================================*

CREATE TABLE ALUMNO (

    dni CHAR(9) PRIMARY KEY,

    especialidad ENUM('Ingeniería', 'Filología', 'Biología', 'Medicina', 'Humanidades') NOT NULL,  *--Elige especialidad*

    FOREIGN KEY (dni) REFERENCES PERSONA(dni)

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

*-- ===========================================*

*--  TABLA PROFESOR (subtipo de PERSONA)*

*-- ===========================================*

CREATE TABLE PROFESOR (

    dni CHAR(9) PRIMARY KEY,

    salario\_anual DECIMAL(8,2) NOT NULL,  *--Permite 8 digitos en total y dos de ellos decimales, como por ejemplo 599999,95*

    FOREIGN KEY (dni) REFERENCES PERSONA(dni)

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CHECK (salario\_anual > 0 AND salario\_anual <= 60000)  *--Revisa que sea mayor que 0 y menor o igual que 60000*

);

*-- ===========================================*

*--  TABLA ADMINISTRATIVO (subtipo de PERSONA)*

*-- ===========================================*

CREATE TABLE ADMINISTRATIVO (

    dni CHAR(9) PRIMARY KEY,

    especializacion VARCHAR(50) NOT NULL,

    FOREIGN KEY (dni) REFERENCES PERSONA(dni)

        ON DELETE CASCADEI

        ON UPDATE CASCADE

);

*-- ===========================================*

*--  TABLA AUDITOR (subtipo de PERSONA)*

*-- ===========================================*

CREATE TABLE AUDITOR (

    dni CHAR(9) PRIMARY KEY,

    años\_experiencia TINYINT UNSIGNED NOT NULL,

    trabaja\_fuera BOOLEAN NOT NULL,   *-- Especifica entre SI o NO valor booleano 1,0*

    FOREIGN KEY (dni) REFERENCES PERSONA(dni)

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

*-- ===========================================*

*--  TABLA VISITA (subtipo de PERSONA)*

*-- ===========================================*

CREATE TABLE VISITA (

    dni CHAR(9) PRIMARY KEY,

    FOREIGN KEY (dni) REFERENCES PERSONA(dni)

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

*-- ===========================================*

*--  TABLA DEPARTAMENTO*

*-- ===========================================*

CREATE TABLE DEPARTAMENTO (

    is\_departamento INT AUTO\_INCREMENT PRIMARY KEY,  *--Genera un ID automaticamente*

    nombre VARCHAR(100) NOT NULL UNIQUE,  *--Nos asegura que no haya dos nombres iguales con UNIQUE*

    años\_existencia TYNYINT UNSIGNED NOT NULL,

    primer\_profesor CHAR(9),

    FOREIGN KEY (primer\_profesor) REFERENCES PROFESOR(dni)

    ON DELETE SET NULL  *--Si el profesor se elimina lo dejamos en NULL para no perder el departamento*

    ON UPDATE CASCADE

);

*-- ===========================================*

*--  TABLA CURSO\_ORIENTACION*

*-- ===========================================*

CREATE TABLE CURSO\_ORIENTACION (

    id\_curso INT AUTO\_INCREMENT PRIMARY KEY,

    nombre VARCHAR(100) NOT NULL,

    horas TINYINT UNSIGNED NOT NULL,

    id\_departamento INT UNIQUE,

    FOREIGN KEY (id\_departamento) REFERENCES DEPARTAMENTO(id\_departamento)

    ON DELETE CASCADE  *--Si eliminamos el departamento eliminamos tambien su curso asociado*

    ON UPDATE CASCADE

);

*-- ===========================================*

*--  TABLA AULA*

*-- ===========================================*

CREATE TABLE AULA (

    id\_aula INT AUTO\_INCREMENT PRIMARY KEY,

    numero VARCHAR(10) NOT NULL,

    planta VARCHAR(10) NOT NULL,

    edificio VARCHAR(50) NOT NULL,

    capacidad TINYINT UNSIGNED,

    tiene\_tablon BOOLEAN

);

*-- ===========================================*

*--  TABLA ASIGNATURA*

*-- ===========================================*

CREATE TABLE ASIGNATURA (

    id\_asignatura INT AUTO\_INCREMENT PRIMARY KEY,

    nombre VARCHAR(100) NOT NULL,

    id\_profesor CHAR(9) NOT NULL,

    id\_aula INT,

    id\_prerrequisito INT,

    FOREIGN KEY (id\_profesor) REFERENCES PROFESOR(dni)

        ON DELETE RESTRICT  *--No podemos borra un profesor si hay asignaturas que dependen de el*

        ON UPDATE CASCADE,

    FOREIGN KEY (id\_aula) REFERENCES AULA(id\_aula)

        ON DELETE SET NULL

        ON UPDATE CASCADE,

    FOREIGN KEY (id\_prerrequisito) REFERENCES ASIGNATURA(id\_asignatura)

        ON DELETE SET NULL  *--Si eliminamos la asignatura que es prerrequisito) eliminamos la relacion pero no la asignatura principal*

        ON UPDATE CASCADE

);

*-- ===========================================*

*--  TABLA CONVOCATORIA*

*-- ===========================================*

CREATE TABLE CONVOCATORIA (

    id\_convocatoria INT AUTO\_INCREMENT PRIMARY KEY,

    nombre VARCHAR(50) NOT NULL

);

*-- ===========================================*

*--  TABLA MATRICULA (relación ternaria)*

*-- ===========================================*

CREATE TABLE MATRICULA (

    dni\_alumno CHAR(9),

    id\_asignatura INT,

    id\_convocatoria INT,

    PRECIO decimal(6,2) NOT NULL,

    nota DECIMAL(4,2),  *--Puede ser NULL si aún no se ha evaluado*

    PRIMARY KEY (dni\_alumno, id\_asignatura, id\_convocatoria),  *--Clave primaria compuesta por la relacion ternaria*

    FOREIGN KEY (dni\_alumno)REFERENCES ALUMNO(dni)  *--Si se borra un alumno se elimina la matrícula*

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    FOREIGN KEY (id\_asignatura) REFERENCES ASIGNATURA(id\_asignatura)  *--Si se borra una asignatura se elimina la matrícula*

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    FOREIGN KEY (id\_convocatoria) REFERENCES CONVOCATORIA(id\_convocatoria)  *--Si se borra una convocatoria se elimina la matrícula*

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CHECK (precio > 0)  *--Comprobamos que el precio es mayor que 0*

);

*-- ===========================================*

*--  TABLA REPRESENTA (relación reflexiva en ALUMNO)*

*-- ===========================================*

CREATE TABLE REPRESENTA (

    dni\_representado CHAR(9) PRIMARY KEY,

    dni\_representante CHAR(9) NOT NULL,

    FOREIGN KEY (dni\_representado) REFERENCES ALUMNO(dni)

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    FOREIGN KEY (dni\_representante) REFERENCES ALUMNO(dni)

        ON DELETE RESTRICT  *-- No podemos eliminar el representante si otro alumno lo esta usando.*

        ON UPDATE CASCADE

);

*-- ===========================================*

*--  TABLA AUDITORIA\_DOCENTE (agregación)*

*-- ===========================================*

CREATE TABLE AUDITORIA\_DOCENTE (

    id\_auditoria INT AUTO\_INCREMENT PRIMARY KEY,

    dni\_auditor CHAR(9) NOT NULL,

    dni\_profesor CHAR(9) NOT NULL,

    id\_asignatura INT NOT NULL,

    id\_curso\_orientacion INT NOT NULL,

    FOREIGN KEY (dni\_auditor) REFERENCES AUDITOR(dni)

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    FOREIGN KEY (dni\_profesor) REFERENCES PROFESOR(dni)

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    FOREIGN KEY (id\_asignatura) REFERENCES ASIGNATURA(id\_asignatura)

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    FOREIGN KEY (id\_curso\_orientacion) REFERENCES CURSO\_ORIENTACION(id\_curso)

        ON DELETE RESTRICT

        ON UPDATE CASCADE

);

*-- ===========================================*

*--  TABLA PREMIO*

*-- ===========================================*

CREATE TABLE PREMIO (

    id\_premio INT AUTO\_INCREMENT PRIMARY KEY,

    nombre VARCHAR(100) NOT NULL,

    descripcion TEXT  *--Cuadro de texto para describir el premio.*

);

*-- ===========================================*

*--  TABLA IMPARTICION\_PREMIO (agregación)*

*-- ===========================================*

CREATE TABLE IMPARTICION\_PREMIO (

    dni\_profesor CHAR(9),

    id\_asignatura INT,

    id\_premio INT,

    PRIMARY KEY (dni\_profesor, id\_asignatura, id\_premio),  *--La imparticion se define por PROFESOR + ASIGNATURA*

    FOREIGN KEY (dni\_profesor) REFERENCES PROFESOR(dni)

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    FOREIGN KEY (id\_asignatura) REFERENCES ASIGNATURA(id\_asignatura)

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    FOREIGN KEY (id\_premio) REFERENCES PREMIO(id\_premio)

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

*-- ===================================================================*

*--                           [[[TRIGGERS]]]*

*-- ===================================================================*

*-- ===========================================*

*-- Validación de precio de matrícula > 0*

*-- ===========================================*

DELIMITER $$

CREATE TRIGGER tr\_precio\_matricula\_check

BEFORE INSERT ON MATRICULA

FOR EACH ROW

BEGIN

    IF NEW.precio <= 0 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE\_TEXT = 'El precio de la matrícula debe ser mayor que 0'

    END IF;

END$$

DELIMITER ;

*-- ===============================================================*

*-- Un máximo de 4 convocatorias por asignatura*

*-- ===============================================================*

*-- Este trigger debería activarse antes de insertar una nueva matrícula.*

*-- Se consulta cuántas convocatorias tiene ya ese alumno*

*-- para la misma asignatura, y cancela la inserción si hay 4 o más.*

*-- Evento: BEFORE INSERT ON MATRICULA*

*-- Lógica:*

*--   SELECT COUNT(\*) FROM MATRICULA*

*--   WHERE dni\_alumno = NEW.dni\_alumno*

*--     AND id\_asignatura = NEW.id\_asignatura;*

*-- Si el resultado ≥ 4  lanzar error (con SIGNAL SQLSTATE)*

*-- ===============================================================*

*-- No permitir representantes sin representados*

*-- ===============================================================*

*-- Este trigger debería ejecutarse después de un DELETE en la tabla REPRESENTA para verificar si el alumno que era representante sigue teniendo representados.*

*-- Evento: AFTER DELETE ON REPRESENTA*

*-- Lógica esperada:*

*--   SELECT COUNT(\*) FROM REPRESENTA*

*--   WHERE dni\_representante = OLD.dni\_representante;*

*-- Si el resultado = 0  lanzar error (no se puede dejar a un representante sin ningún representado)*

*-- =====================================================================*

*-- Validar curso de orientación antes de auditar*

*-- =====================================================================*

*-- Este trigger debería activarse antes de insertar en AUDITORIA\_DOCENTE*

*-- Evento: BEFORE INSERT ON AUDITORIA\_DOCENTE*

*-- Lógica esperada:*

*--   1. Obtener el departamento al que pertenece la asignatura*

*--   2. Verificar que el curso de orientación realizado por el auditor*

*--      pertenece a ese mismo departamento.*

*-- Consulta que sería necesaria:*

*--   SELECT d.id\_departamento*

*--   FROM ASIGNATURA a*

*--   JOIN DEPARTAMENTO\_ASIGNATURA da ON a.id\_asignatura = da.id\_asignatura*

*--   JOIN DEPARTAMENTO d ON da.id\_departamento = d.id\_departamento*

*--   WHERE a.id\_asignatura = NEW.id\_asignatura;*

*-- Comparar ese id\_departamento con el departamento del curso que hizo el auditor.*

*-- Si NO coinciden lanzar SIGNAL con error*