**APUNTES BASES DE DATOS UD7: Diseño Relacional**

**FASES**

**ANÁLISIS**

* Diseño conceptual (¿Qué problema voy a resolver?) **[UD4 ED]**

**DISEÑO**

* Diseño lógico (¿Cómo voy a resolver el problema?) **[UD7 BD]**

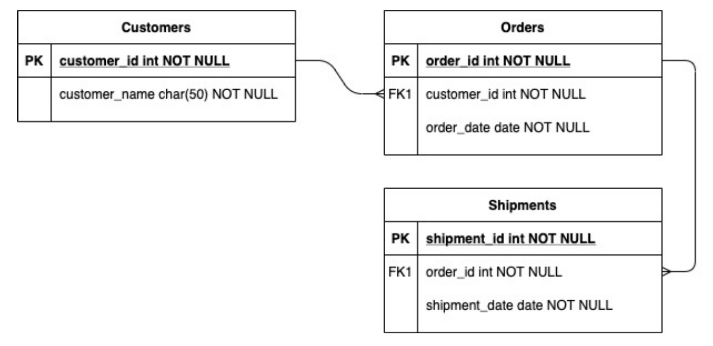
**CODIFICACIÓN**

* Diseño físico (Lo resuelvo “asín”.) **[UD6 BD]**

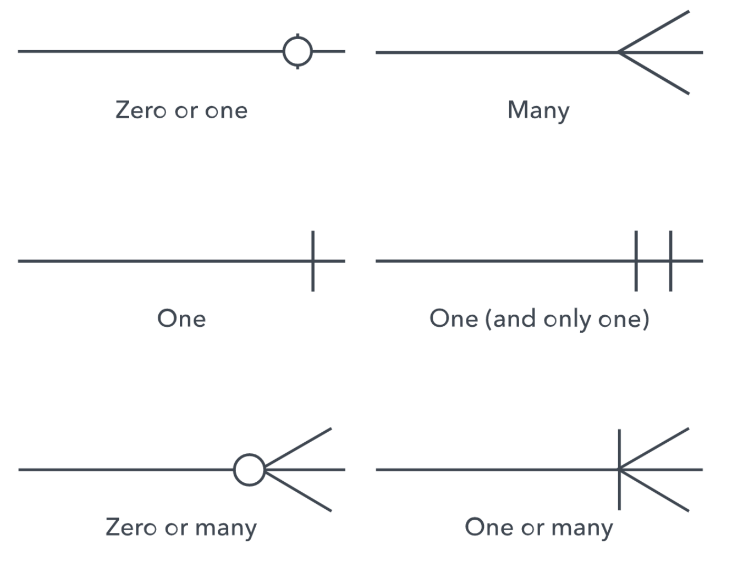
El concepto de encapsulamiento no existe en el diseño relacional. Da lo mismo un atributo público que privado que protegido que empaquetado.

| **Tipo / Fase** | **Análisis** | **Diseño** | **Codificación** |
| --- | --- | --- | --- |
| **Estructurado** | Modelo E/R | Grafo relacional | Script SQL (CREATE TABLE) |
| **Orientado a objetos** | Diagrama de clases UML (análisis) | Diagrama de clases UML (diseño) | Conjunto de clases en Java |
| **Bases de datos** | Diagrama E/R [notación UML] | Grafo relacional [diagrama de patas de gallo] | Script SQL [CREATE TABLE] |

Ejemplo de diagrama de patas de gallo:



Simbología del diagrama **(no lo vamos a utilizar, pero para saberlo)**:

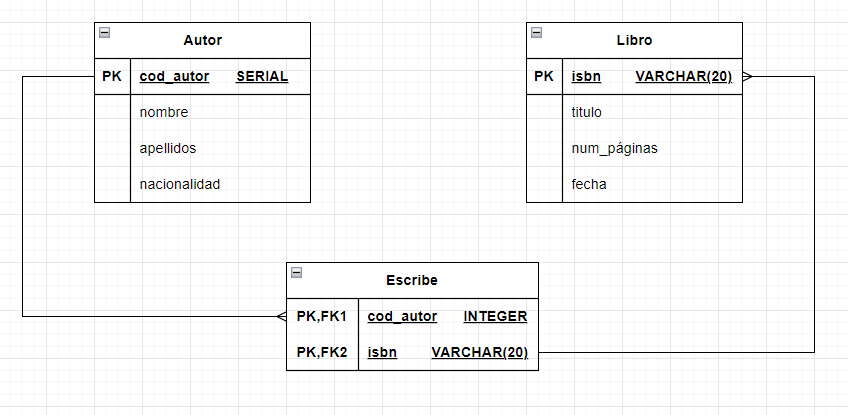


**DIAGRAMA DE RELACIONES (N-M)**

Diagrama corriente:

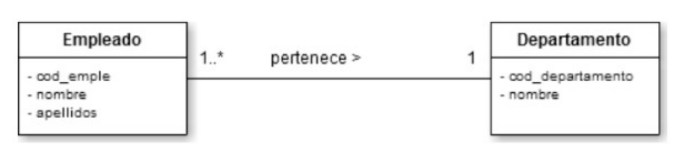


**Ejemplo de diagrama de relación N-M en draw.io:**

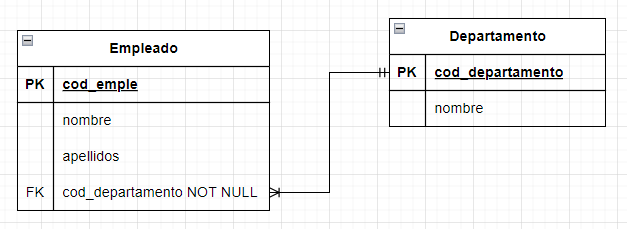


**DIAGRAMA DE RELACIONES (1-N)**

Diagrama corriente:

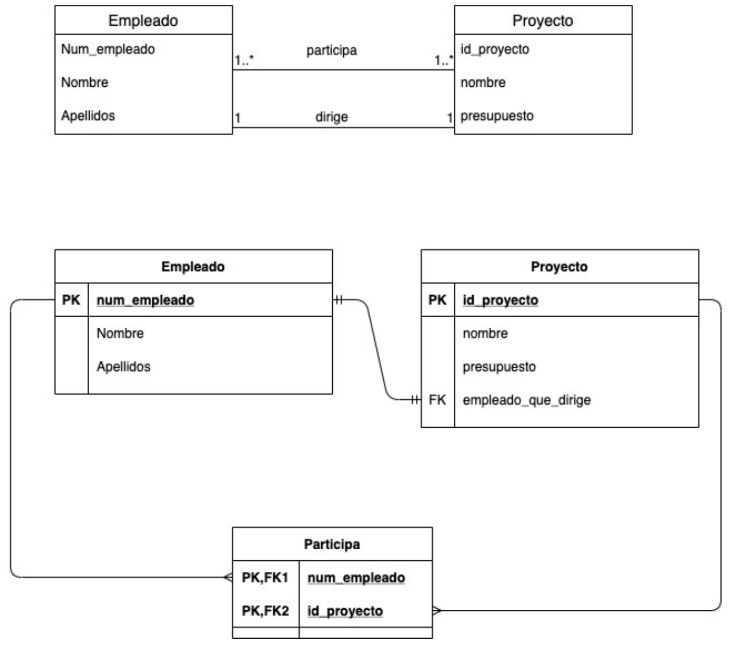


**Ejemplo de diagrama de relación 1-N en draw.io:**



**DIAGRAMA DE RELACIONES (1-1)**

En este caso, la relación 1:1 es la relación “Empleado” dirige “Proyecto”:



**ASOCIACIONES**

**N:M** 🡪 Trasladar:

* Cada clase como tabla.
* La asociación como 3ª tabla:
  + Los atributos son PKs.
  + Todos los atributos conforman PK.

**1:N** 🡪 Trasladar:

* Cada clase como tabla.
* Propaga la PK lado 1 a lado N como FK.

**1:N (Composición)** 🡪 Trasladar:

* Cada clase como tabla.
* Propaga la PK de lado 1 a lado N como FK o PK.

**1:1** 🡪 A gusto del consumidor, se propaga la PK de un lado 1 al otro como FK.

**En el 99% que tengamos una clase de asociación siempre necesitaremos una tercera tabla.**

**PASOS PARA HACER UN DIAGRAMA DE RELACIÓN CORRECTO**

**1 -** Trasladar cada clase como tabla.

**2 -** Trasladar asociaciones.

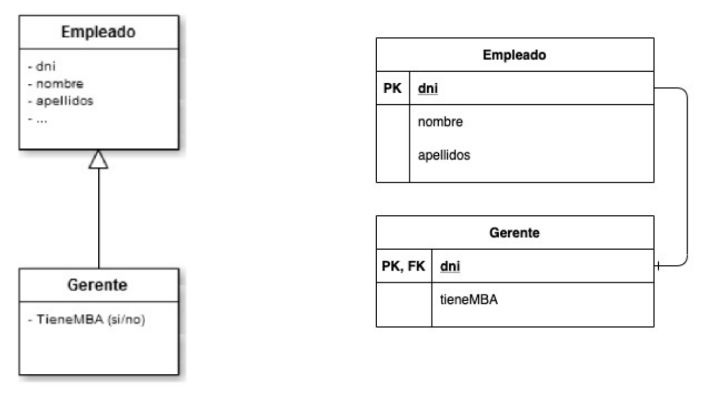
**2.1 -** Composición (ya que modifican las PKs de otras tablas).

**2.2 -** N:M

**2.3 -** 1:N, 1:1

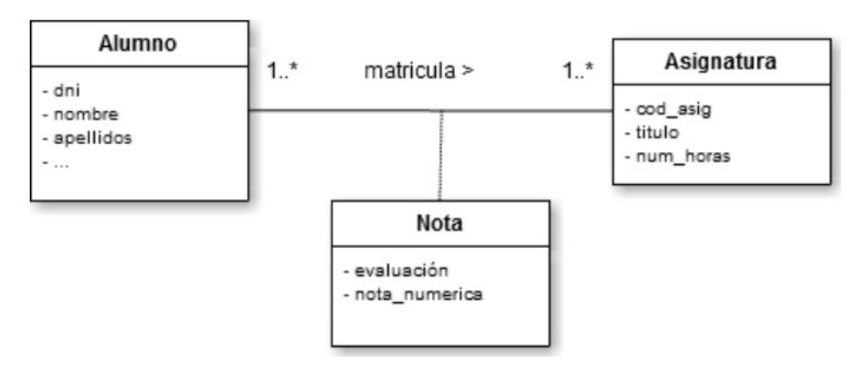
| **MULTIPLICIDAD** | **SOLUCIÓN** |
| --- | --- |
| N:M | Nueva tabla PK + FK |
| 1:N | PK1 🡪 FK lado N |
| 1:1 | PK1 🡪 FK1 (Semántica) |
| Clase asoc. | Nueva tabla, PK depende de la multiplicidad. |

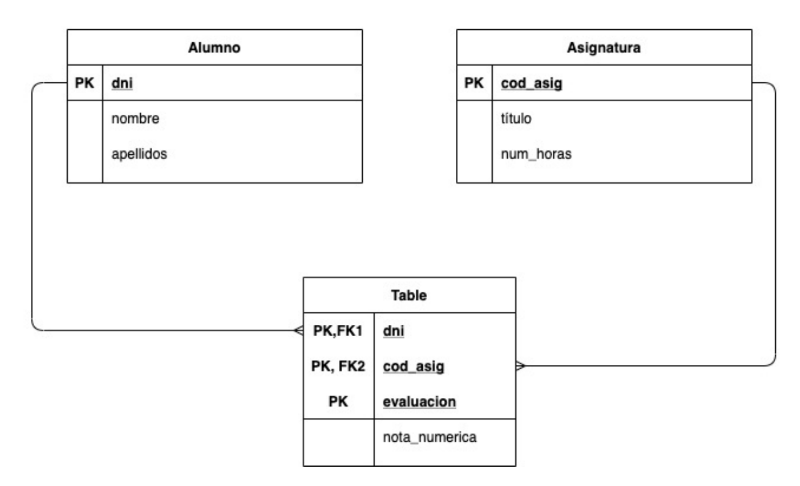
**HERENCIA**



SI LA HERENCIA TIENE UNA ASOCIACIÓN Y LA CLASE MADRE ES ABSTRACTA, ESTA SE TIENE QUE DIBUJAR.

**ASOCIACIÓN CON CLASE DE ASOCIACIÓN**





**Si hay herencia:**

* ¿La clase tiene asociaciones?
  + SÍ → Se traslada como tabla. **(JOINED)**
  + NO:
    - **JOINED** si la clase base es abstracta.
    - **TABLE PER SUBCLASS** si la clase base no es abstracta.
* ¿Alguna de las hijas tiene alguna asociación?
  + SÍ → Se trasladan todas las hijas del mismo nivel de jerarquía como tabla. **(JOINED)**
  + NO → ¿Tiene atributos?
    - SÍ → **JOINED**.
    - NO → **Single table + discriminante en tabla base**.

