Unidad Didáctica 4:
Diseño de Bases de Datos Relacionales

Parte 2:
Diseño Conceptual

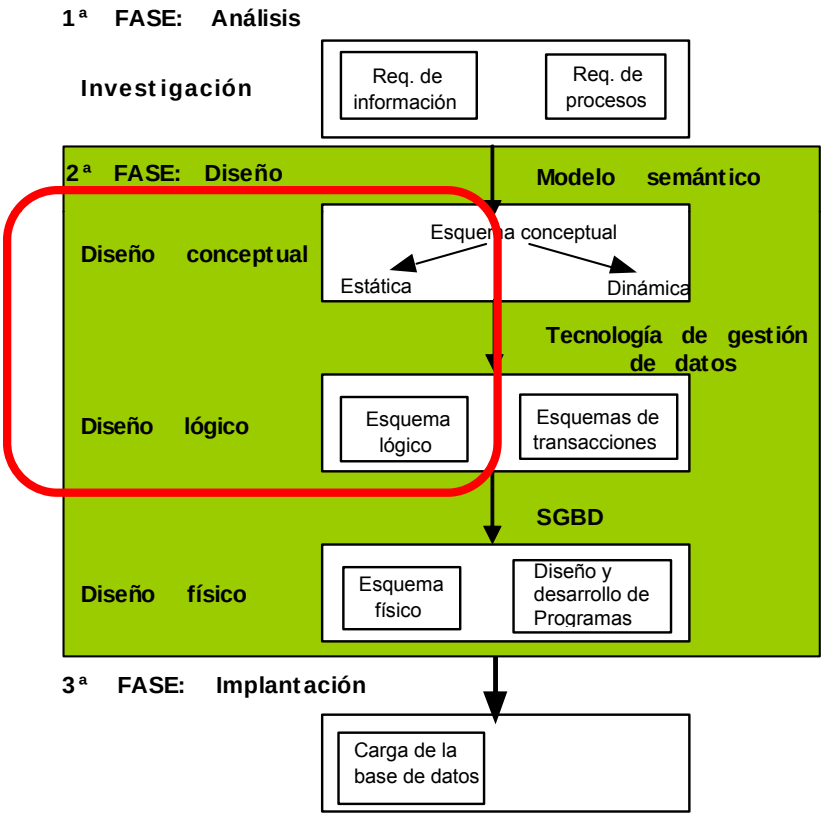
U.D. 4.2

DSIC Bases de Datos y Sistemas de información
Departamento de Sistemas Informáticos y Computación / Universidad Politécnica de Valencia

V. 15.12

UD 4.1 Conceptos básicos de diseño. UML

1 Introducción



UD 4.1 Conceptos básicos de diseño. UML

2. Diseño Conceptual

Diseño conceptual: fase del diseño de una BD cuyo objetivo es "obtener una representación de la realidad que capture las propiedades estáticas y dinámicas de la misma que son necesarias para satisfacer los requisitos; esta representación debe suponer una imagen fiel del comportamiento del mundo real".

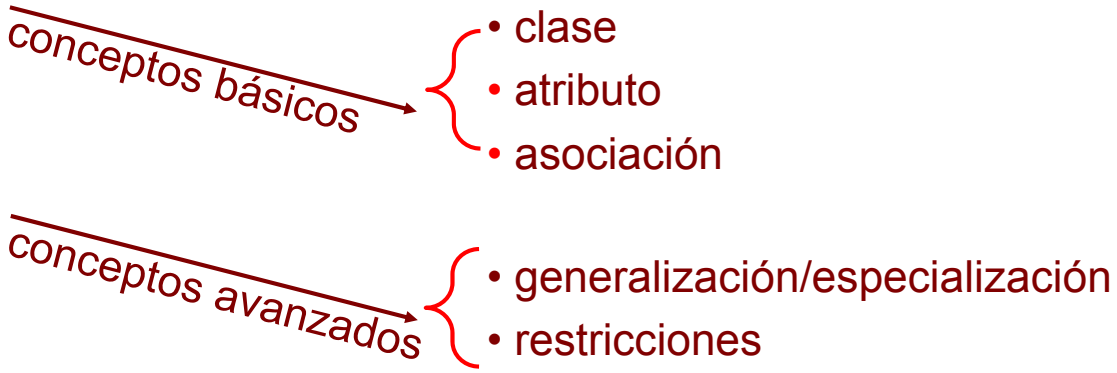
¿qué se va a usar para esa representación?

Diagrama de clases del UML

UD 4.1 Conceptos básicos de diseño. UML

2.1 El diagrama de clases del UML

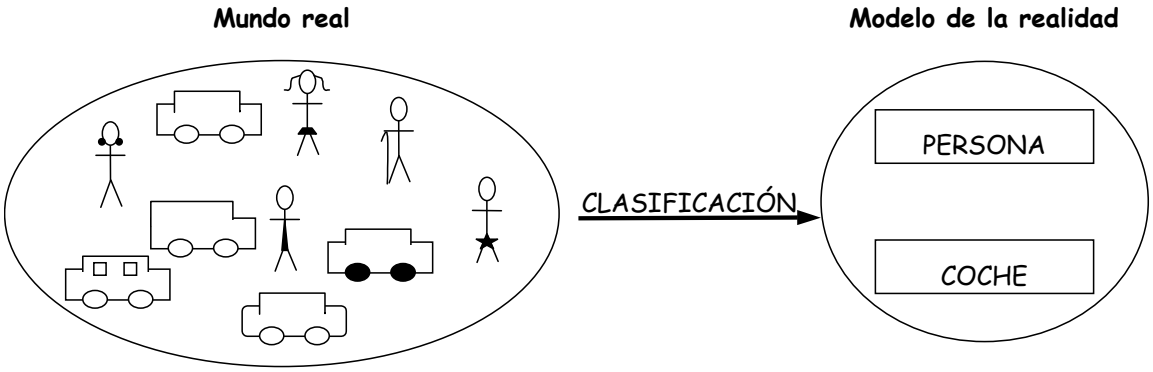
El *diagrama de clases* permite representar las estructuras que constituyen el contenido del sistema de información junto con restricciones de distintos tipos que limitan las ocurrencias válidas de las mismas.



UD 4.1 Conceptos básicos de diseño. UML

2.1.1 Clase

La observación de la realidad permite detectar el conjunto de "objetos" (físicos o conceptuales) de los que se quiere almacenar información para, mediante el uso de la clasificación, que es uno de los mecanismos de abstracción más primario que existen, descubrir el conjunto de "clases" (o tipos de objetos) que son de interés para la organización

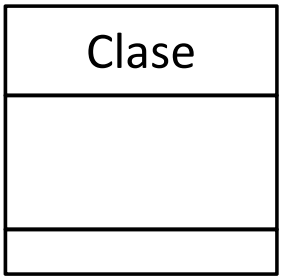


2.1.1 Clase

- Los componentes básicos de un SI son los objetos de los que se quiere almacenar información; todos los objetos que son del mismo tipo se representan con un una clase.

  Una clase es la descripción de un grupo de objetos con estructura, comportamiento y relaciones similares.

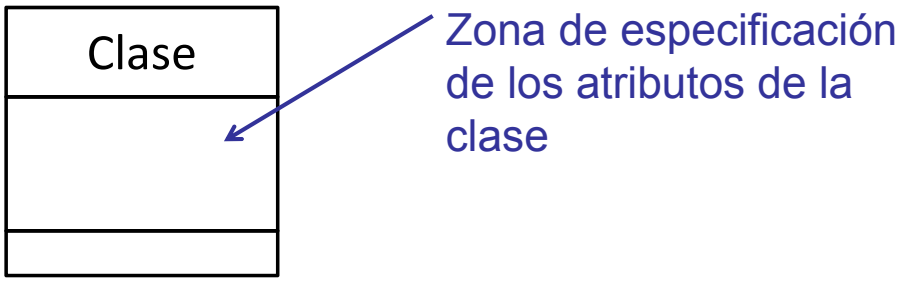
- Con una clase se representará cualquier persona, concepto, suceso o evento (en definitiva cualquier "cosa") sobre el que se quiera almacenar información.



UD 4.1 Conceptos básicos de diseño. UML

2.1.2 Atributo

Un atributo es una propiedad de la clase identificado con un nombre cuyas instancias pueden tomar valor de un conjunto que se especifica.



2.1.2 Atributo

A cada atributo se le puede especificar:

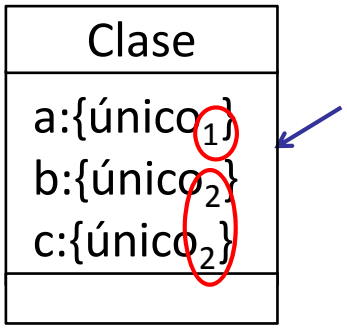
- Tipo de dato asociado que determina los valores que puede tomar el atributo:
  - El nombre de un tipo de datos conocido.
  - Una enumeración de valores posibles entre paréntesis.
  - Registro.

- Restricciones:
  - De unicidad
  - De cardinalidad
  - De identificación

2.1.2 Atributo

Restricción de unicidad

Las distintas ocurrencias de una clase deben tomar valores distintos para el atributo (o conjunto de atributos) sobre los que se define



No puede haber dos ocurrencias de la clase
1. con el mismo valor en *a*
2. ni con el mismo valor a la vez en *b* y en *c*.

2.1.2 Atributo

Restricción de cardinalidad

Expresan el número de valores que puede tomar el atributo para cada ocurrencia de la clase

.

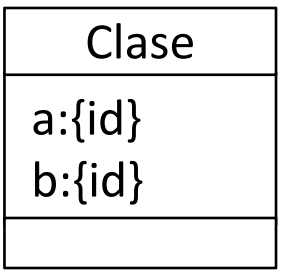
- {1..1}: el atributo tiene exactamente un valor para cada ocurrencia de la clase. ---> VNN.
- {0..1}: el atributo puede tener como mucho un valor para cada ocurrencia de la clase o puede no tener valor (es el valor por defecto).
- {1..*}: el atributo puede tener más de un valor para cada ocurrencia de la clase pero al menos tiene un valor.
- {0..*}: el atributo puede tener más de un valor para cada ocurrencia de la clase o puede no tener valor.

2.1.2 Atributo

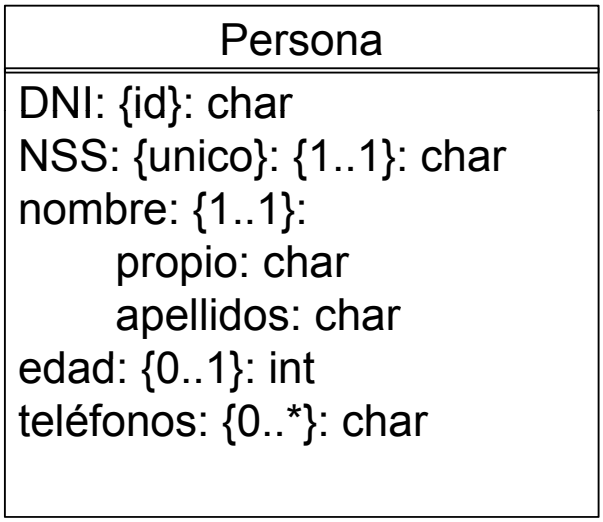
Restricción de identificación

Un identificador es un conjunto de atributos con restricción de unicidad y con cardinalidad {1..1} que permite distinguir dos ocurrencias de la clase.

Sólo hay un identificador aunque puede constar de varios atributos



2.1.2 Atributo



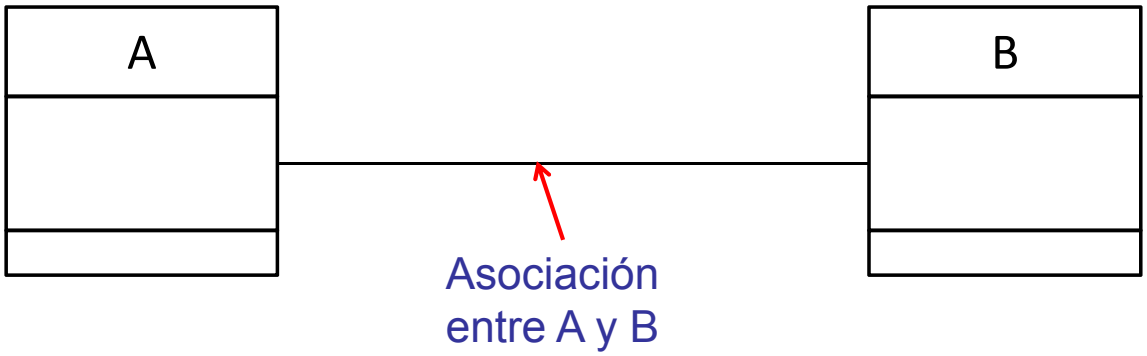
UD 4.1 Conceptos básicos de diseño. UML

2.1.3 Asociación

Las asociaciones permiten representar las posibles relaciones existentes entre las clases del sistema de información.

> Una asociación es una relación estructural que especifica que los objetos de una clase están conectados con los objetos de otra.

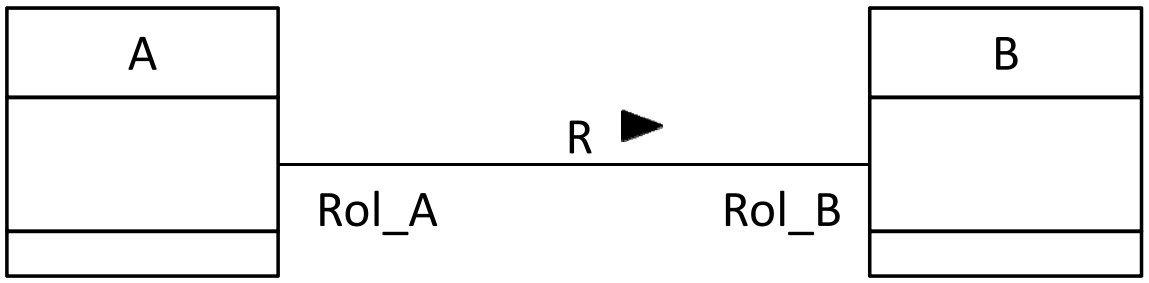
Una asociación que conecta exactamente dos clases se dice que es binaria.



2.1.3 Asociación

Adornos en una asociación:

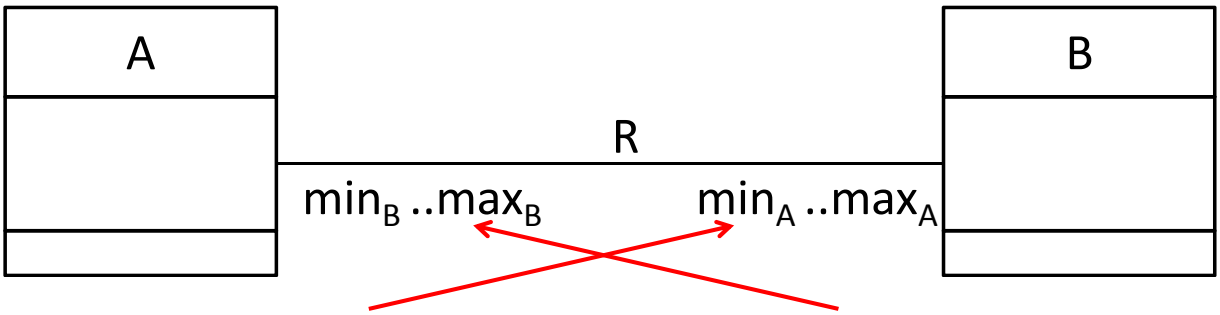
- Sentido
- Nombre
- Rol (opcional, para clarificar la relación)
- Cardinalidad



2.1.3 Asociación

Cardinalidad

$R(A(min_A..max_A), B(min_B..max_B))$



Cada ocurrencia de *A* se relaciona con, como mínimo min_A ocurrencias de *B* y como máximo con max_A ocurrencias de *B*

Cada ocurrencia de *B* se relaciona con, como mínimo min_B ocurrencias de *A* y como máximo con max_B ocurrencias de *A*

2.1.3 Asociación

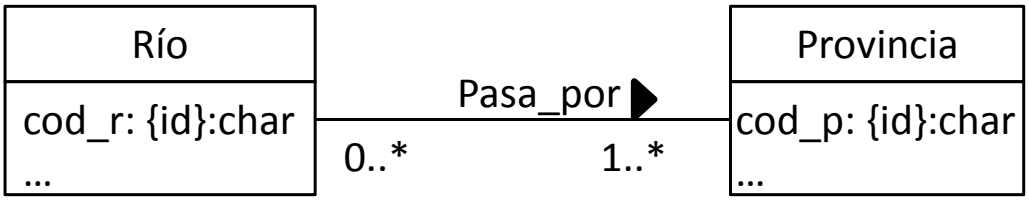
Cardinalidad: Valores más usuales

- min_A=1 y max_A=1
- min_A=0 y max_A=*
- min_A=0 y max_A=1
- min_A=1 y max_A=*

Cuando la cardinalidad mínima de una clase es distinta de 0 se dice que esa clase tiene restricción de existencia respecto a la asociación.

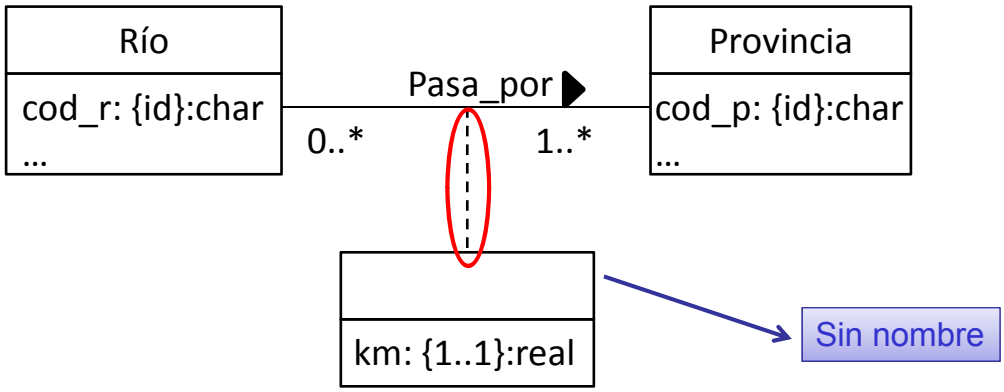
2.1.3 Asociación (ejemplo)

Diagrama que representa la información de qué ríos pasan por qué provincias:
"Un río puede pasar por varias provincias, al menos por una, y por una provincia pueden pasar varios ríos o ninguno."



2.1.3 Asociación. Atributos de enlace

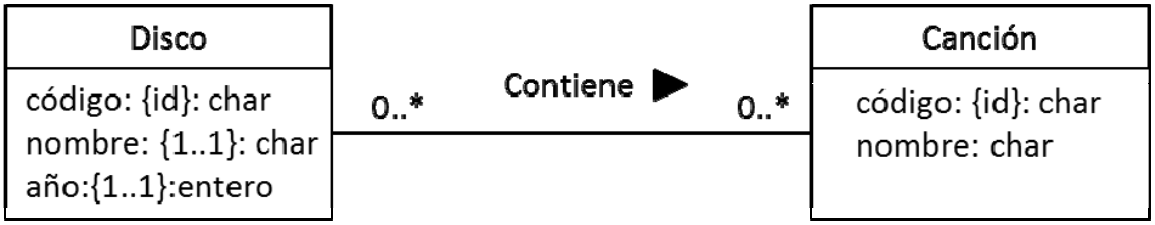
Si se quiere incluir la información: "durante cuántos kilómetros pasa cada río por cada provincia que atraviesa", la asociación *Pasa_por* tiene un *atributo de enlace (o clase anónima)*



LOS ATRIBUTOS DE ENLACE NO SE PUEDEN CONECTAR CON CLASES, SE DEBEN PROMOCIONAR A CLASE (PONIÉNDOLES NOMBRE)

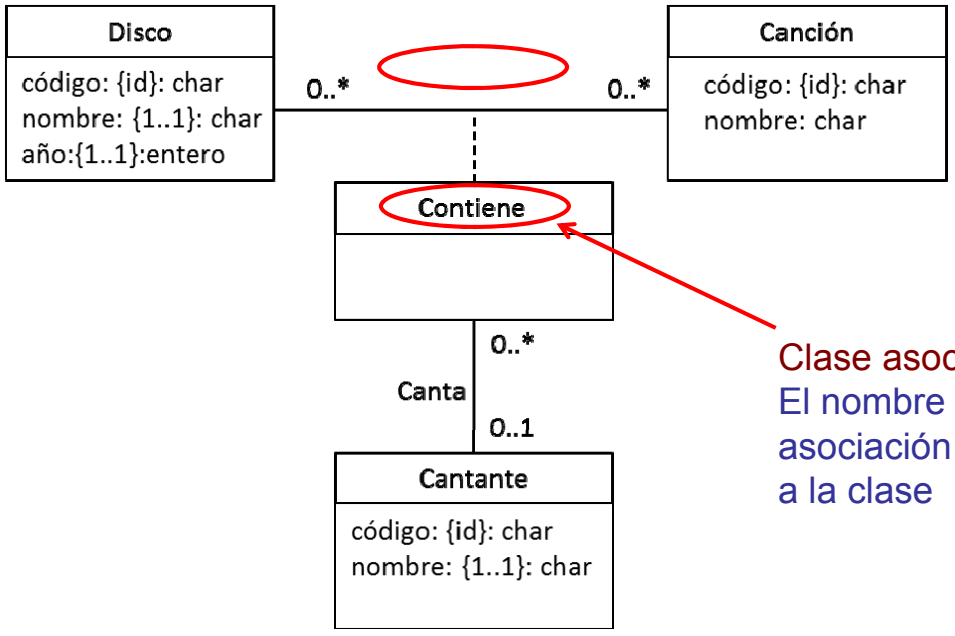
2.1.3 Asociación. Clase asociación.

El diagrama representa la información sobre discos y canciones y las canciones que contiene cada disco.



2.1.3 Asociación. Clase asociación.

…además se quiere información sobre cantantes y :
"qué cantante canta cada canción en un disco"



Clase asociación
El nombre de la asociación se traslada a la clase

UD 4.1 Conceptos básicos de diseño. UML

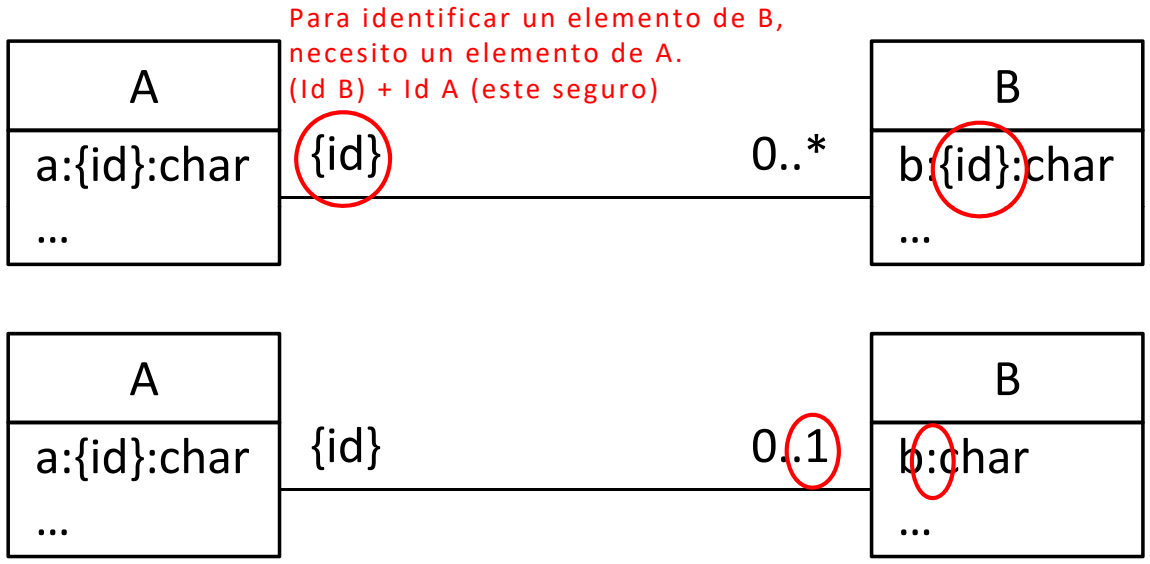
2.1.4 Clases débiles

Una clase sufre restricción de dependencia de identificación cuando no puede identificarse con sus propios atributos. Sus ocurrencias son distinguibles gracias a su asociación con otra/s clase/s.

A este tipo de clases se les denomina clases débiles.

Esta restricción se representa con la etiqueta {id} en lugar de la cardinalidad. La clase débil cuenta con el o los atributos identificadores de la/s clase/s de las que depende (no se representan explícitamente).

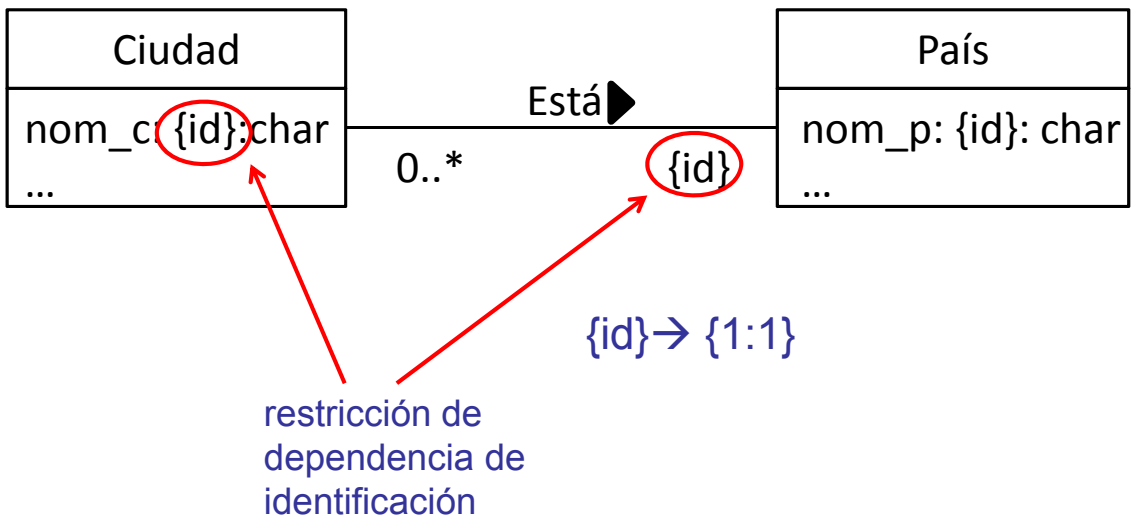
2.1.4 Clases débiles



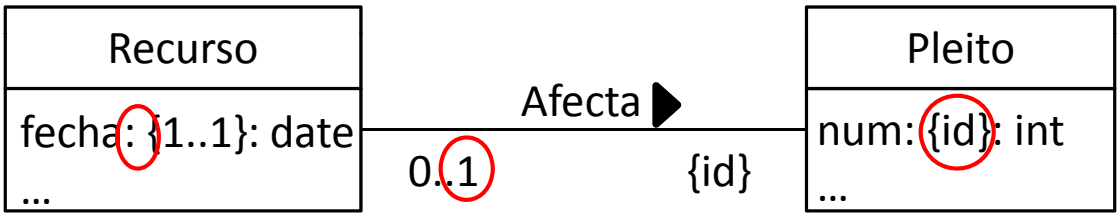
Clases débiles

2.1.4 Clases débiles (Ejemplo)

Se necesita el nom_c junto con el nom_p para identificar la ciudad.
Ej. Valencia. ¿De Venezuela o España?)



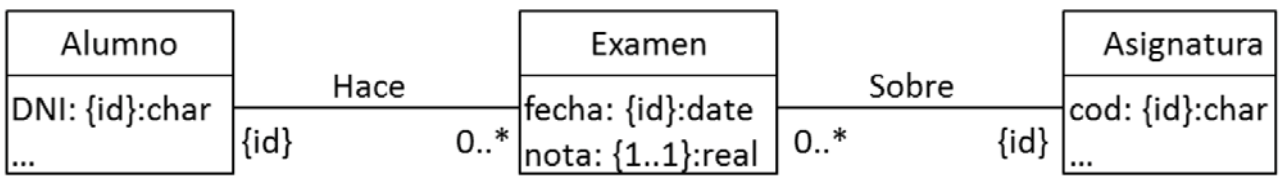
2.1.4 Clases débiles (Ejemplo)



Un litigio solo puede tener un recurso:
Se puede identificar un recursos por el litigio al que pertenece

2.1.4 Clases débiles (Ejemplo)

Se quiere almacenar información sobre las notas que han obtenido los alumnos en distintas asignaturas teniendo en cuenta que un alumno se puede presentar varias veces a una asignatura, pero en fechas distintas, y que los exámenes no tienen identificador.



UD 4.1 Conceptos básicos de diseño. UML

2.1.5 Asociaciones ternarias

Aunque en el diagrama de clases de UML podrían representarse explícitamente relaciones de grado > 2 vamos a utilizar sólo asociaciones binarias.

Las relaciones ternarias conseguiremos representarlas mediante:

a) clases asociación o
b) clases débiles.

2.1.5 Asociaciones ternarias (ejemplo)

Ejemplo:

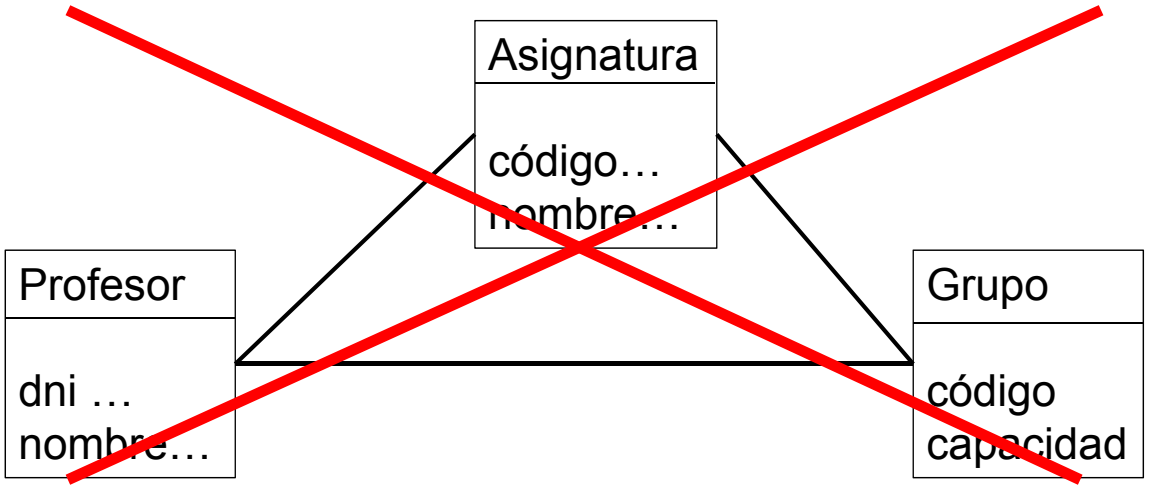
Asociación ternaria entre profesor-asignatura-grupo:

- Un profesor puede impartir cualquier número de asignaturas en cualquier número de grupos
- Una asignatura puede ser impartida por cualquier número de profesores y de grupos
- Un grupo puede tener cualquier número de asignaturas impartidas por cualquier número de profesores

2.1.5 Asociaciones ternarias (ejemplo)

Ejemplo:

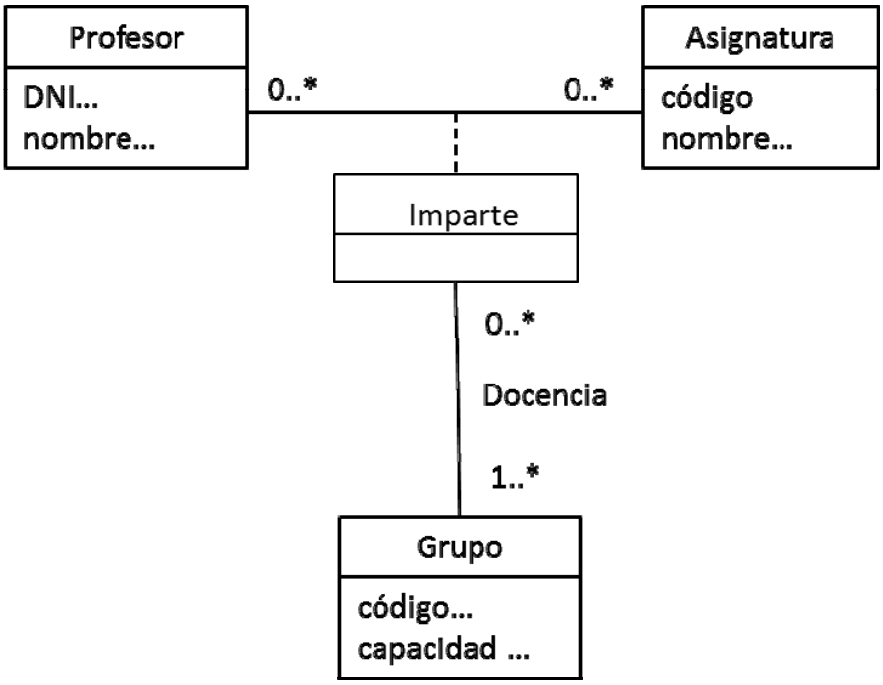
- No se puede modelar como 3 asociaciones binarias porque perderíamos información.



Un profesor imparte varias asignaturas. Un profesor imparte varios grupos pero ¿en qué grupo imparte cada asignatura un profesor ?

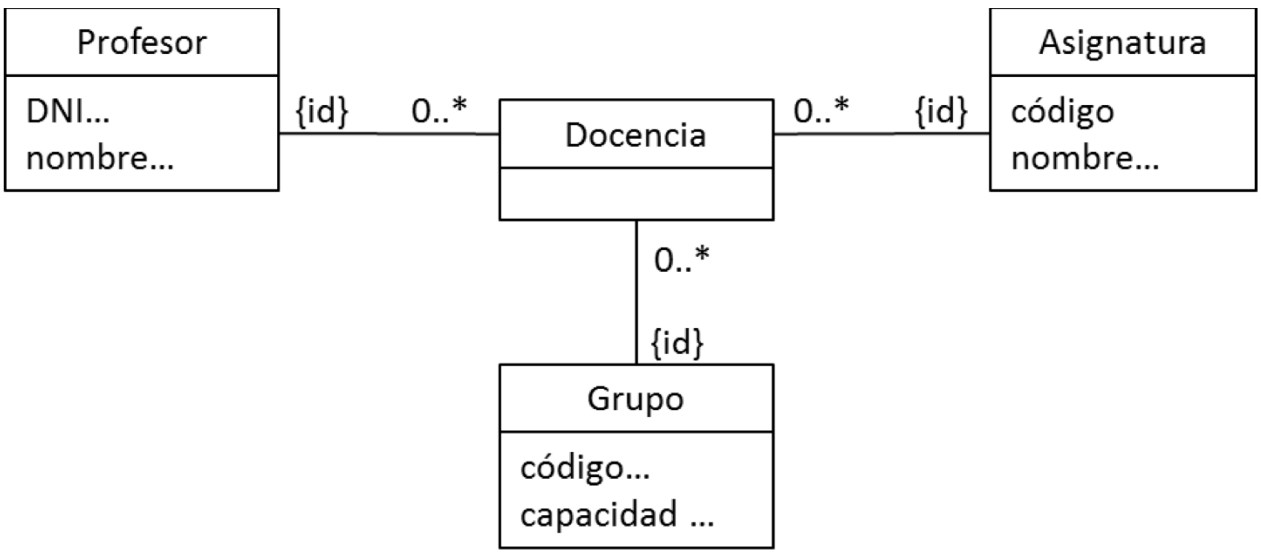
2.1.5 Asociaciones ternarias (ejemplo)

A) Asociación ternaria entre profesor-asignatura-grupo representada usando una clase asociación *Imparte*



2.1.5 Asociaciones ternarias (ejemplo)

B) Asociación ternaria entre profesor-asignatura-grupo representada usando una clase débil *Docencia*



UD 4.1 Conceptos básicos de diseño. UML

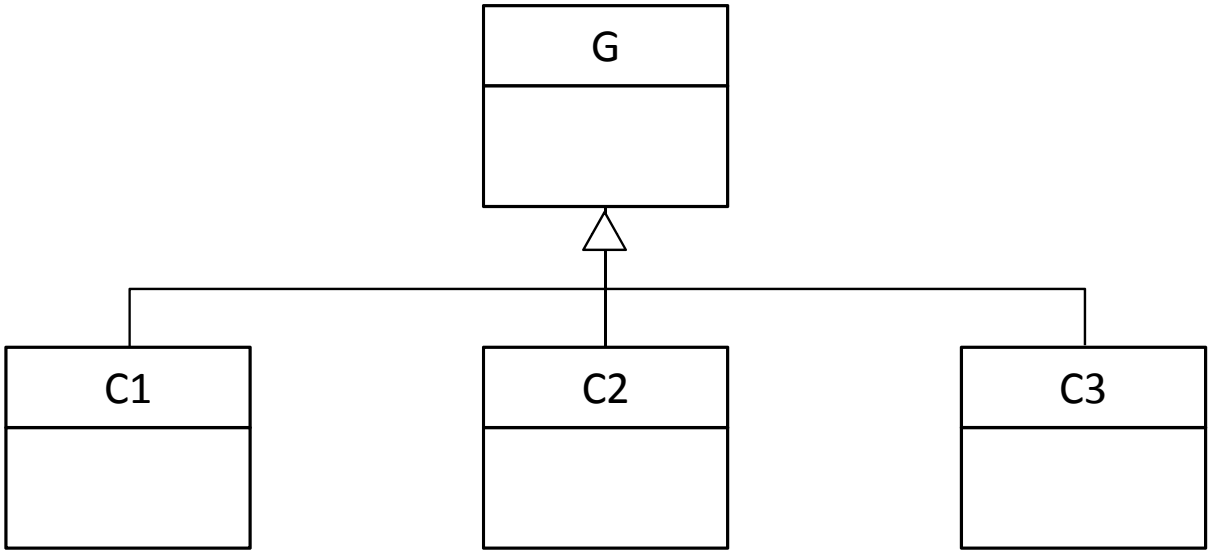
2.1.6 Generalización/Especialización

Cuando entre distintas clases existe una relación de inclusión, se expresa por medio de la Generalización/Especialización.

La clase más general se especializa en una o varias subclases, o dicho a la inversa, una o varias clases se generalizan en una superclase.

Las subclases, además de sus atributos propios, tienen todos los atributos de sus superclases (en cualquier nivel), aunque no se representan en el diagrama.

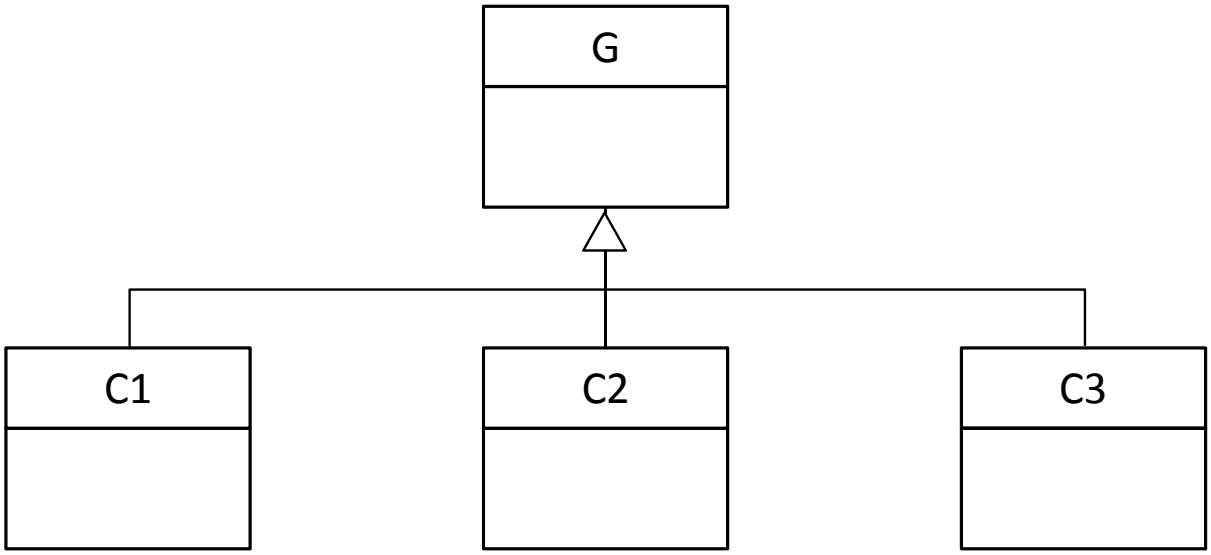
2.1.6 Generalización/Especialización



Total: toda ocurrencia de la clase *G* se especializa en *C*1, *C*2 o *C*3

Parcial: puede haber ocurrencias de *G* que no se especialicen ni en *C*1 ni en *C*2 ni en *C*3. (Valor por defecto)

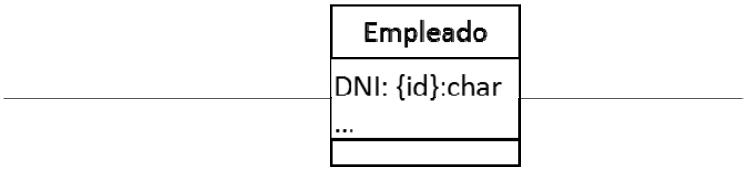
2.1.6 Generalización/Especialización



Disjunta: no puede haber ocurrencias de *G* que estén a la vez en dos subclases

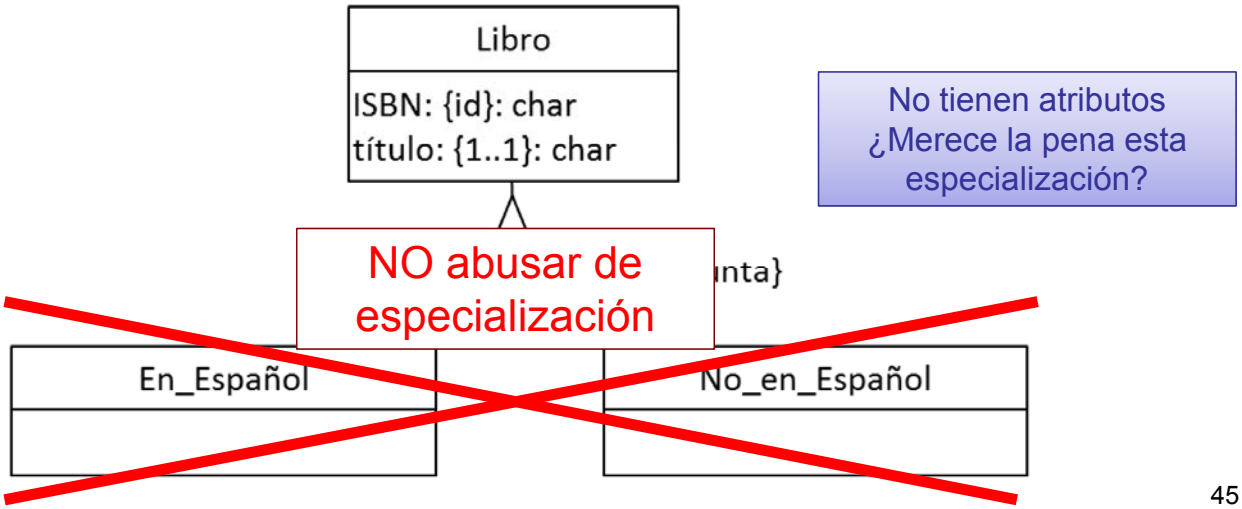
Solapada: puede haber ocurrencias de *G* que se especialicen en más de una subclase. (Valor por defecto)

2.1.6 Generalización/Especialización. Ejemplo



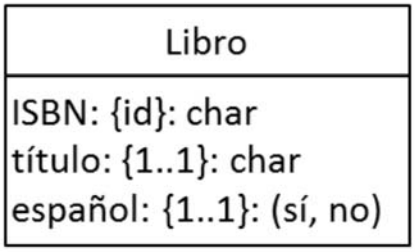
2.1.6 Generalización/Especialización. Ejemplo

En una biblioteca se almacena de cada libro su ISBN y su título.
Los libros pueden estar escritos en español o en cualquier otra lengua.



2.1.6 Generalización/Especialización. Ejemplo

En una biblioteca se almacena de cada libro su ISBN y su título.
Los libros pueden estar escritos en español o en cualquier otra lengua.



Ejemplo

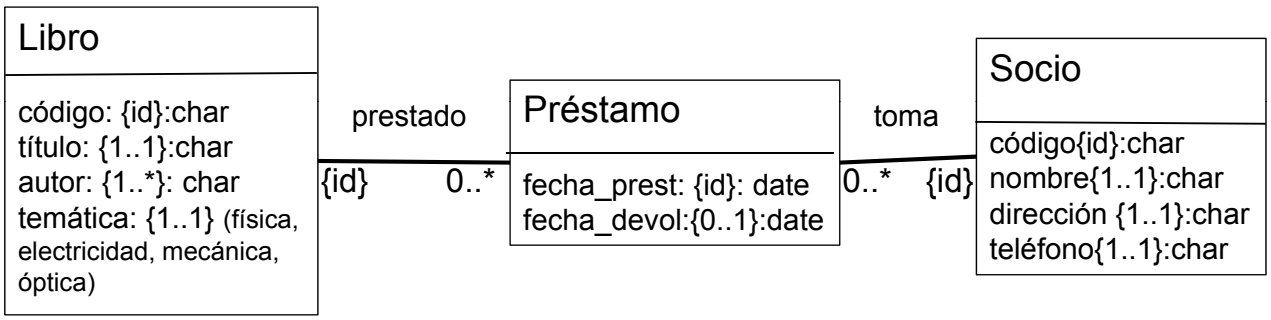
Se desea diseñar una BD para la gestión de una biblioteca. Se han identificado los requisitos siguientes:

- Consultar los datos de un libro: código del libro, título, autor (o autores), temática y en caso de estar prestado, el socio que lo tiene en préstamo.
- Consultar la información sobre un socio: código del socio, nombre, dirección, teléfono y libros que tiene en préstamo así como la fecha de préstamo.
- Gestionar la información sobre todos los préstamos: Prestar un libro y registrar su devolución

Algunas restricciones de integridad que se han detectado son:

- El código del libro identifica unívocamente al libro.
- El código del socio identifica unívocamente al socio.
- Los temas utilizados para clasificar un libro son: física, electricidad, mecánica y óptica.

Ejemplo



UD 4.1 Conceptos básicos de diseño. UML

2.2 Obtención del diagrama de clases

1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales

1. Identificar clases con sus atributos

Para cada tipo de objeto de la realidad se definirá una clase.

Una clase viene definida por los atributos que representan la información que se desea conocer de cada tipo de objeto.

Para cada atributo se debe:

- Asociar un dominio o, si es derivado, especificar la forma de derivación; y
- Especificar la cardinalidad y las restricciones de unicidad.

- Elegir identificador; si no existe, la clase será débil y habrá que decidir sobre qué asociaciones se apoya para identificarse.
- Posibilidad de reconsiderar clases elegidas …

2. Identificar generalizaciones/especializaciones

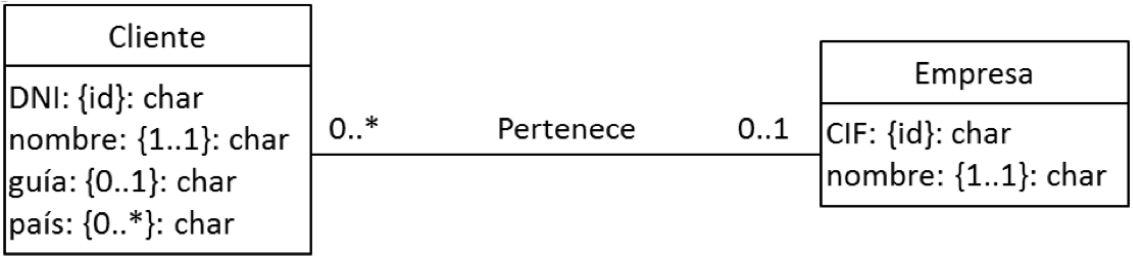
1. Estrategia descendente (especialización)
2. Estrategia ascendente (generalización)
3. Jerarquización (es_un)

No abusar del mecanismo.

2. Identificar generalizaciones/especializaciones

1. Estrategia descendente (especialización)

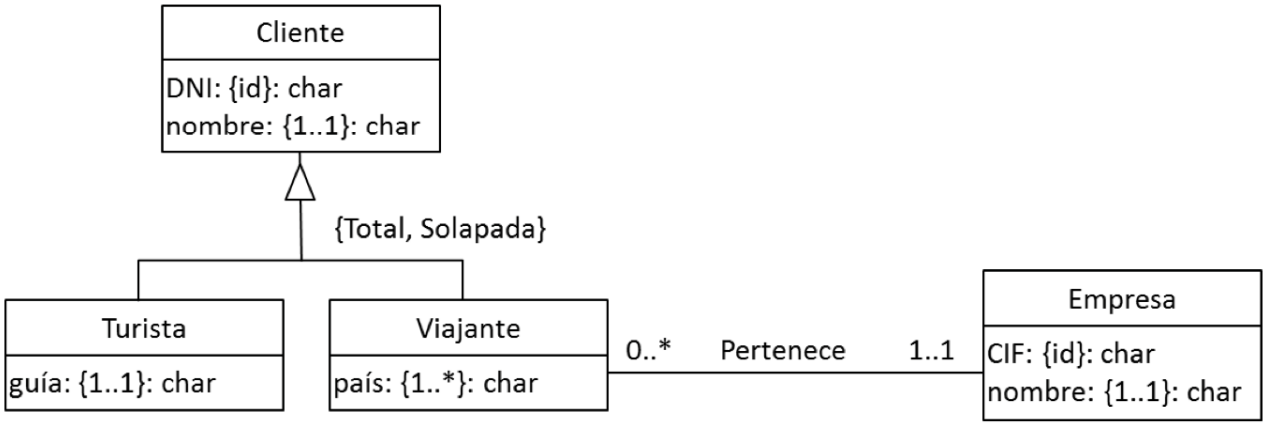
Especialización: proceso por el que se clasifica una clase de objetos en subclases más especializadas.



2. Identificar generalizaciones/especializaciones

1. Estrategia descendente (especialización)

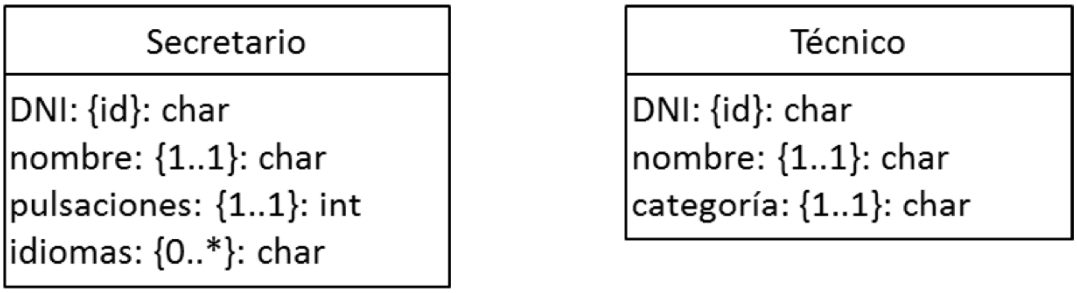
Especialización: proceso por el que se clasifica una clase de objetos en subclases más especializadas.



2. Identificar generalizaciones/especializaciones

2. Estrategia ascendente (generalización)

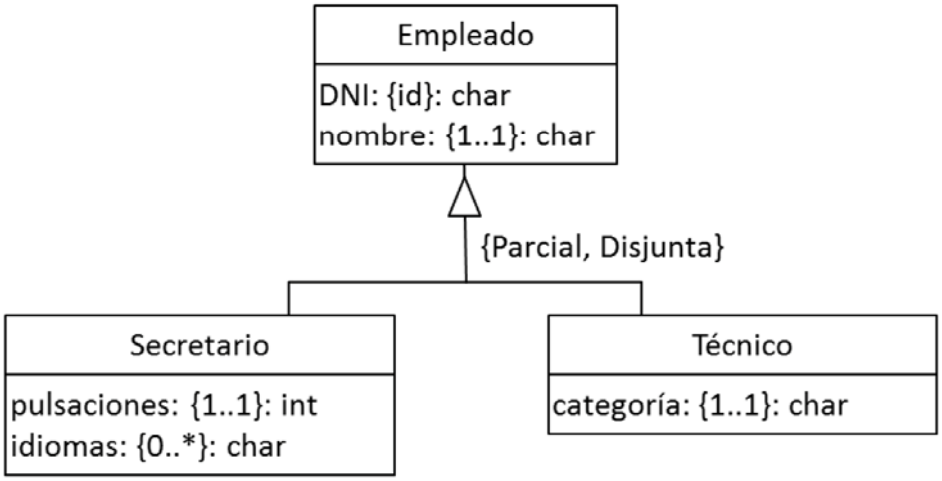
Generalización: Proceso por el que se generalizan varias clases para obtener una abstracta de más alto nivel que incluya los objetos de todas estas clases.



2. Identificar generalizaciones/especializaciones

2. Estrategia ascendente (generalización)

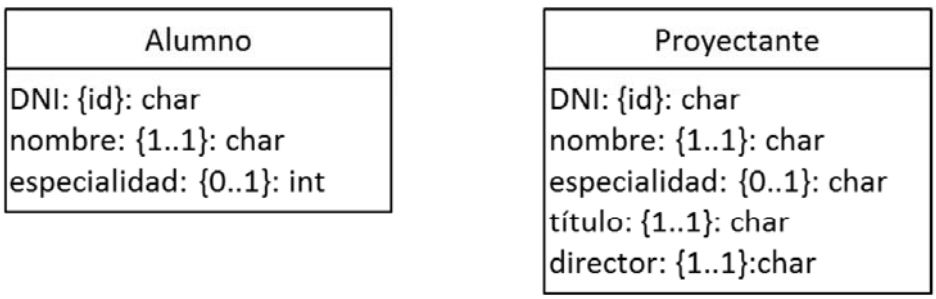
Generalización: Proceso por el que se generalizan varias clases para obtener una abstracta de más alto nivel que incluya los objetos de todas estas clases.



2. Identificar generalizaciones/especializaciones

3. Jerarquización (es_un)

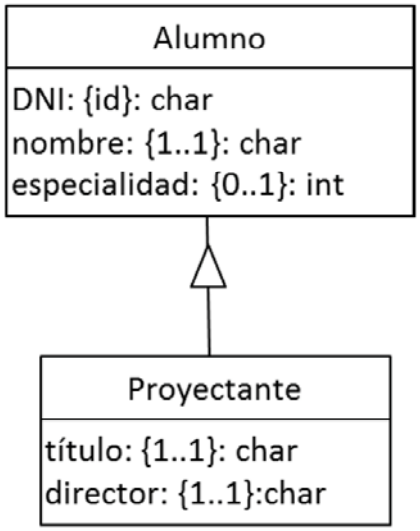
Jerarquización: Clases entre las que existe una relación “es_un”



2. Identificar generalizaciones/especializaciones

3. Jerarquización (es_un)

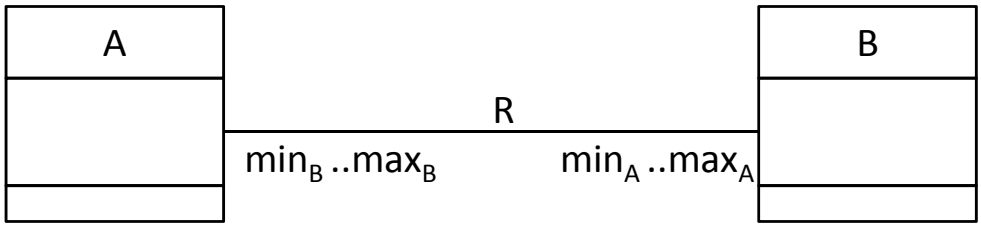
Jerarquización: Clases entre las que existe una relación “es_un”



3. Identificar asociaciones entre clases

Estudiar asociaciones entre clases.

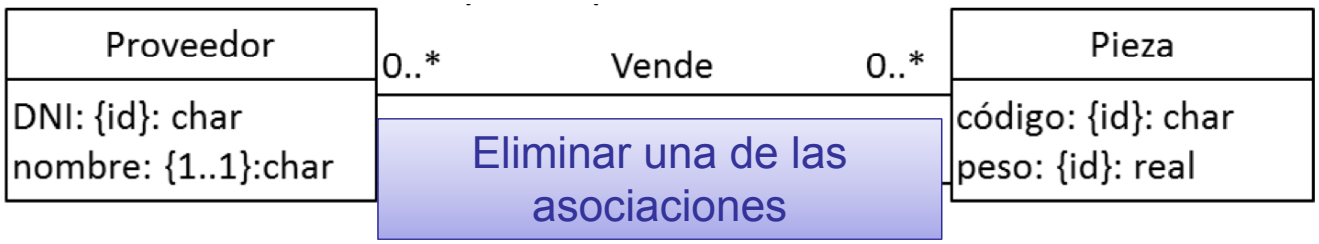
3.1. Determinan las cardinalidades máximas y mínimas.



Ante la duda elegir las cardinalidades menos restrictivas.

3. Identificar asociaciones entre clases

3.2. Eliminar asociaciones redundantes



3. Identificar asociaciones entre clases

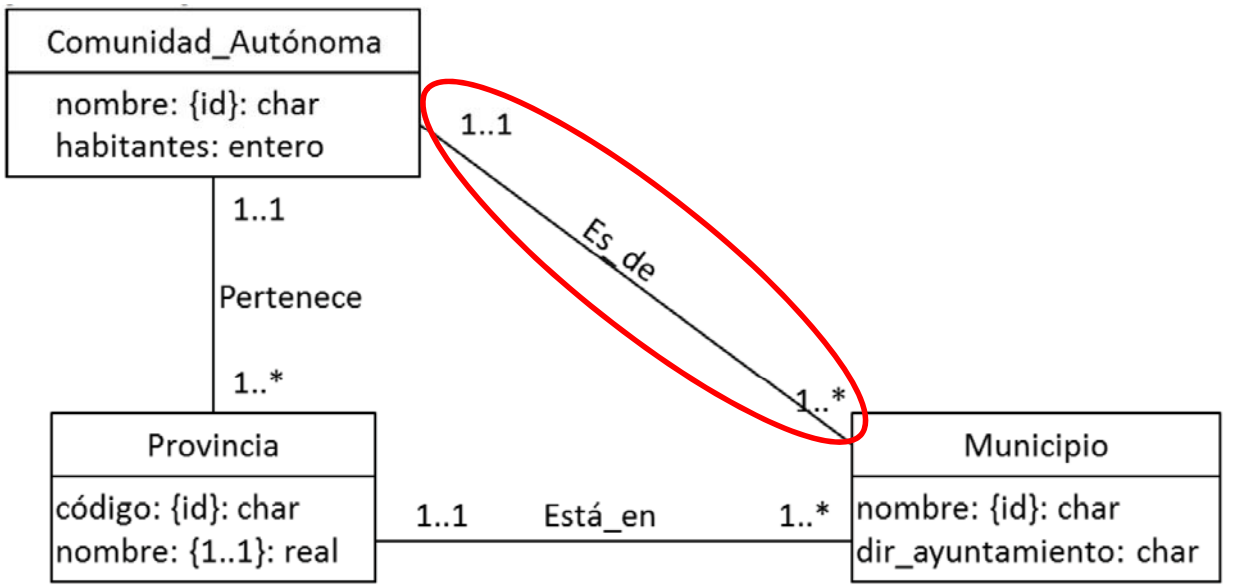
3.2. Eliminar asociaciones redundantes.

Las 2 asociaciones son necesarias:
Representan información distinta

3. Identificar asociaciones entre clases

3.3. Eliminar redundancias por ciclos.

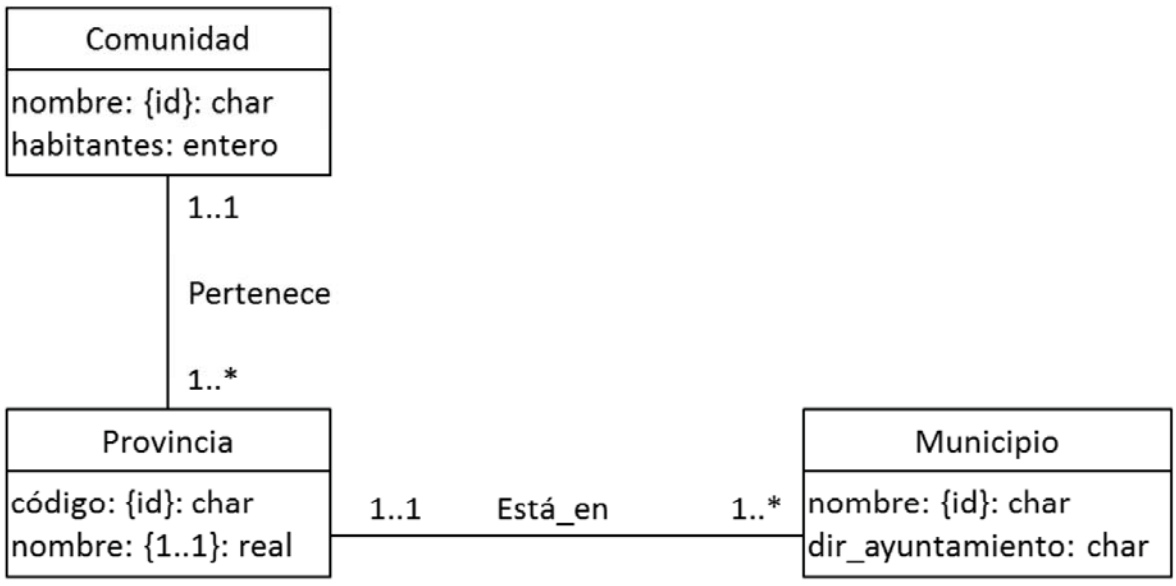
Una asociación es redundante por dependencias transitivas
cuando pueda derivarse a través de otras asociaciones (dos o más)



3. Identificar asociaciones entre clases

3.3. Eliminar redundancias por ciclos.

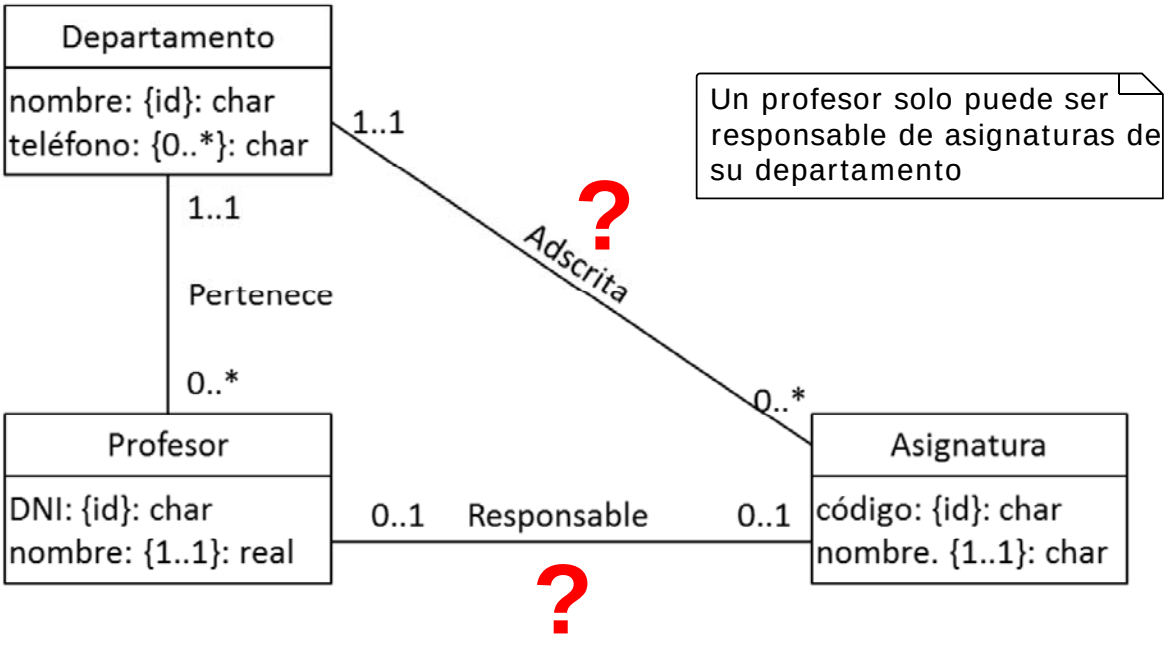
Una asociación es redundante por dependencias transitivas
cuando pueda derivarse a través de otras asociaciones (dos o más)



3. Identificar asociaciones entre clases

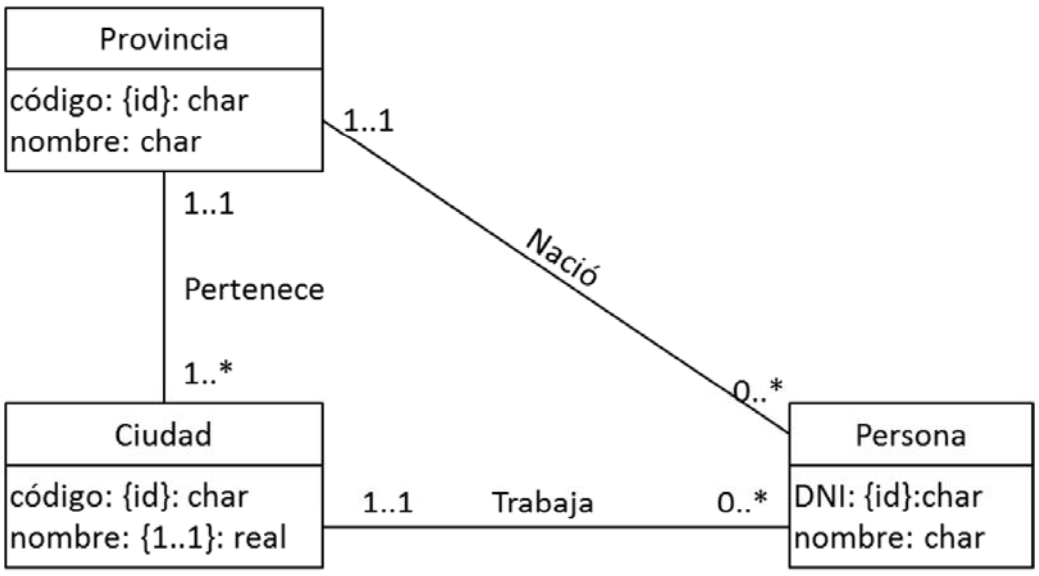
3.3. Eliminar redundancias por ciclos.

No siempre puede eliminarse



3. Identificar asociaciones entre clases

3.3. Eliminar redundancias por ciclos.

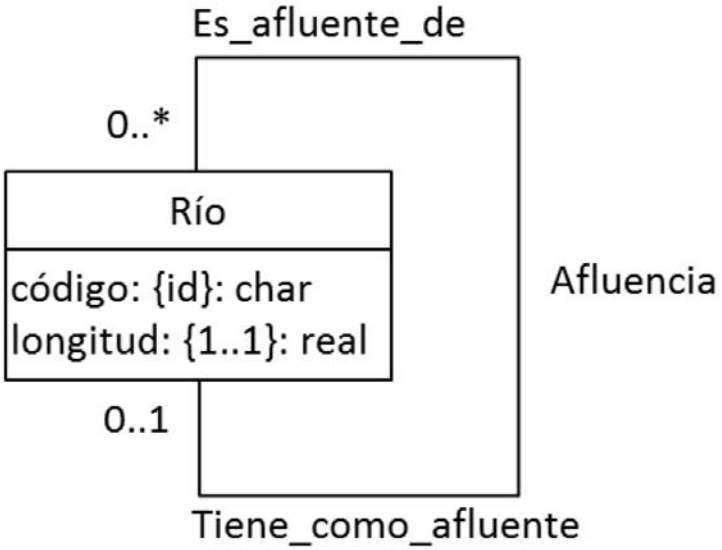


No eliminar:

Una persona no tiene por qué haber nacido en la misma provincia en la que está la ciudad en la que trabaja.

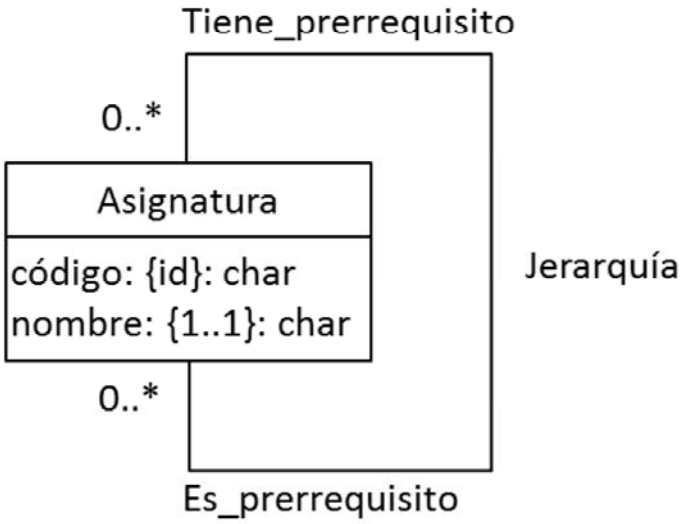
3. Identificar asociaciones entre clases

3.4. Especificar roles en asociaciones de una clase consigo misma



3. Identificar asociaciones entre clases

3.4. Especificar roles en asociaciones de una clase consigo misma



3. Identificar asociaciones entre clases

3.4. Especificar roles en asociaciones de una clase consigo misma

Cuidado con las asociaciones reflexivas:
Normalmente exigen que se especifiquen ciertas propiedades que no quedan contempladas en la definición de la asociación.

Ejemplo anterior (prerrequisitos)

- Es antisimétrica:
  Una asignatura no puede ser prerrequisito de un prerrequisito suyo

- Es antirreflexiva:
  Una asignatura no puede ser prerrequisito de sí misma

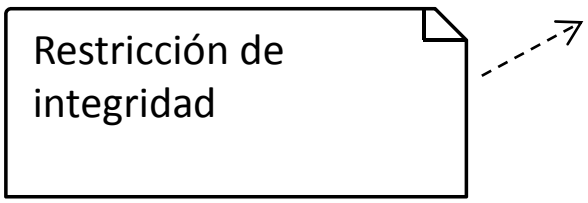
Estas propiedades no están expresadas en el diagrama.

4. Especificar restricciones de integridad

Incluir aquellas propiedades de la realidad que no hayan quedado expresadas en el diagrama de clases

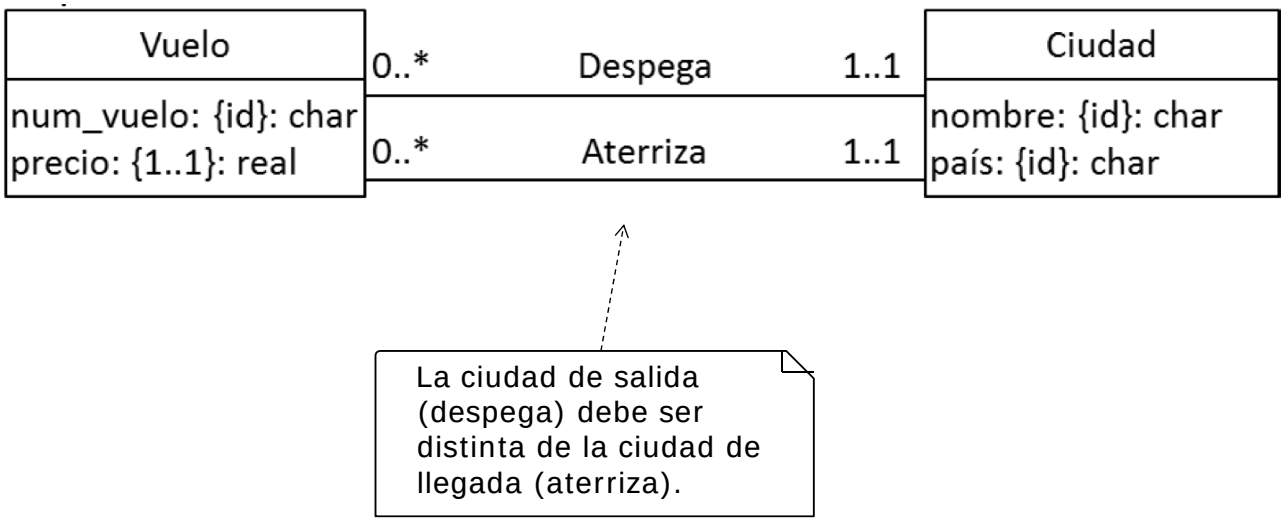
Usar elementos de anotación de los diagramas de UML:

Comentarios que se pueden aplicar para describir, clarificar y hacer observaciones sobre cualquier elemento del modelo.



4. Especificar restricciones de integridad

Ejemplo



5. Comprobaciones finales

- Nombres:
  - No se usa el mismo nombre para distintas clases, asociaciones roles
  - No se repiten nombres de atributos en una clase/subclase
- Identificadores
  - Toda clase tiene identificador (o es débil o es subclase)
  - Las clases asociación no tienen identificadores
  - Una clase no tiene atributos para referirse a otras clases: Se usan asociaciones

UD 4.1 Conceptos básicos de diseño. UML

2.3 Ejercicio 1: Restaurante

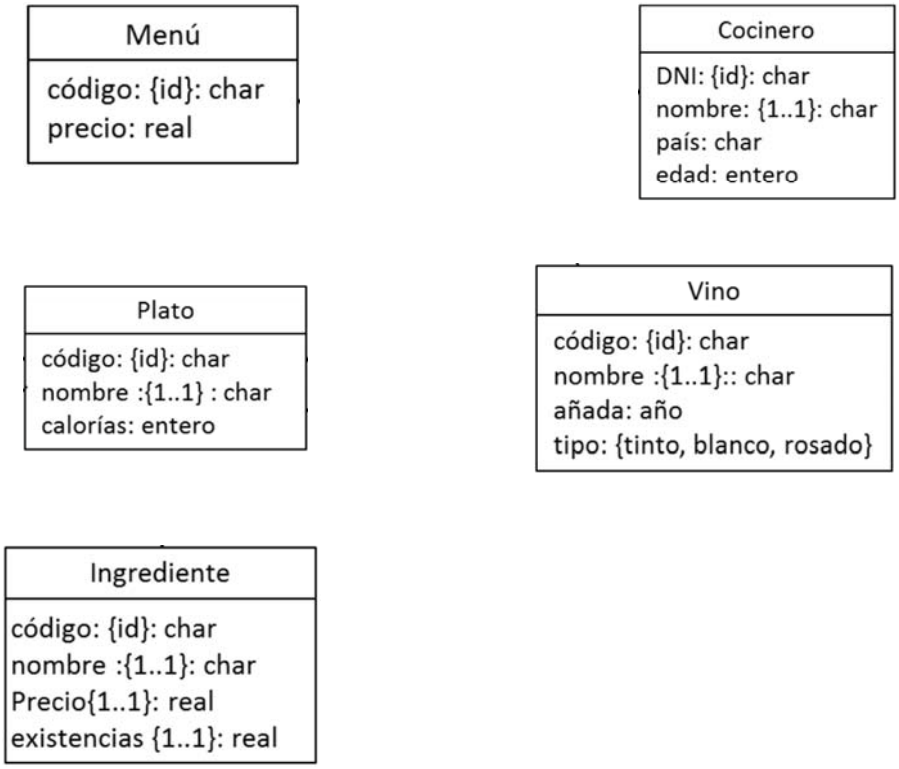
"Un restaurante desea que sus menús puedan ser consultados desde terminales y para ello ha decidido diseñar una base de datos relacional. En el restaurante se ofertan varios menús de los que se quiere saber el código, el precio y los platos que lo componen (al menos uno); cada plato puede haber sido diseñado como mucho por un cocinero del restaurante (que tiene DNI, nombre, país de origen y edad como datos de interés); de los platos se desea saber el código asignado al plato, el nombre y las calorías aproximadas; también puede interesar almacenar para algunos platos los ingredientes necesarios (indicando la cantidad) y el vino que se recomienda para su completo disfrute. De cada vino disponible en el restaurante se quiere conocer su código (interno al restaurante), el tipo, el nombre y la añada. Por último y para poder controlar la despensa, de cada ingrediente es interesante almacenar un código, el nombre, el precio en mercado y las existencias que quedan (obligatorio)."

2.3 Ejercicio 1: Restaurante

1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales

2.3 Ejercicio 1: Restaurante

"Un restaurante desea que sus menús puedan ser consultados desde terminales y para ello ha decidido diseñar una base de datos relacional. En el restaurante se ofertan varios menús de los que se quiere saber el código, el precio y los platos que lo componen (al menos uno); cada plato puede haber sido diseñado como mucho por un cocinero del restaurante (que tiene DNI, nombre, país de origen y edad como datos de interés); de los platos se desea saber el código asignado al plato, el nombre y las calorías aproximadas; también puede interesar almacenar para algunos platos los ingredientes necesarios (indicando la cantidad) y el vino que se recomienda para su completo disfrute. De cada vino disponible en el restaurante se quiere conocer su código (interno al restaurante), el tipo (blancos, tinto o rosado), el nombre y la añada. Por último y para poder controlar la despensa, de cada ingrediente es interesante almacenar un código, el nombre, el precio en mercado y las existencias que quedan (obligatorio)."

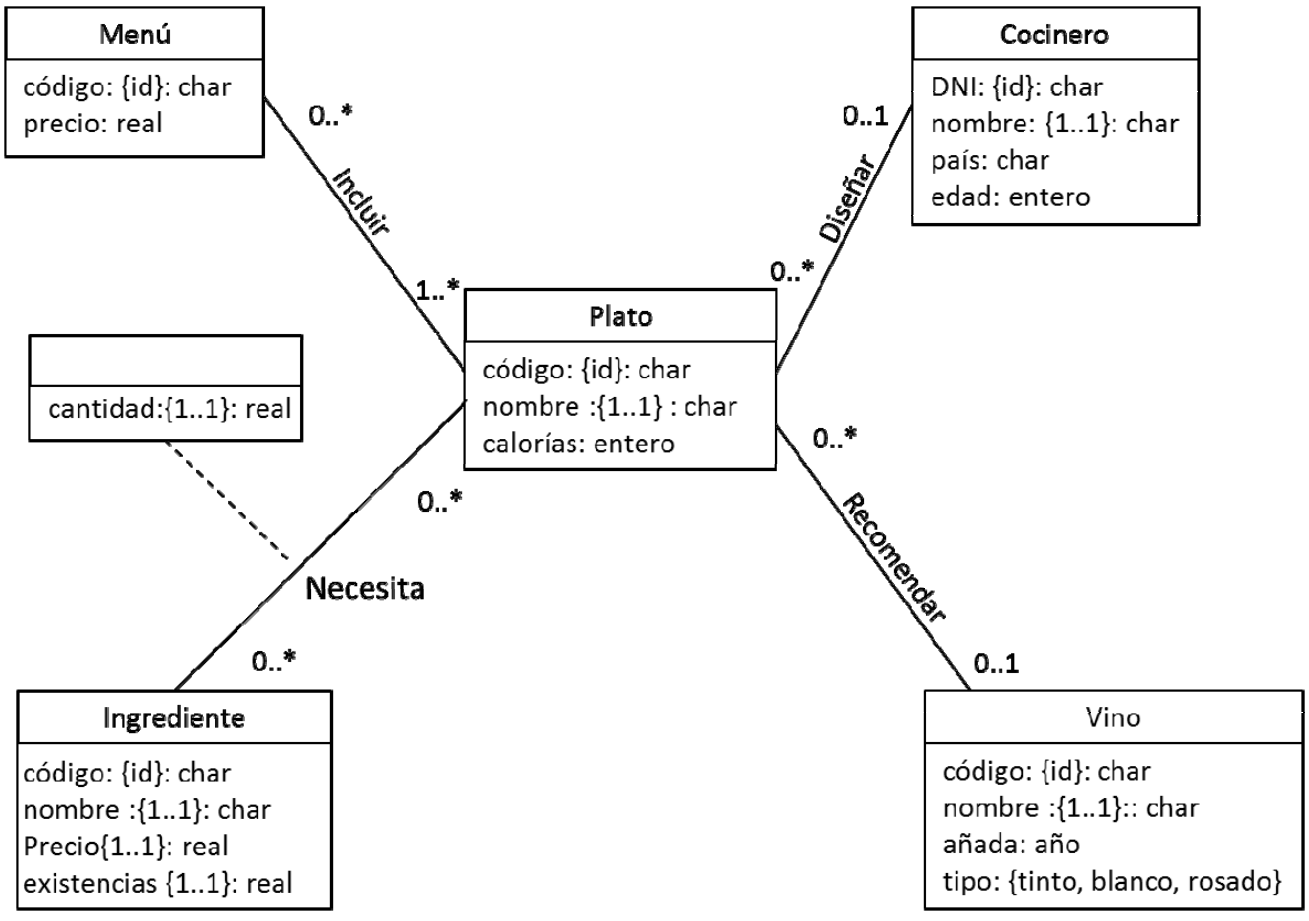


2.3 Ejercicio 1: Restaurante

1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales

2.3 Ejercicio 1: Restaurante

1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales



2.3 Ejercicio 1: Restaurante

1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales

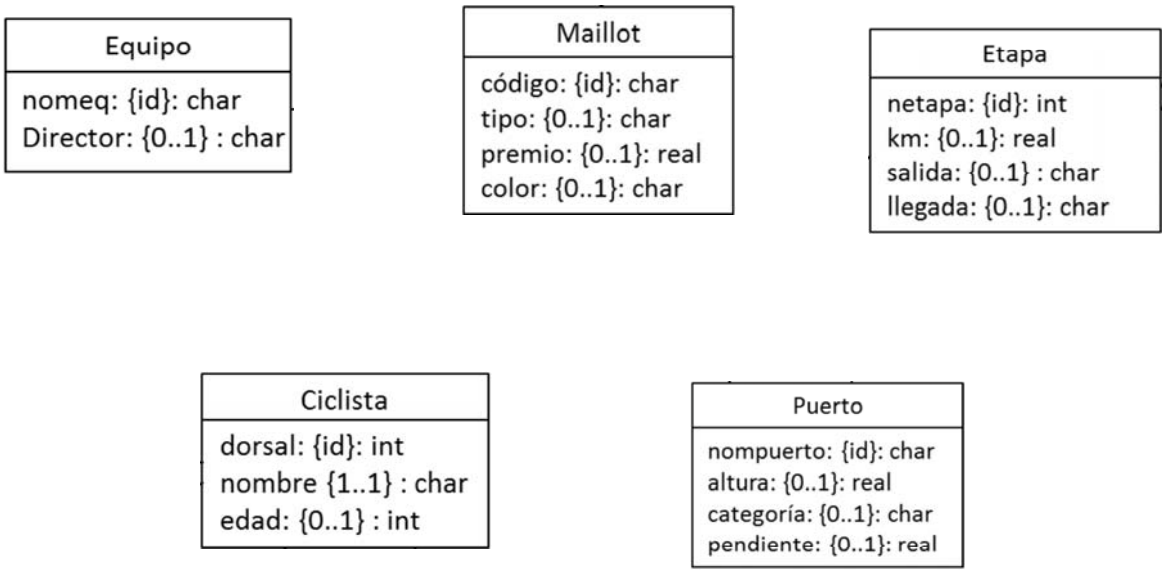
2.3 Ejercicio 1: Restaurante

1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales

UD 3.1 Conceptos básicos de diseño. UML

2.3 Ejercicio 2: Ciclismo

1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales

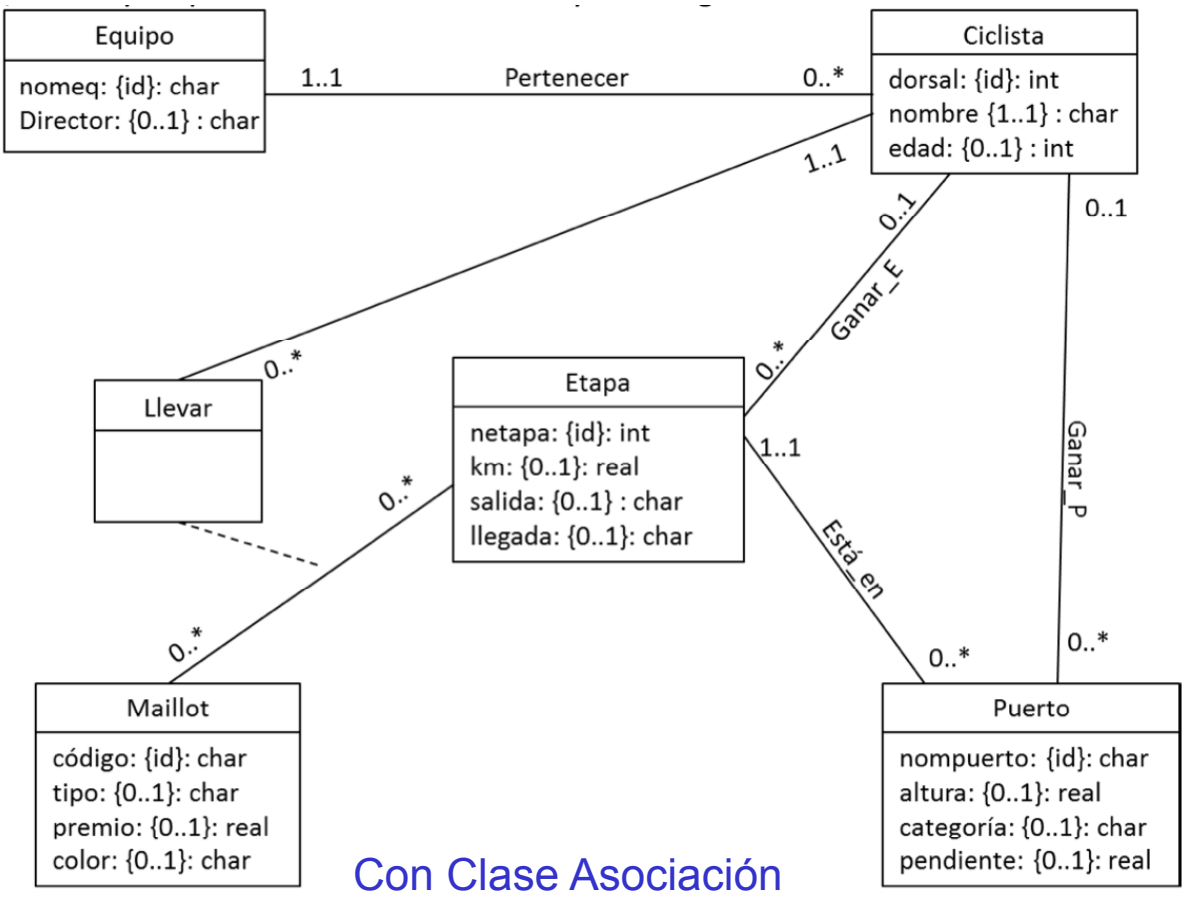


2.3 Ejercicio 2: Ciclismo

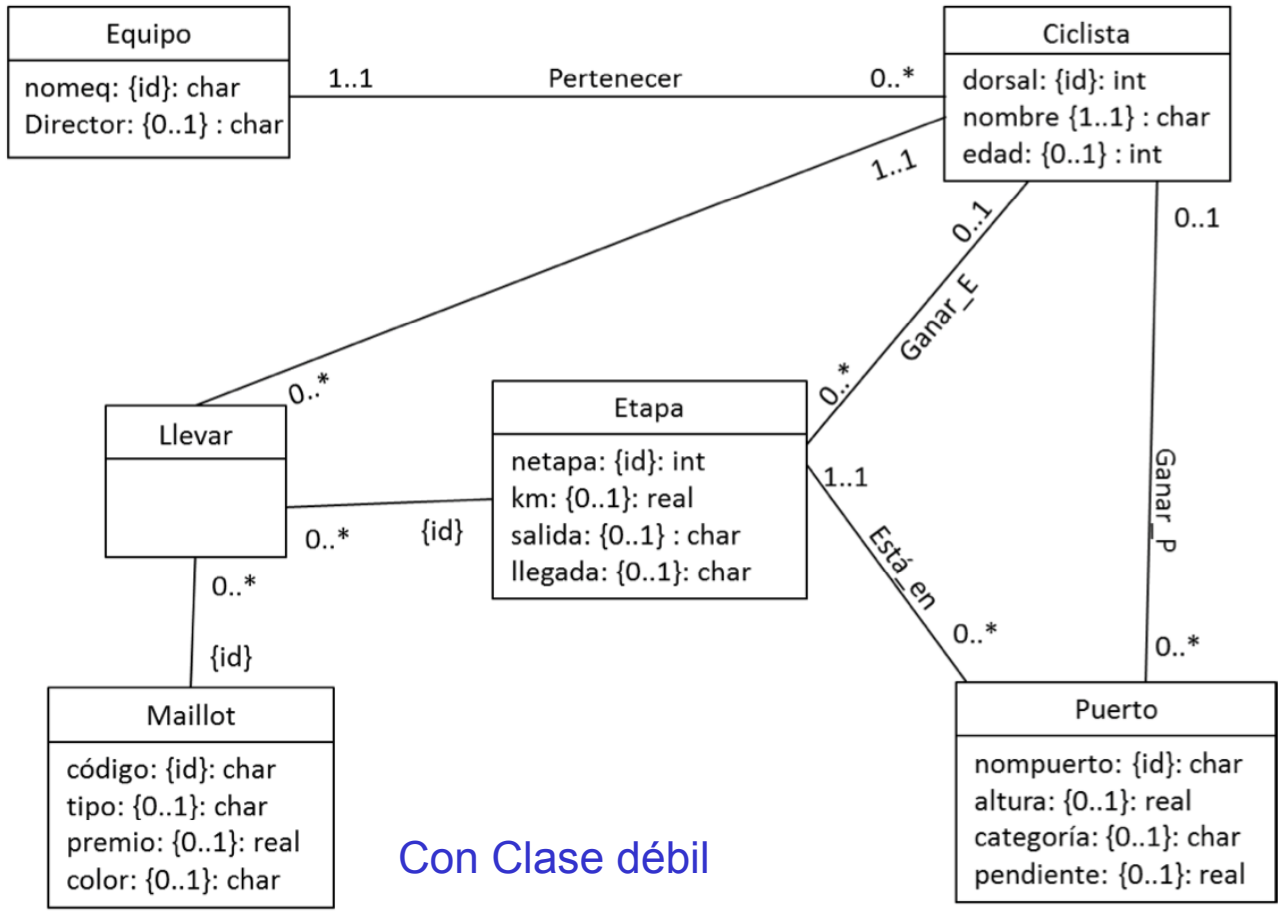
1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales

2.3 Ejercicio 2: Ciclismo

1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales



Con Clase Asociación



Con Clase débil

2.3 Ejercicio 2: Ciclismo

1. Identificar las clases con sus atributos,
2. Identificar generalizaciones/especializaciones,
3. Identificar asociaciones entre clases, y
4. Especificar restricciones de integridad.
5. Comprobaciones finales