Unidad 2

Diseño de bases de datos relacionales
Parte 2: Transformación del diagrama ER

Unidad 2. Diseño de bases de datos relacionales

Contenido:

2.4. Paso del diagrama E-R al modelo relacional

- Aunque los diagramas E-R y el modelo relacional se inventaron separadamente, ambos tienen una conexión muy estrecha que hace que los diagramas E-R puedan convertirse en bases de datos relacionales con facilidad.
- El paso de un modelo al otro se hace mediante unas **reglas**, aunque en determinados casos hay que saber saltárselas si existe una buena razón para ello.

- **El modelo relacional:**
  - La información se almacena en tablas.
    - Las filas se llaman registros o tuplas.
    - Las columnas se llaman campos.
  - Cada registro o tupla contiene la información correspondiente a una ocurrencia de una entidad o relación.
  - Por ejemplo, la entidad CLIENTES puede convertirse en una tabla como esta:

CódCliente	Nombre	Apellido1	Apellido2	Domicilio	Población	Teléfono
1	Arturo	Pérez	Montalbán	C/ Jón 13	Almería	139809239
2	Miguel	Asturias	Galeano	C/ Madera 20	Almería	294829384
3	Susana	Pérez	Aldecoa	C/ Perdiz 1	El Ejido	543111953
4	Estrella	Saavedra	López	Pza. Constución 2	Roquetas de Mar	237854344
5	Rocío	Molina	Jímenez	C/ Pez 5	Tabernas	783345546
...etc...	...	...	...	...	...	...

- # Paso del modelo ER al modelo relacional:
  - El modelo conceptual (diagrama ER) no puede representarse directamente en un SGBD relacional, porque éste carece de relaciones: sólo contiene tablas.
  - Para no perder información y construir una Base de Datos lo más optimizada posible, seguiremos siempre estos dos pasos:
    1. **Convertir el diagrama ER (modelo conceptual) en un conjunto de tablas (modelo lógico)**
       - En esta transformación no se debe perder ninguna información, es decir, el modelo resultante debe ser semánticamente equivalente al modelo original.
       - Este proceso se suele denominar paso a tablas y siempre implica introducir cierta redundancia en los datos.
    2. **Normalizar la Base de Datos**
       - Este proceso sirve para asegurar que la redundancia se mantiene a un nivel mínimo y que no se van a producir problemas durante la utilización de la Base de Datos.
  - El resto de esta parte del tema lo dedicaremos a aprender cómo dar estos dos pasos.

Conceptos previos:

- Tabla o relación
  - Conjunto de **celdas** dispuesto en filas (**registros**) y columnas (**campos**).
  - Cada **tabla** guarda la información correspondiente a una entidad o relación del modelo ER.
  - Cada **registro** guarda una ocurrencia de la entidad o relación.
  - Cada **campo** guarda un atributo de la entidad o relación.
  - Las celdas deben contener valores **atómicos**.
- Dominio
  - Es el conjunto de valores que puede tomar un campo.

Conceptos previos (cont.)

- Clave primaria
  - Es un campo o conjunto de campos cuyos valores determinan unívocamente el valor del resto de campos del registro.
  - Dicho de otra manera: el resto de campos del registro dependen funcionalmente de la clave primaria (véase “Normalización”)
- Clave candidata
  - Es un campo o conjunto de campos que *podrían* ser clave primaria, pero no lo son.
  - En algunas tablas encontraremos varias claves candidatas, pero sólo una puede ser elegida como clave primaria.
- Clave ajena, externa o foránea
  - Es un campo o conjunto de campos de una tabla que, sin ser clave en dicha tabla, sí que forman la clave primaria en alguna otra tabla de la base de datos.

Ejemplo: tabla ALUMNOS

Representación de tablas

- En el modelo relacional *sólo existen tablas*.
  - La distinción entre “entidad” y “relación” desaparece.
  - Tanto las “entidades” como las “relaciones” del modelo ER deben convertirse en tablas, campos, claves, etc... ¡sin perder información!

- Cómo representar la estructura de una tabla:
  - Por ahora sólo nos interesa la **estructura**, no los datos concretos.
  - Ejemplos:
    - ALUMNOS (**dni**#, nombre, apellido1, apellido2, domicilio, teléfono, fecha_nac, cód-grupo)
    - GRUPOS (**cód-grupo**#, denominación, aula)

2.4.a. Reglas de paso del diagrama ER al modelo relacional

Se trata de un conjunto de reglas que establecen como convertir al modelo relacional todos los elementos del diagrama ER:

- Entidades y sus atributos
- Relaciones 1:1
- Relaciones 1:N
- Relaciones N:N
- Relaciones de dependencia
- Relaciones reflexivas
- Relaciones N-arias
- Relaciones de jerarquía

Reglas de paso del diagrama ER al modelo relacional

- Las **entidades** del modelo ER se convierten en **tablas** del modelo relacional.
  - Los **atributos** de la entidad pasan a ser **campos** de la tabla.
  - Si algún atributo no es atómico, tendremos que arreglarlo durante la normalización.
- Ejemplo:

Entidad:

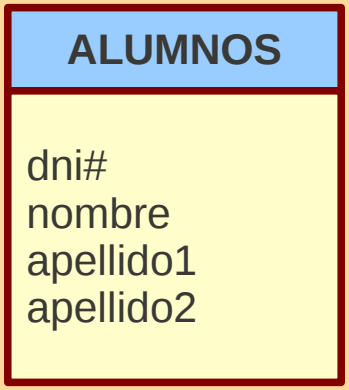


Tabla:

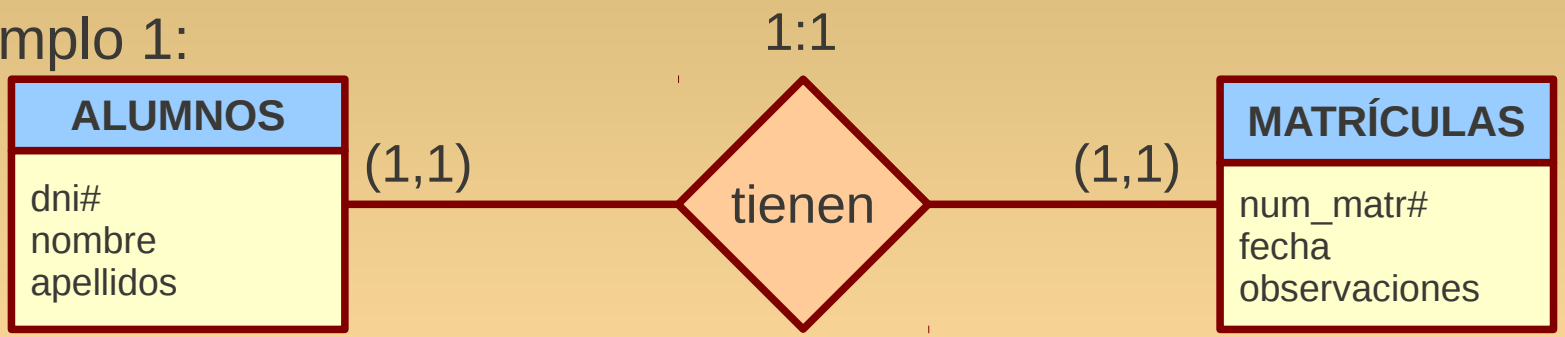
ALUMNOS (**dni**#, nombre, apellido1, apellido2)

- Las **relaciones 1:1** del modelo ER se convierten en **claves ajenas** del modelo relacional.
  - Si las dos entidades participan con **cardinalidad (1,1)**, nos llevamos la clave primaria de una entidad como clave ajena a la otra (en principio, da igual cuál elijamos)
  - Si una de las entidades participa con **cardinalidad (0,1)** y la otra con **cardinalidad (1,1)**, nos llevamos la clave de la parte (1,1) a la parte (0,1) para evitar valores nulos.

  (En estos dos casos, si la relación tiene atributos, nos los llevamos a la entidad donde hayamos migrado la clave)

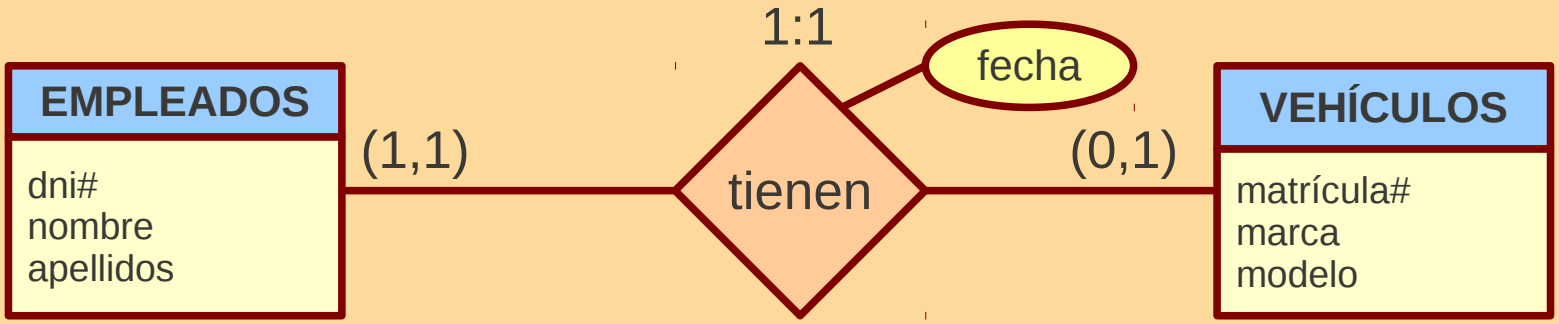
  - Si las dos entidades participan con **cardinalidad (0,1)**, crearemos una tabla nueva para la relación, que contendrá las claves de las dos entidades más los atributos propios de la relación, si los hay. La clave de la nueva tabla será una cualquiera de las claves ajenas.

- Ejemplo 1:



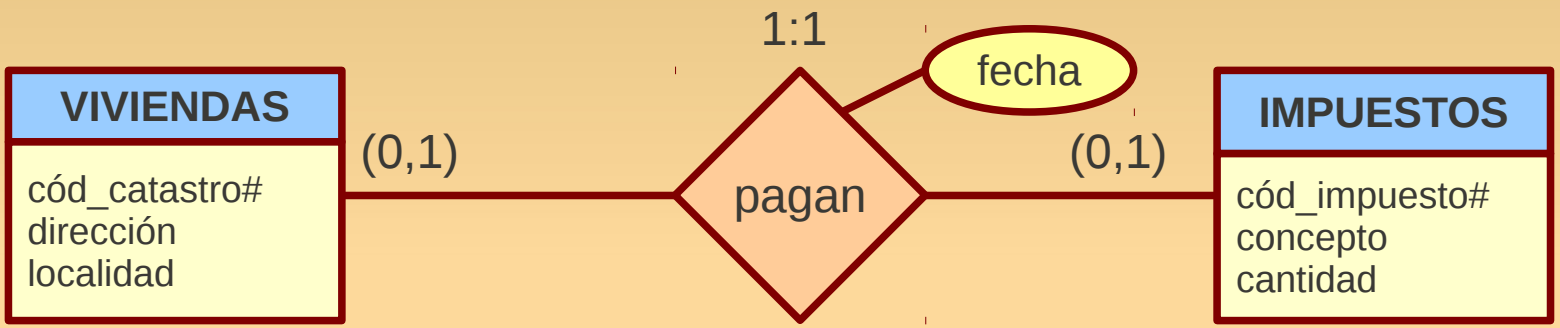
ALUMNOS (**dni**#, nombre, apellidos, num_matr)
MATRICULAS (**num_matr**#, fecha, observaciones)

- Ejemplo 2:



EMPLEADOS (**dni**#, nombre, apellidos)
VEHÍCULOS (**matrícula**#, marca, modelo, dni_empleado, fecha)

- Ejemplo 3:



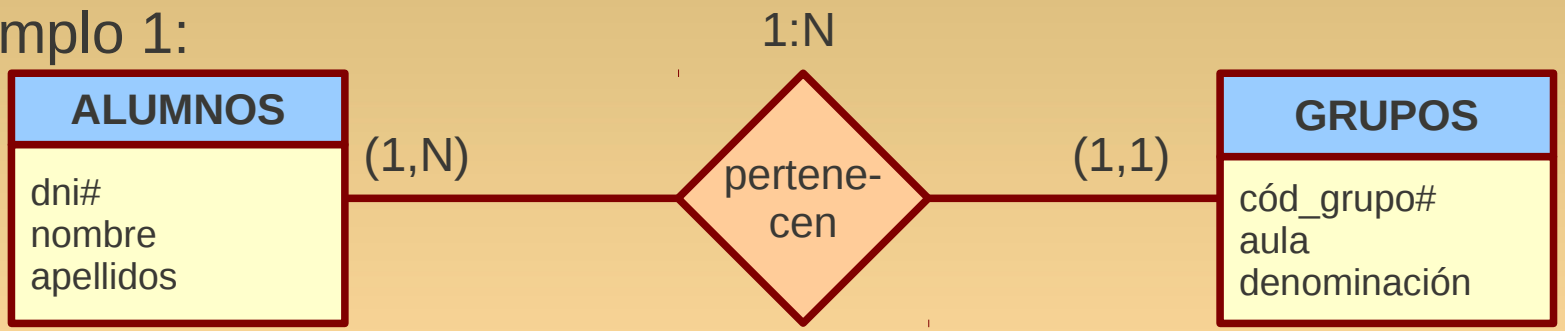
VIVIENDAS (**cód_catastro**#, dirección, localidad)
IMPUESTOS (**cód_impuesto**#, concepto, cantidad)
PAGAN (**cód_catastro**#, cód_impuesto, fecha)

- Las **relaciones 1:N** del modelo ER se convierten en **claves ajenas** del modelo relacional.
  - La clave de la entidad que participa con cardinalidad 1 pasa como clave ajena a la entidad que participa con cardinalidad N.
  - Si la relación tiene atributos, nos los llevamos a la entidad donde hayamos migrado la clave.
  - **Excepción**: si la entidad que participa con cardinalidad 1 lo hace con (0,1) en lugar de (1,1), *debe* crearse una tabla para la relación si queremos suprimir los valores nulos.

    Esta nueva tabla tendrá como campos las claves de las otras dos tablas y, como clave, la de la entidad con cardinalidad N.

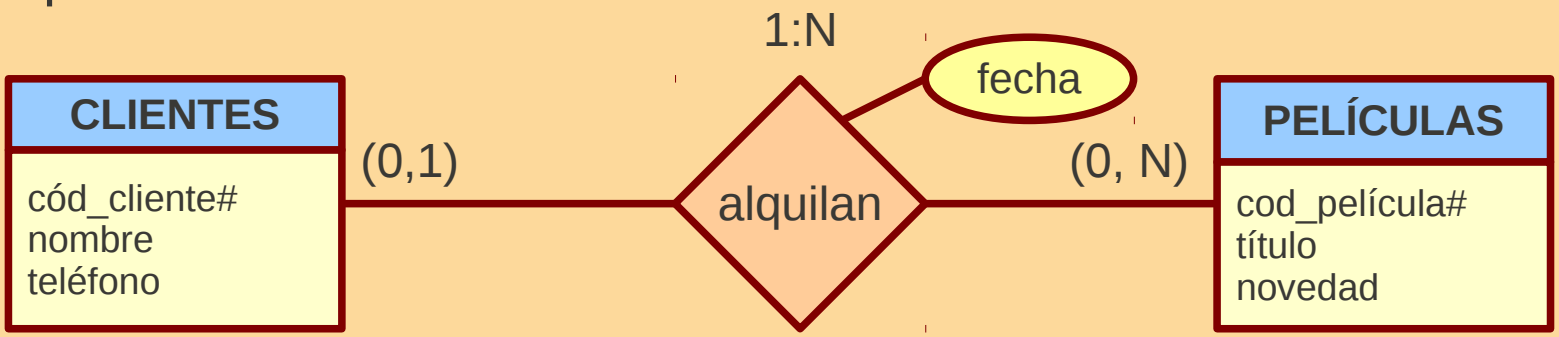
    (Si esto ocurre en la parte N, es decir, la cardinalidad es (0,N) en lugar de (1,N), el paso a tablas se hace normalmente)

- Ejemplo 1:



ALUMNOS (**dni**#, nombre, apellidos, cód_grupo)
GRUPOS (**cód_grupo**#, aula, denominación)

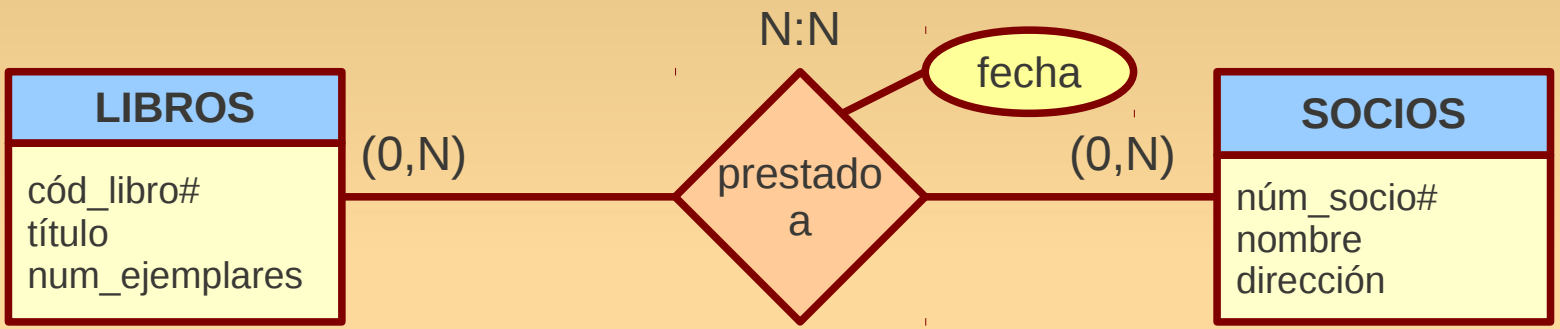
- Ejemplo 2:



CLIENTES (**cód_cliente**#, nombre, teléfono)
PELÍCULAS (**cód_película**#, título, novedad)
ALQUILAN (**cód_película**#, cód_cliente, fecha)

- Las **relaciones N:N** del modelo ER se convierten en **tablas** del modelo relacional.
  - Los campos de la nueva tabla serán las claves primarias de las dos entidades.
  - La clave primaria de la nueva tabla será el conjunto de claves ajenas.
  - Si hay atributos, se añaden a la nueva tabla como campos no clave.
  - Si alguna cardinalidad mínima es 0, la relación se resuelve exactamente igual.
  - Debemos poner siempre al principio los campos clave, ordenados según los que más se vayan a utilizar para realizar consultas para mejorar la eficiencia
    (Sólo si podemos prever qué tipo de consultas van a ser más frecuentes, claro)

- Ejemplo:



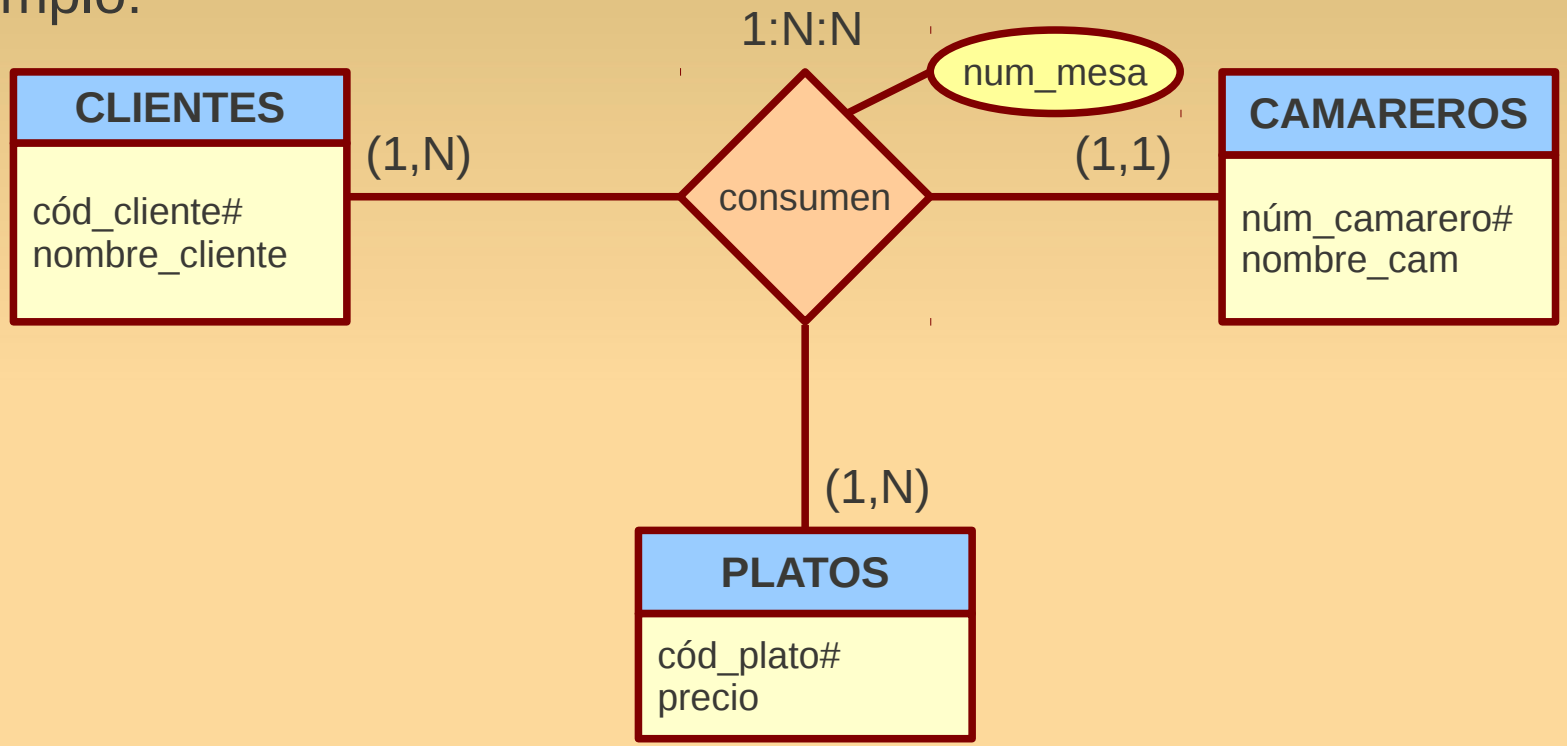
LIBROS (**cód_libro**#, título, num_ejemplares)
SOCIOS (**núm_socio**#, nombre, dirección)
PRESTADO_A (**cód_libro**#, **num_socio**#, fecha)

- La relaciones de **dependencia** se resuelven según su **cardinalidad**, siguiendo las mismas reglas que el resto de relaciones.
  - Las dependencias en identificación suelen tener cardinalidad 1:1 o 1:N, por lo que no suelen generar tabla.
  - Excepción: si la dependencia es en **identificación**, la clave de la entidad fuerte debe expandirse a la entidad débil y formar parte de su clave primaria.

    Además, es aconsejable que dicha clave se coloque *en primer lugar* en la lista de campos de la entidad débil.

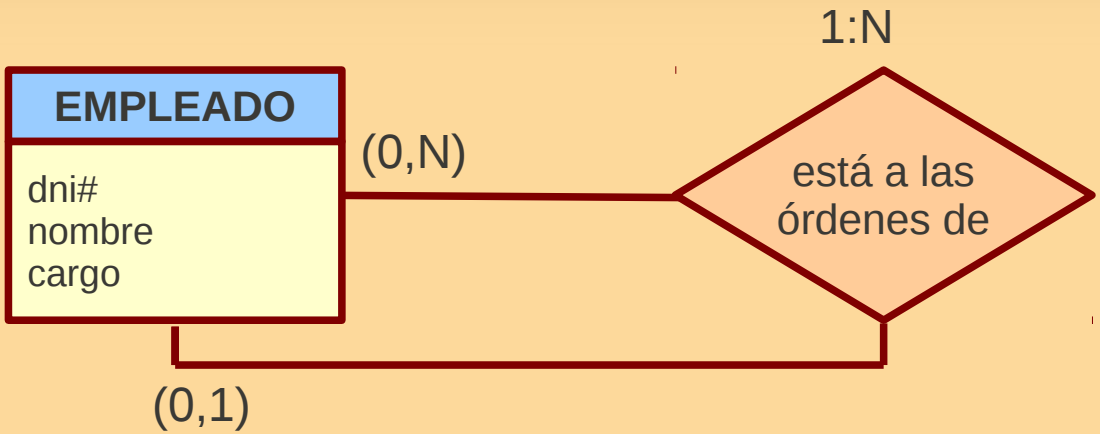
- Las **relaciones N-arias** se resuelven según las cardinalidades de las entidades participantes, siguiendo las reglas vistas anteriormente.
  - Normalmente generan una nueva tabla.
  - Los campos de la nueva tabla serán las claves de las entidades que participan con cardinalidad N en la relación, más los atributos propios de la relación.
  - La clave de la nueva tabla será el conjunto de claves ajenas.
  - La clave de las entidades que participen con cardinalidad 1, no tienen por qué formar parte de la clave de la nueva tabla.

- Ejemplo:



CLIENTES (**cód_cliente**#, nombre_cliente)
CAMAREROS (**núm_camarero**#, nombre_cam)
PLATOS (**cód_plato**#, precio)
CONSUMEN (**cód_cliente**#, **cód_plato**#, núm_camarero, num_mesa)

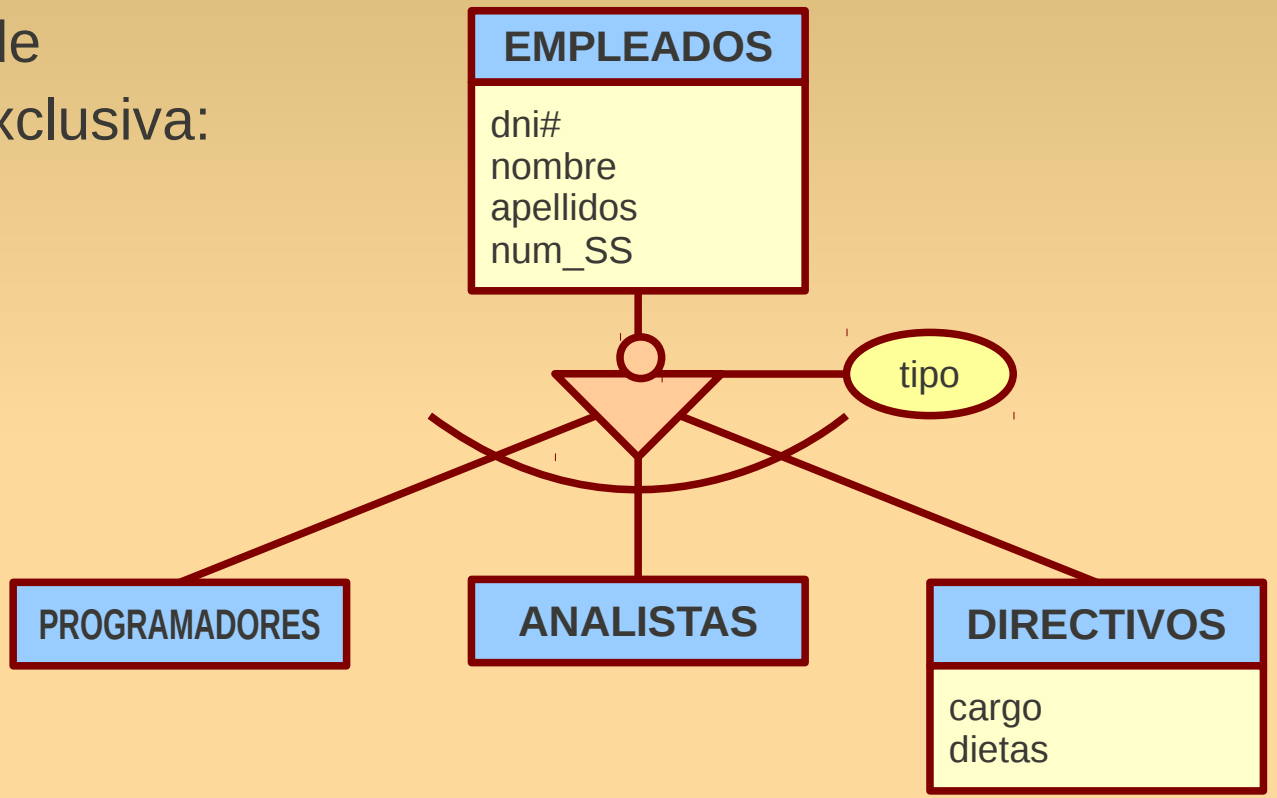
- La relaciones **reflexivas** se resuelven según su **cardinalidad**, siguiendo las mismas reglas que el resto de relaciones.



EMPLEADO (**dni**#, nombre, cargo)
ESTÁ_A_LAS_ÓRDENES_DE (**dni_empleado**#, dni_jefe, fecha)

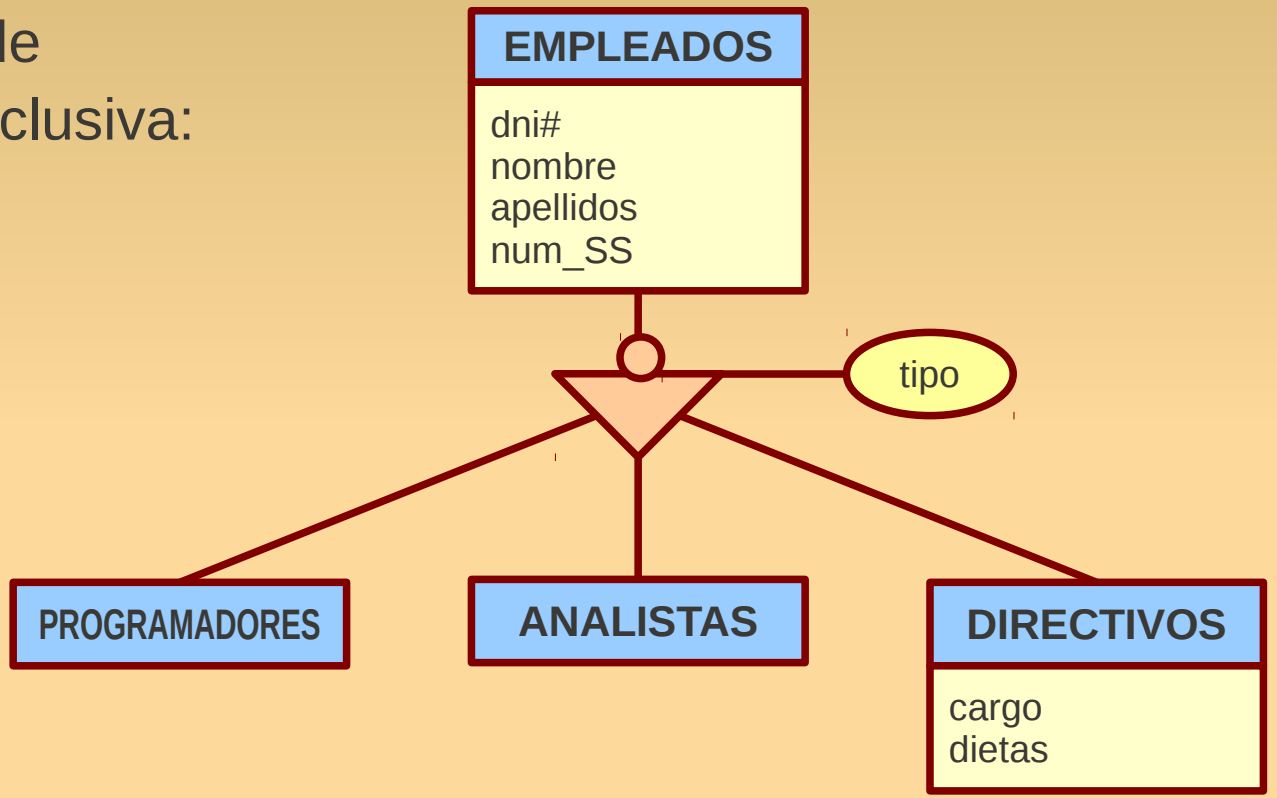
- Las **relaciones jerárquicas** no tienen un modo fijo de resolverse, sino que dependen de cada caso.
- Como normas aproximadas podemos dar éstas:
  - Crear una tabla para la entidad **supertipo**, a no ser que tenga pocos (o ningún) atributo.
  - Crear una tabla para cada entidad **subtipo** que tenga atributos. Si no tiene clave propia, la hereda del supertipo. Si el subtipo no tiene atributos, desaparece.
  - Si la relación es **exclusiva**, el atributo de la relación se añade a la tabla del supertipo, excepto si todos los subtipos han generado tabla.
  - Si la relación es inclusiva, se pueden tomar dos caminos:
    - Crear una tabla para el supertipo y para cada subtipo, tengan o no atributos. No añadir el atributo de la relación al supertipo.
    - Crear una tabla para la relación que contenga como clave la clave del supertipo y su propio atributo. Crear tablas sólo para los subtipos que tengan atributos.

- Ejemplo de relación exclusiva:



EMPLEADOS (**dni**#, nombre, apellidos, num_SS, tipo)
DIRECTIVOS (**dni**#, cargo, dietas)

- Ejemplo de relación inclusiva:



OPCIÓN 1
EMPLEADOS (**dni**#, nombre, apellidos, num_SS)
PROGRAMADORES (**dni**#)
ANALISTAS (**dni**#)
DIRECTIVOS (**dni**#, cargo, dietas)

OPCIÓN 2
EMPLEADOS (**dni**#, nombre, apellidos, num_SS)
DIRECTIVOS (**dni**#, cargo, dietas)
ES_UN (**dni**#, **tipo**#)

2.5. Normalización

- **Objetivo** de la normalización:
  - Prevenir los problemas de **redundancia** en los datos y las anomalías en modificaciones, inserciones y borrados.
  - ¡La redundancia es la mayor enemiga de las bases de datos!
    - Puede producir incoherencias.
    - Y las incoherencias hacen que la base de datos deje de ser útil.
- **Definición** de la normalización:
  - *"Conjunto de técnicas que transforman un modelo de bases de datos en otro equivalente donde las redundancias de información están minimizadas"*
- Para lograrlo, el nuevo modelo debe cumplir determinadas restricciones conocidas como **formas normales**.
  - NOTA: Detrás de las formas normales y de la normalización existen una extensa teoría matemática que la justifica, en cuyos detalles no vamos a entrar.

- **Dependencias**:
  - Los datos de la base de datos, por su propia naturaleza, tienen una serie de **dependencias** entre sí.
  - Por ejemplo: el “nombre de cliente” depende del “dni de cliente” porque cada dni tiene asociado un (y sólo un) nombre.
  - Puede haber dependencias de varios tipos:
    - Funcionales
    - Multivaluadas
    - De combinación

- Para nuestro propósito nos basta con las **dependencias funcionales**

- **Dependencias funcionales**:
  - Un **descriptor** es un campo o un conjunto de campos de una tabla.
  - Sean X e Y dos descriptores de una tabla cualquiera.
  - Se dice que Y depende funcionalmente de X si y sólo si cada valor de X tiene asociado en todo momento un único valor de Y.
  - Se representa así:

    $X \rightarrow Y$

  - Por ejemplo, en la tabla CLIENTES:

    dni_cliente $\rightarrow$ nombre_cliente
    - *Esto significa que cada valor de "dni_cliente" tiene asociado un único valor de "nombre_cliente"*
    - *Por lo tanto, diremos que "nombre_cliente" depende funcionalmente de "dni_cliente"*

- **Otro ejemplo de dependecia funcional**:
  - La tabla FALTAS_ASISTENCIA del sistema “Instituto” tenía esta forma:

FALTAS ASISTENCIA
cod_alumno#
fecha#
hora#
tipo_falta
observaciones

  - Encontramos esta dependencia funcional

    (cod_alumno, fecha, hora) $\rightarrow$ tipo_falta
    - *Esto significa que cada valor del descriptor (cod_alumno, fecha, hora) tiene asociado un único valor de “tipo_falta”*
  - Pero no existe esta otra dependencia

    cod_alumno $\not\rightarrow$ tipo_falta
    - *Es decir, cada valor de “cod_alumno” no tiene asociado un único valor de “tipo_falta”, sino varios.*

- # Las formas normales
  - Una **forma normal** es un *conjunto de restricciones que deben cumplir los descriptores de una tabla respecto de las dependencias funcionales que tienen entre ellos*.
  - Si todos los descriptores cumplen las restricciones de la forma normal X, se dice que la tabla *está en la forma normal X.*
  - Existen varias formas normales, cada vez más restrictivas:
    - **1FN**: primera forma normal.
    - **2FN**: segunda forma normal.
    - **3FN**: tercera forma normal.
    - **FNBC**: forma normal de Boyce-Codd.
    - **4FN**: cuarta forma normal.
    - **5FN**: quinta forma normal.
  - Nuestro objetivo será alcanzar la FNBC (es decir, transformar nuestras tablas hasta que estén en FNBC)

Concepto intuitivo de forma normal (mediante un ejemplo)

- La 1FN establece la restricción de que “no puede haber grupos de datos compuestos en un campo”
- He aquí un ejemplo de tabla que *no está en 1FN*:

COD_ALUMNO	NOMBRE	ASIGNATURAS
1032	Antonio Suárez Pérez	Matemáticas, Física
5050	Sonia Robles Jiménez	Matemáticas, Inglés
0010	Marta Arévalo Collado	Lengua, Latín, Filosofía
0728	Juan Dorado Rubio	Lengua, Inglés, Física

- Esta tabla no está en 1FN porque el campo ASIGNATURAS contiene valores múltiples: *no es atómico*.
- Esta tabla no se puede implementar en un SGBD relacional sin pasarla antes, al menos, a 1FN.
- ¿Cómo se puede modificar para que esté en 1FN?

Concepto intuitivo de forma normal

- **Posibilidad 1** para poner la tabla del ejemplo en 1FN

COD_ALUMNO	NOMBRE	ASIGNATURAS
1032	Antonio Suárez Pérez	Matemáticas
1032	Antonio Suárez Pérez	Física
5050	Sonia Robles Jiménez	Matemáticas
5050	Sonia Robles Jiménez	Inglés
0010	Marta Arévalo Collado	Lengua
0010	Marta Arévalo Collado	Latín
0010	Marta Arévalo Collado	Filosofía
0728	Juan Dorado Rubio	Lengua
0728	Juan Dorado Rubio	Inglés
0728	Juan Dorado Rubio	Física

Esta tabla ya está en 1FN, pero tiene **problemas de redundancia**

- **Posibilidad 2** para poner la tabla del ejemplo en 1FN

COD_ALUMNO	NOMBRE
1032	Antonio Suárez Pérez
5050	Sonia Robles Jiménez
0010	Marta Arévalo Collado
0728	Juan Dorado Rubio

COD_ALUMNO	ASIGNATURAS
1032	Matemáticas
1032	Física
5050	Matemáticas
5050	Inglés
0010	Lengua
0010	Latín
0010	Filosofía
0728	Lengua
0728	Inglés
0728	Física

Las dos tablas están en 1FN. **Se ha reducido la redundancia** respecto de la solución anterior, pero **ha aumentado el número de tablas**.

- **Posibilidad 3** para poner la tabla del ejemplo en 1FN

COD_ALUMNO	NOMBRE	ASIG1	ASIG2	ASIG3
1032	Antonio Suárez Pérez	Matemáticas	Física	
5050	Sonia Robles Jiménez	Matemáticas	Inglés	
0010	Marta Arévalo Collado	Lengua	Latín	Filosofía
0728	Juan Dorado Rubio	Lengua	Inglés	Física

Esta tabla está en 1FN, pero plantea nuevos problemas:

- Aparecen campos con valores vacíos (**nulos**)
- Es **poco flexible**: ¿qué ocurre si necesitamos ampliar el número de asignaturas?

- Pregunta: ¿Cuál de las tres soluciones es mejor?
- Respuesta:
  - Será preferible la que esté en **FNBC**.
  - Si no hay ninguna en FNBC, elegiremos, por orden de preferencia, la que esté en 3FN, 2FN o 1FN.

- Por lo tanto, tenemos que conocer en qué consiste cada una de las formas normales para poder transformar nuestras tablas y dejarlas en un estado óptimo.

Primera forma normal (1NF)

- Una tabla está en **1NF** si y sólo si **todos sus campos son atómicos**.
- El primer paso en la normalización de una tabla debe ser, por tanto, eliminar los campos que contengan grupos de datos.
- El modelo relacional cumple esta restricción por definición: en una celda de una tabla no pueden coexistir múltiples valores.
- **Ejemplo**:

  EMPLEADOS (cód_empleado#, nombre, apellido1, apellido2, idioma)

  - Si un empleado habla varios idiomas, en el campo “idioma” habrá que guardar múltiples valores.
  - Por lo tanto, esta tabla no está en 1NF.
  - Solución: dividir la tabla en dos
    - EMPLEADOS (cód_empleado#, nombre, apellido1, apellido2)
    - IDIOMAS (cód_empleado#, idioma#)

Segunda forma normal (2NF)

- **Dependencia funcional completa** (o plena):
  - Dado un descriptor X compuesto (X1, X2), se dice que otro descriptor Y tiene *dependencia funcional completa* respecto de X si depende funcionalmente de X y no lo hace de ningún subconjunto de X
  - (Es decir, Y no depende funcionalmente ni de X1 ni de X2)

- Una tabla está en **2NF** si y sólo si:
  - Está en 1NF
  - Todos los campos no clave tienen una *dependencia funcional completa* respecto de las claves candidatas.

- Las dependencias no completas con la clave son una enorme fuente de errores en la implementación.

- **Ejemplo 1:**

  VENTAS (cód_pieza#, cód_almacén#, cantidad, dirección_almacén)

  - Esta tabla no está en 2NF porque:
    - (cód_pieza, cód_almacén) $\rightarrow$ dirección_almacén
    - cód_almacén $\rightarrow$ dirección_almacén
  - Solución: dividir la tabla en dos
    - VENTAS (cód_pieza#, cód_almacén#, cantidad)
    - ALMACENES (cód_almacén#, dirección_almacén)

- **Ejemplo 2:**

  PROYECTOS (cód_empleado#, cód_proyecto#, función_empleado)

  - Esta tabla sí está en 2NF porque:
    - (cód_empleado, cód_proyecto) $\rightarrow$ función_empleado
    - cód_empleado $\not\rightarrow$ función_empleado
    - cód_proyecto $\not\rightarrow$ función_empleado

(Suponiendo que un empleado pueda trabajar a la vez en varios proyectos, desempeñando una función diferente en cada uno)

Tercera forma normal (3NF)

- **Dependencia funcional transitiva**:
  - Dada una tabla con tres descriptores, X, Y y Z, verificándose las siguientes dependencias funcionales:
    - $X \rightarrow Y$
    - $Y \rightarrow Z$
    - $Y \not\rightarrow X$

    ...se dice que Z tiene una dependencia funcional transitiva con X a través de Y, y se representa:
    - $X \rightarrow Z$
- Una tabla está en **3NF** si y sólo si:
  - Está en 2NF.
  - Los campos no clave dependen **no** transitivamente de la clave.
- Las dependencias transitivas con la clave pueden provocar muchos errores en la implementación.

- **Ejemplo:**

  VUELOS (num_vuelo#, fecha, origen, destino, cód_compañía, nombre_compañía)

  - Esta tabla no está en 3NF porque:
    - num_vuelo $\rightarrow$ cód_compañía
    - cód_compañía $\rightarrow$ nombre_compañía
    - cód_compañía $\not\rightarrow$ num_vuelo
    - num_vuelo $\rightarrow$ nombre_compañía

      (esta es la dependencia transitiva)

  - La solución vuelve a ser dividir la tabla en dos:
    - VUELOS (num_vuelo#, fecha, origen, destino, cód_compañía)
    - COMPAÑÍAS (cód_compañía#, nombre_compañía)

Forma normal de Boyce-Codd (BCNF)

- **Determinante**:
  - Se llama determinante a cualquier conjunto de campos del cual otro campo depende funcionalmente de forma completa.

- Una tabla está en **BCNF** si y sólo si todo determinante es una clave candidata.

- **Ejemplo:**

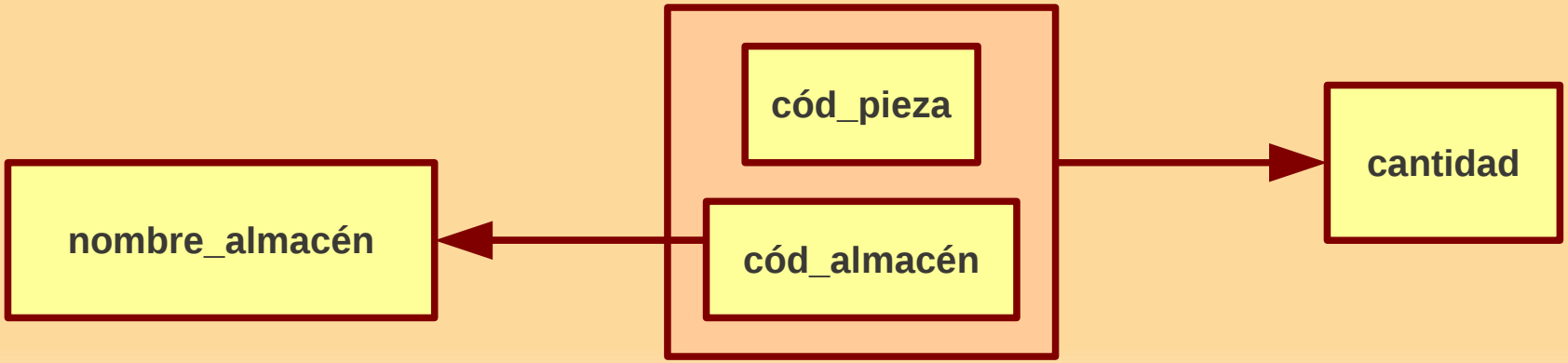
  VENTAS (cód_pieza#, cód_almacén#, nombre_almacén, cantidad)

  - Supondremos que los almacenes se pueden identificar unívocamente tanto por el código como por el nombre.
  - Hay una clave primaria (cód_pieza, cód_almacén) y una clave candidata (cód_pieza, nombre_almacén)
  - La tabla no está en BCFN porque existen cuatro determinantes:
    - (cód_pieza, cód_almacén) $\rightarrow$ cantidad
    - (cód_pieza, nombre_almacén) $\rightarrow$ cantidad
    - cód_almacén $\rightarrow$ nombre_almacén
    - nombre_almacén $\rightarrow$ cód_almacén
  - La solución es, como siempre, dividir la tabla en dos:
    - VENTAS (cód_venta#, cód_almacén#, cantidad)
    - ALMACENES (cód_almacén#, nombre_almacén#)

Diagramas de dependencias

- Son una herramienta útil para **determinar las dependencias funcionales** dentro de una tabla (paso previo a la normalización)
- Consisten en una **representación gráfica** de las dependencias.
- Ejemplo: supongamos esta tabla con las siguientes dependencias:

  VENTAS (cód_pieza#, cód_almacén#, nombre_almacén, cantidad)
  - (cód_pieza, cód_almacén) → cantidad
  - cód_almacén → nombre_almacén
- El diagrama de dependencias permite observar las dependencias con más facilidad (sobre todo si son muchas y complejas):



- **Conclusión:**
  - Las técnicas de normalización son sólo **guías**.
  - Una base de datos normalizada contendrá menos redundancias y provocará menos anomalías en sus actualizaciones que otra que no lo esté.
- **Inconvenientes de la normalización**:
  - Existen otras dependencias, además de las funcionales, que pueden presentarse en la práctica y dar al traste con nuestra normalización.
  - La descomposición de una tabla en varias puede no ser única. La normalización no proporciona criterios para decidir cuál de las posibles divisiones es más conveniente.
  - No todas las redundancias pueden ser eliminadas mediante normalización.
  - En ocasiones, la base de datos se complica en exceso si insistimos en llegar hasta la BCFN.