Para diseñar el modelo entidad-relación de un Instituto, vamos a seguir un proceso paso a paso, creando primero el modelo conceptual, luego el modelo lógico y finalmente el modelo físico.

### Modelo Conceptual

El modelo conceptual es la representación más abstracta de los datos y las relaciones entre ellos.

Entidades:

1. Profesor

2. Módulo

3. Alumno

4. Curso

5. Grupo

Relaciones:

- Un profesor imparte varios módulos (1:N)

- Un módulo es impartido por un solo profesor (1:1)

- Un alumno está matriculado en varios módulos (M:N)

- Un curso tiene varios grupos (1:N)

- Un grupo pertenece a un solo curso (1:1)

- Un grupo tiene varios alumnos (1:N)

- Un alumno puede ser delegado de un grupo (1:1)

### Modelo Lógico

El modelo lógico representa las entidades y sus relaciones en términos de tablas y campos.

Tablas:

1. Profesor

- DNI (clave primaria)

- Nombre

- Dirección

- Teléfono

2. Módulo

- Código (clave primaria)

- Nombre

- DNI\_Profesor (clave foránea)

3. Alumno

- Nº\_Legajo (clave primaria)

- Nombre

- Apellido

- Fecha\_Nacimiento

4. Curso

- ID\_Curso (clave primaria)

- Nombre

5. Grupo

- ID\_Grupo (clave primaria)

- ID\_Curso (clave foránea)

- Nº\_Legajo\_Delegado (clave foránea)

6. Matricula

- Nº\_Legajo (clave foránea)

- Código\_Módulo (clave foránea)

Relaciones:

- Un profesor imparte varios módulos (1:N): Módulo.DNI\_Profesor -> Profesor.DNI

- Un módulo es impartido por un solo profesor (1:1): Módulo.DNI\_Profesor -> Profesor.DNI

- Un alumno está matriculado en varios módulos (M:N): Matricula(Nº\_Legajo, Código\_Módulo)

- Un curso tiene varios grupos (1:N): Grupo.ID\_Curso -> Curso.ID\_Curso

- Un grupo pertenece a un solo curso (1:1): Grupo.ID\_Curso -> Curso.ID\_Curso

- Un grupo tiene varios alumnos (1:N): Matricula.Nº\_Legajo -> Alumno.Nº\_Legajo

- Un alumno puede ser delegado de un grupo (1:1): Grupo.Nº\_Legajo\_Delegado -> Alumno.Nº\_Legajo

### Modelo Físico

El modelo físico representa cómo se implementarán las tablas y relaciones en una base de datos específica. Aquí usaremos MySQL como ejemplo.

```sql

CREATE DATABASE Instituto;

USE Instituto;

CREATE TABLE Profesor (

DNI VARCHAR(9) PRIMARY KEY,

Nombre VARCHAR(50),

Dirección VARCHAR(100),

Teléfono VARCHAR(15)

);

CREATE TABLE Módulo (

Código CHAR(5) PRIMARY KEY,

Nombre VARCHAR(50),

DNI\_Profesor VARCHAR(9),

FOREIGN KEY (DNI\_Profesor) REFERENCES Profesor(DNI)

);

CREATE TABLE Alumno (

Nº\_Legajo INT AUTO\_INCREMENT PRIMARY KEY,

Nombre VARCHAR(50),

Apellido VARCHAR(50),

Fecha\_Nacimiento DATE

);

CREATE TABLE Curso (

ID\_Curso INT AUTO\_INCREMENT PRIMARY KEY,

Nombre VARCHAR(50)

);

CREATE TABLE Grupo (

ID\_Grupo INT AUTO\_INCREMENT PRIMARY KEY,

ID\_Curso INT,

Nº\_Legajo\_Delegado INT,

FOREIGN KEY (ID\_Curso) REFERENCES Curso(ID\_Curso),

FOREIGN KEY (Nº\_Legajo\_Delegado) REFERENCES Alumno(Nº\_Legajo)

);

CREATE TABLE Matricula (

Nº\_Legajo INT,

Código\_Módulo CHAR(5),

PRIMARY KEY (Nº\_Legajo, Código\_Módulo),

FOREIGN KEY (Nº\_Legajo) REFERENCES Alumno(Nº\_Legajo),

FOREIGN KEY (Código\_Módulo) REFERENCES Módulo(Código)

);

```

### Resumen y Mejores Prácticas

En este diseño hemos seguido varias mejores prácticas:

1. Normalización: Las tablas están normalizadas para evitar redundancia de datos y mejorar la integridad referencial.

2. Claves primarias: Se han definido claves primarias únicas para cada tabla.

3. Claves foráneas: Se han establecido relaciones entre tablas usando claves foráneas.

4. Tabla intermedia: La tabla Matricula actúa como una relación muchos-a-muchos entre Alumno y Módulo.

5. Autoincremento: Los campos que requieren valores únicos pero no son identificadores naturales (como ID\_Curso e ID\_Grupo) utilizan autoincremento.

6. Tipos de datos apropiados: Se han elegido tipos de datos adecuados para cada campo (VARCHAR para cadenas, DATE para fechas, etc.).

Este diseño permite almacenar todos los datos solicitados y mantener las relaciones entre ellos de manera eficiente y sin redundancias.

Citations: