INDICE DE CONTENIDOS

1. Introducción

El modelo de datos relacional fue desarrollado por **Edgar Frank Codd** para IBM durante los años setenta. Está basado en dos ramas de las matemáticas: la teoría de conjuntos y la lógica de predicados de primer orden. Su base matemática hace que el modelo sea predecible, fiable y seguro.

Objetivos del modelo de **Codd**:

- **Independencia física de los datos**, el modo de almacenamiento de los datos no debe influir en su manipulación lógica.
- **Independencia lógica de los datos**, los cambios que se realicen en los objetos de la base de datos no deben repercutir en los programas y usuarios que acceden a la misma.
- **Flexibilidad**, para presentar a los usuarios los datos de la forma más adecuada a la aplicación que utilicen.
- **Uniformidad,** en la presentación de las estructuras lógicas de los datos, que son tablas, lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- **Sencillez**, pues las características anteriores, así como unos lenguajes de usuario sencillos hacen que este modelo sea fácil de comprender y utilizar por el usuario.

2. Elementos del Modelo Relacional.

2.1.La relación.

La ***relación*** o tabla es el elemento básico del modelo relacional.

- Una relación tiene un **nombre**, un **conjunto de atributos** que representa sus propiedades y un **conjunto de registros** (filas de la tabla o tuplas). Cada casilla (celda) contiene el valor de cada atributo para cada registro de la relación (el dato, datos).

- Podemos ver una tabla como una hoja Excel donde **los atributos** son **las columnas de la tabla** y **los registros son las filas** y **el dato** es el **contenido de cada celda**.

Esto es una anotación

MR

PROFESOR

Cód_prof	*Nombre*	*DNI*	*Dirección*	*Teléfono*	*Materia*
00001	Juan	12223433	Ríos Rosas, 23	670123123	Ing. Software
00002	Coral	54656754	Alacena, 8	567983456	Bases de datos
00003	Belén	53567523	Getafe, 4	692678S4	Orientación objetos
00004	Goyo	97856757	Pez, 102	679345763	Sistemas operativos
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
03568	Roberto	34534522	Fundación, 10	639456239	Redes

CLAVE PRIMARIA

PROFESOR (*Cód_prof, Nombre, DNI, dirección, Teléfono, Materia*)

Se define Grado de una tabla o relación como el número de columnas que tiene y definimos cardinalidad de una tabla como el número de filas que tiene la tabla (número de registros)

El modelo relacional impone una serie de restricciones inherentes al concepto de relación:

- **En una tabla no puede haber dos registros iguales** (es obligatorio que haya una clave primaria que diferencie a todos los registros y que haga de **identificador único**).
- **Ningún atributo que forme la clave primaria puede tener un valor nulo.**

- El **orden de los registros** de la tabla **no es relevante**.
- **Cada atributo** solamente puede tomar **un único valor. No pueden existir** por tanto **atributos multivaluados.**

2.2. Atributos y dominios

Los atributos son las características que describen la entidad o la relación.

El dominio es el conjunto de todos los posibles valores para una o más columnas de una tabla relacional. El dominio debe definirse previamente para poder establecer restricciones respecto a valores que puede tomar un campo.

Pueden distinguirse dos tipos de Dominios:

- **Generales o Continuos**: Contienen todos los valores posibles entre un máximo y un mínimo definidos o en un ámbito.

  Por ejemplo, el atributo cod_empleado, su dominio sería un número entero positivo de 4 dígitos. Sería cualquier número entre el valor 0 y el 9999.
- **Restringidos o Discretos**: Contiene ciertos valores predeterminados o contenidos entre un máximo y un mínimo. Por ejemplo, el estado civil, será un carácter que puede tomar los valores S, V, C o D, el sexo H o M.

2.3.Restricciones semánticas.

Representan la semántica del mundo real, Condiciones que deben cumplir los datos para su correcto almacenamiento. Hay varios tipos:

- La restricción de unicidad (**UNIQUE**), permite definir claves alternativas, los valores de los atributos no pueden repetirse. Declararemos estas restricciones con la abreviatura **UK**.
- La restricción de clave primaria (**PRIMARY KEY**) identifica unívocamente cada registro de la tabla. Puede estar formado por varios atributos. Declararemos esta restricción con la abreviatura **PK**.

  Ejemplo:

  ALUMNOS (cod_alumno, nombre, apellidos, DNI, calle, ciudad, provincia, teléfono)

  PK: cod_alumno.

  UK: DNI.

  ***Nota**: no puede haber en la tabla de alumnos dos DNIs iguales.*

- La restricción de obligatoriedad (**NOT NULL**), permite declarar si uno o varios atributos no pueden tomar valores nulos. Por defecto cuando declaramos la restricción de Primary Key sobre un atributo este no puede ser nulo. No tendríamos que declarar nada más. Ocurre lo mismo cuando declaramos una restricción de tipo UNIQUE. Si no se indica nada, el atributo es por defecto NULL por lo que sería opcional.

Ejemplo: en la tabla anterior, convertir el teléfono en campo obligatorio.

ALUMNOS (cod_alumno, nombre, apellidos, DNI, calle, ciudad, provincia, teléfono)

PK: cod_alumno.

UK: DNI.

NN: teléfono.

(*) Por defecto, todos los atributos de una tabla son NULOS (opcionales) excepto los que forman parte de la PK y los que son UK. Si queremos que un atributo sea obligatorio, debemos declararlo NN.

- Integridad referencial o restricción de clave ajena (**FOREIGN KEY, FK**), se utiliza para enlazar relaciones, mediante claves ajenas, de una base de datos. La integridad referencial indica que los valores de la clave ajena en la relación hijo se corresponden con los de la clave primaria en la relación padre.

Ejemplo: *tenemos dos tablas, ALUMNOS y MATRÍCULAS que relacionan los alumnos y la información general de la matrícula del alumno:*

ALUMNOS (cod_alumno, nombre, apellidos, DNI, calle, ciudad, provincia, teléfono)

PK (Primay Key): cod_alumno.

UK (Unique): DNI.

NOT NULL: teléfono.

MATRÍCULAS(cod_matricula, ciclo, curso, num_módulos, observaciones)

PK (Primay Key): cod_matricula.

NN(NOT NULL): ciclo, curso, num_módulos.

*Para **relacionar** ALUMNOS y MATRÍCULAS y poder **saber qué matrícula se corresponde con qué alumno**, debemos **proyectar la clave de una tabla a otra**. Esta proyección se realiza aplicando las normas que veremos a continuación dependiendo de la cardinalidad obtenida en los diagramas entidad/relación. Supongamos que se proyecta la clave de alumno a matrícula:*

MATRÍCULAS(**cod_matricula**, ciclo, curso, num_módulos, observaciones, ***cod_alumno***)

PK (Primay Key): cod_matricula.

NN(NOT NULL): ciclo, curso.

FK(FOREIGN KEY): cod_alumno → ALUMNO(cod_alumno)

*Decimos que "**cod_alumno es clave extranjera** o foránea **que referencia al cod_alumno de la tabla de ALUMNOS**". Este campo deberá obligatoriamente tener un valor válido de la tabla ALUMNOS o en su defecto, ser un valor nulo.*

Otro EJEMPLO: *tenemos una tabla de departamentos, cuya clave es NUMERO_DEP. Una tabla de EMPELADO cuya clave es el nombre del empleado (¿?), el campo DEPARTAMENTO sería clave foránea que referencia al NUMERO_DEPT de DEPARTAMENTO.*

EMPLEADO

NOMBRE	DEPARTAMENTO	SALARIO	FECHA_NAC	EXT_TLF
Pablo Montero	14	220.000	10-11-67	6543
Beatriz Cristóbal	13	300.000	20-9-68	6577
J. Luis Martín	11	150.000	25-6-77	6433
Almudena López	13	350.000	4-5-60	6422
Ángel Vallejo	14	400.000	15-4-72	6321
Pedro García	11	200.000	12-3-70	6323

DEPARTAMENTO

NUMERO_DEPT	NOMBRE
11	Contabilidad
13	Marketing
14	Informática

EMPLEADO (NOMBRE, DEPARTAMENTO, SALARIO, FECHA_NAC, EXT_TELEFÓNICA)

DEPARTAMENTO (NUMERO_DEPT, NOMBRE)

- Restricciones de verificación (CHECK). Esta restricción permite especificar condiciones que deban cumplir los valores de los atributos.

  Ejemplo: *supongamos que los alumnos solo pueden matricularse en DAM, DAW y ASIR y que además, no pueden estar matriculados en más de 6 módulos. Podríamos imponer una restricción CHECK sobre el campo ciclo:*

  CK(CHECK): ciclo in (“AISIR”, “DAM”, “DAW”), num_módulos <=6

- Aserciones (ASSERTION), son parecidas a la anterior, pero en este caso en lugar de afectar a una relación como CHECK, puede afectar a dos o más relaciones, en lugar de a un atributo.

- Disparadores (TRIGGER). Las restricciones anteriores son declarativas, sin embargo, este tipo es procedimental, el usuario podrá especificar una serie de acciones distintas ante una determinada condición. El usuario escribe el procedimiento a aplicar dependiendo del resultado de la condición.

2.4. Claves.

- **Superclave**: Identifican a una entidad (pueden ser o no mínimas). Por ejemplo, para un empleado, las superclaves posibles son el DNI, o el DNI+Nombre, o el DNI+Nombre+Numero de la Seguridad Social, etc.
- **Clave Candidata**: Es la mínima Superclave (en el caso anterior el DNI, o el Número de la seguridad social).
- **Clave Primaria**: Es la clave candidata elegida por el diseñador como clave definitiva (en el ejemplo anterior se elegiría el DNI por ser la más representativa para un empleado).
- **Clave foránea**: Es un atributo de una entidad, que es clave en otra entidad.

Ejemplo: *tenemos la tabla perro, con los siguientes dato.*

PERRO (DNI_dueño, nombre_perro, raza, peso, altura, observaciones, cod_perro, chip)

- ✓ *Las superclaves serían todas las combinaciones que identificaran unos perros de otros:*

- *DNI_dueño+ nombre_perro → solo DNI_dueño no sirve porque puede tener más de un perro.*
- *Cod_perro.*
- *Chip.*
- *DNI_dueño+nombre_perro+raza+peso+…(cualquier combinación con DNI_dueño+nombre, cod_perro o chip)*

✓ *Las claves candidatas (candidatas a ser clave primaria) son las mínimas:*
- *Cod_perro. (4 dígitos)*
- *Chip. (12 dígitos)*

✓ *La Primaria sería la más óptima de las dos. En este caso que son las dos numéricas, elegimos el más corto.*

Otro ejemplo!!

PACIENTE_INGRESADO (DNI, nombre, apellidos, tlfno, tlfno_familiar_contacto, dirección, fecha_nacimiento, cod_ingreso, nss, cod_paciente, núm_historia_clínica)

Dni: 8 caracteres alfanuméricos. Nombre: 20 caracteres. Apellidos: 40 caracteres. Teléfonos: 11 caracteres alfanuméricos. Dirección: 60 caracteres alfanuméricos. Fecha_nacimento: tipo fecha. Cod_ingres: 8 caracteres alfanuméricos. Nss: 12 dígitos. Cod_paciente: 8 dígitos. Num_historia_clinica: 20 caracteres alfanuméricos.

✓ *Superclaves:*
- *DNI*
- *DNI+nombre+apellidos*
- *DNI+teléfno*
- *Nombre+apellidos+fecha_nto*
- *Cod_paciente*
- *Cod_paciente+nombre+apellidos*
- *Cod_ingreso*
- *Nss*
- *Num_historia_clinica*
- *Cod_ingreso+nombre+apellidos*
- *Nss+nombre+apellidos*
- *Num_historia_clinica+nombre+apellidos*

✓ *Claves candidatas (las mínimas, candidatas a primaria):*
- *DNI*
- *Nss*
- *Cod_ingreso*
- *Cod_paciente*
- *Num_historia_clínica*

✓ *Primaria (de las candidatas, la más óptima)*
- *Cod_paciente.*

✓ *El resto de claves candidatas, se convierten en alternativas: DNI, Nss, cod_ingreso y Num_historia_clínica.*

3. Transformación al modelo Relacional

La transformación sigue unas reglas básicas, que son:

- Todas las entidades se convierten en tablas (relaciones).
- Todo atributo se transforma en columna dentro de una tabla.
- El identificador único de la entidad se convierte en clave primaria.

Para convertir el MER al modelo relacional estudiaremos las cardinalidades existentes entre las entidades, así como el número de entidades que participan en la ***relación***. Esto va a condicionar la propagación de claves entre las tablas para relacionar la información de las mismas de forma coherente. El estudio de la cardinalidad también nos va permitir prescindir en ocasiones de alguna tabla o la elección de la clave en el caso de relaciones ternarias o n-arias.

En cualquier caso, la transformación obedece a unas reglas que veremos a continuación.

3.1.Transformación de dominios.

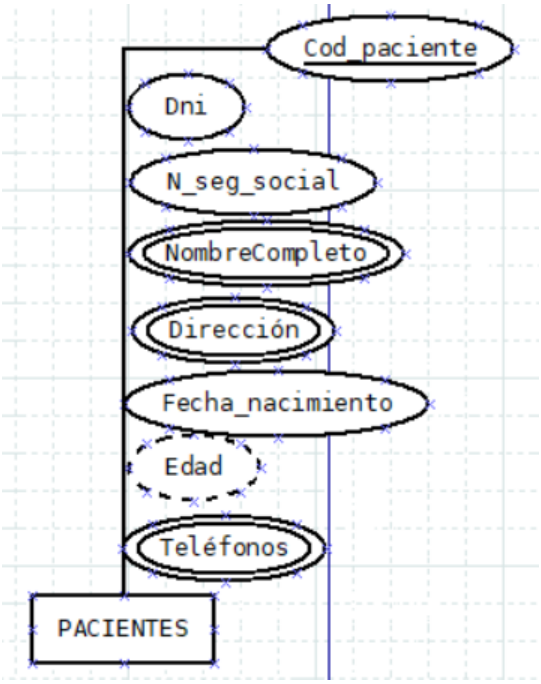
Cuando nos encontramos con un atributo en una entidad del modelo entidad/relación con dominio enumerado como el estado civil, hay que declarar una restricción de tipo CHECK al pasarlo al modelo relacional de la siguiente forma:

PERSONAS(dni, nombre, apellidos, estado_civil, dirección)
PK: dni
CK: estado_civil in [“S”,”C”,”V”,”D”]

3.2.Transformación de atributos.

- **Identificador principal:** Dan lugar a los atributos de la clave primaria **PRIMARY KEY.**
- **Identificador alternativos** Dan lugar a atributos con la restricción **UNIQUE.**
- **Atributos obligatorios:** Dan lugar a atributos con la restricción **NOT NULL.**
- **Atributos optativos:** Dan lugar a atributos con la restricción **NULL.**
- **Atributos compuestos:** Dan lugar a **tantos atributos** como partes componen el atributo compuesto.
- **Atributos multivaluados:** Dan lugar a **una nueva tabla** (relación) cuya clave principal es la **concatenación de la clave primaria de la Entidad en la que se encuentra el atributo multivaluado, y el nombre del atributo multivaluado**.
- **Atributos derivados:** Dan lugar a atributos cuyos valores se obtienen como resultado de algún cálculo sobre otros atributos.

Ejemplo:



Análisis de los atributos de PACIENTES:

- **Identificador**:
  Cod_paciente → PRIMARY KEY
- **Identificadores alternativos**:
  N_seg_social y Dni
- **Compuestos**:
  NombreCompleto → nombre, ape1, ape2.
  Dirección → calle, num, cp, población y provincia.
- **Calculados o derivados**:
  Edad = fecha_Actual – Fecha_nacimiento.
- **Multivalor**:
  Teléfonos que dará lugar a la siguiente tabla

TELEFONOS(Cod_paciente, telefno)
PK: Cod_paciente, teléfono
FK: Cod_paciente → PACIENTES

La traducción a Relacional de los atributos de PACIENTES sería:

PACIENTES (Cod_paciente, Dni, N_seg_social, nombre, ape1, ape2, calle, num, cp, ciudad, provincia, edad)
PK: Cod_paciente.
UK: Dni, N_seg_social
NN: nombre, ape1, ape2.. (los campos que consideremos obligatorios)

TELEFONOS(Cod_paciente, telefono)
PK: Cod_paciente, teléfono
FK: Cod_paciente → PACIENTES

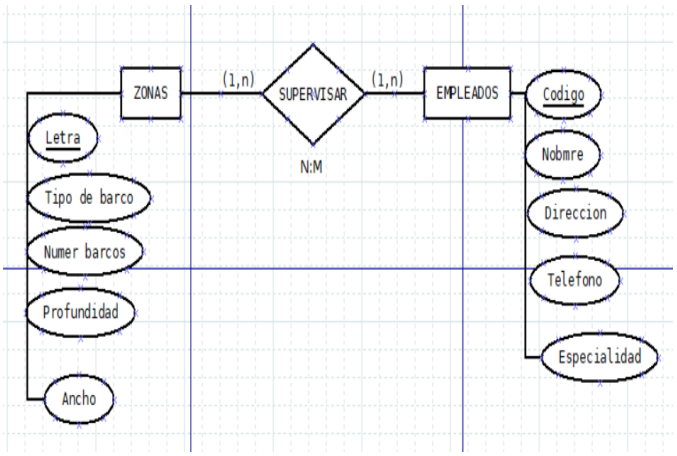
3.3. Transformación Relaciones Binarias.

3.3.1. Binaria. Cardinalidad N:N.

El resultado son **TRES TABLAS**: una por cada entidad y una que genera de la relación.

- ✓ El nombre de las tablas será el nombre de la entidad en plural.
- ✓ El nombre de la tabla de la relación, puede componerse de varias formas:
  - VERBO.
  - ENTIDAD1_VERBO
  - ENTIDAD1_VERBO_ENTIDAD2

- ✓ La clave primaria de la tabla de la relación será la combinación de las claves de las entidades.



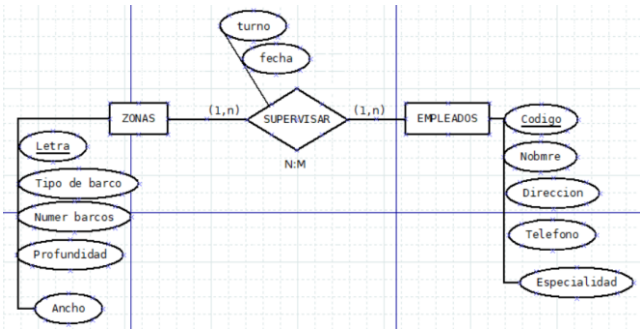
ZONAS(Letra, Tipo_barco, Num_barcos, Profundidad, Ancho)
PK: Letra
CK: Letra in ["A.."Z"]

EMPLEADOS(Codigo, Nombre, Dirección, Telefono, Especialidad)
PK: Codigo

EMPLEADOS_SUPERVISAN_ZONAS(Letra, Codigo)
PK: (Letra, Codigo)
FK: Letra → ZONAS(Letra), Codigo → EMPLEADOS(Codigo)
CK: Letra in ["A.."Z"]

Nota: La tabla podría llamarse EMPLEADOS_SUPERVISAN o SUPERVISAN.

¿Y si hay atributos? Se los lleva la relación.



ZONAS(Letra, Tipo_barco, Num_barcos, Profundidad, Ancho)
PK: Letra
CK: Letra in ["A.."Z"]

EMPLEADOS(Codigo, Nombre, Dirección, Telefono, Especialidad)
PK: Codigo

EMPLEADOS_SUPERVISAN_ZONAS(Letra, Codigo, fecha, turno)
PK: (Letra, Codigo)
FK: Letra → ZONAS(Letra), Codigo → EMPLEADOS(Codigo)
CK: Letra in ["A.."Z"]

Nota: La tabla podría llamarse EMPLEADOS_SUPERVISAN o SUPERVISAN.

En el primer ejemplo la información que podríamos tener en las tablas (la que nos permite la clave primaria) sería esta:

ZONAS				
Letra	**Tipo_barco**	**Num_barco**	**Profundidad**	**Ancho**
A	Tipo1	20	2	6
B	Tipo1	15	2	6
C	Tipo2	30	3	12
D	Tipo2	15	3	12

EMPLEADOS				
Cod	**Nombre**	**Dirección**	**Telefono**	**Especialidad**
100	Ángel	c. Alava…	111111111	Mto.
200	Jose L	c. San juan…	222222222	Vigilancia
300	Sergio	c. Sato Cristo	333333333	Seguridad
400	Albano	c. Budapest…	444444444	vigilancia

EMPLEADOS_SUPERVISAN_ZONAS	
Letra	**Codigo**
A	100
B	100
A	200
B	200
C	200
D	300
C	300
C	400
D	400

Nota: Toda CLAVE EXTRANJERA (FK) DEBE REFERENCIAR A UNA CLAVE PRIMARIA (PK) EN OTRA TABLA. Cuando se trata de una clave primaria SIMPLE (un solo atributo), podemos usar la notación.

Podemos saber qué empleados tienen asignada qué zona.

En el segundo ejemplo, además queremos saber en qué fecha y turno tuvieron asignada una zona, la información de la tabla sería esta:

EMPLEADOS				
Cod	Nombre	Dirección	Telefono	Especialidad
100	Ángel	c. Alava…	111111111	Mto.
200	Jose L	c. San juan…	222222222	Vigilancia
300	Sergio	c. Sato Cristo	333333333	Seguridad
400	Albano	c. Budapest…	444444444	vigilancia

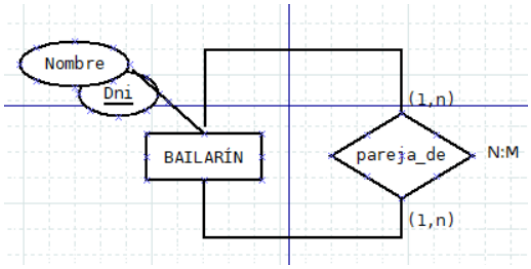
ZONAS				
Letra	Tipo_barco	Num_barco	Profundidad	Ancho
A	Tipo1	20	2	6
B	Tipo1	15	2	6
C	Tipo2	30	3	12
D	Tipo2	15	3	12

EMPLEADOS_SUPERVISAN_ZONAS			
Letra	Codigo	Fecha	turno
A	100	12/02/2023	00:00-8:00
B	100	12/02/2023	00:00-8:00
A	200	12/02/2023	00:00-8:00
B	200	12/02/2023	00:00-8:00
C	200	12/02/2023	00:00-8:00
D	300	12/02/2023	00:00-8:00
A	100	13/02/2023	00:00-8:00
B	100	13/02/2023	00:00-8:00
D	400	13/02/2023	00:00-8:00

En este caso, tenemos que añadir algún atributo a la clave de la tabla EMPLEADOS_SUPERVISAN_ZONAS porque se nos repetiría la clave primaria. La clave primaria será por tanto: (Letra, código, fecha).

¿Y si es una relación reflexiva N:M?

En una academia, los bailarines se emparejan todos con todos con todos. Se supone que todos los bailarines tiene pareja.



¿Cuántos bailarines pueden ser pareja de Pepe Lopez? (1,n).
¿De cuántos bailarines es pareja Pepe López? (1,n).
Cardinalidad → N:M

Como solo hay una entidad**, en este caso se generan dos tablas**:
la entidad y la relación.

BAILARINES(Nombre, Dni)
PK: Dni.
PAREJA_DE(Dni, Dni_pareja)
PK: Dni, Dni_pareja
FK: Dni→BAILARINES(Dni), Dni_pareja→BAILARINES(Dni)

¡¡RAZONA!!

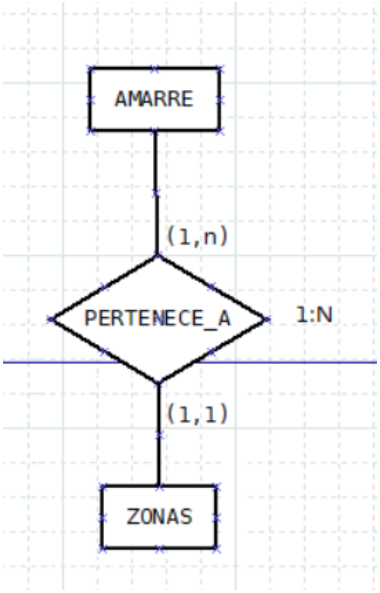
- La academia no solo quiere saber con quién se emparejan los alumnos si no la fecha en lo que lo hicieron. Los bailarines pueden repetir pareja en distintos días. ¿cómo cambiaría la transformación?
- ¿Hay algún caso en el que la tabla PAREJA_DE pueda contener nulos? Dicho de otra forma, ¿algún caso en el que la clave foránea Dni_pareja pudiera ser NULA?

3.3.2. Binaria. Cardinalidad 1:N.

Existen dos soluciones para la transformación de una relación 1:N

A. Solución DOS tablas:

Propagar el identificador principal de la entidad que tiene la cardinalidad 1 hacia la entidad que tiene la cardinalidad N. En este caso no hay que crear una tabla para la relación. Si la relación tuviera atributos también habría que propagarlos hacia la entidad con cardinalidad N.



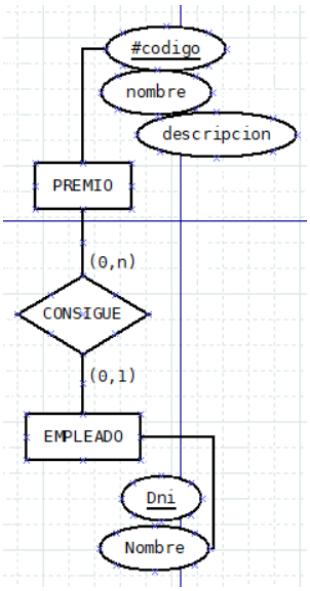
ZONAS (#letra, tipo_barco, num_barcos, profundidad, ancho)
PK: #letra
AMARRES(#num_amarre, lectura_agua, lectura_luz, *#letra*)
PK:num_amarre
FK:#letra → ZONAS(#letra)
NN: #letra (*)

ZONAS				
Letra	Tipo_barco	Num_barco	Profundidad	Ancho
A	Tipo1	20	2	6
B	Tipo1	15	2	6
C	Tipo2	30	3	12

AMARRES			
#num_amarre	Lectura_agua	Lectura_luz	#Letra
1	15	1.10	A
2	35	0.8	A
3	12	0.4	A
10	15	0.9	B
11	20	1.2	B
23	23	4.0	C
34	45	3.4	C
35	67	1.7	C

(*) La clave foránea debe declararse NOT NULL con participación: **(1,1) – (0,n) ó (1,n).** En este ejemplo, al ser la participación de ZONAS (1,1)-(1,n), estamos diciendo que todos los amarres están obligatoriamente en una zona por lo que la clave foránea #letra en la tabla AMARRES siempre tendrá un valor distinto de NULL.

Pero no siempre es así. Con participación (0,1) - (0,n), la clave foránea podría ser NULA y no podemos declararla como NOT NULL. Veamos un ejemplo:



EMPLEADOS (DNI, Nombre)
PK: DNI
PREMIOS (#codigo, nombre, descripción, DNI)
PK:#codigo
FK:#DNI → EMPLEADOS (DNI)

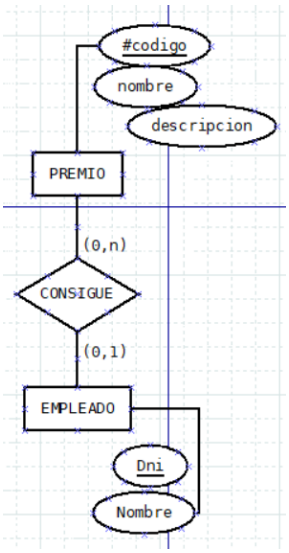
EMPLEADOS	
DNI	Nombre
10D	Ángel
20B	Jose L
30C	Sergio
40C	Albano

PREMIOS			
cod	Nombre	Descripción	DNI
1	Emotive	Motivación y moral en el trabajo	10D
2	Purpose	Sentido del propósito en el trabajo	20B
3	Leal	Lealtad en el trabajo	10D
4	MaxVentas	Ventas superiores al millón de euros.	NULL
5	Cartera	Fidelización de 100 clientes en un año.	NULL
6	Empatía	Empatia en entornos de trabajo.	30C

En este caso, se puede asumir que habrá campos NULOS en las tablas u optar por la solución B, una traducción a TRES TABLAS.

B. Solución TRES tablas:

Como si se tratara de una relación N:M, con la diferencia que la clave primaria solamente utiliza el identificador de la entidad implicada con cardinalidad N.



EMPLEADOS (DNI, Nombre)
PK: DNI
PREMIOS (#codigo, nombre, descripción)
PK:#codigo
EMPLEADOS_CONSIGUEN_PREMIO(#codigo, DNI)
PK: #codigo
FK: #codigo → PREMIOS(#codigo), DNI → EMPLEADOS(DNI)
NN:DNI

EMPLEADOS_CONSIGUEN_PREMIOS	
#codigo	DNI
1	10D
2	20B
3	10D
6	30C

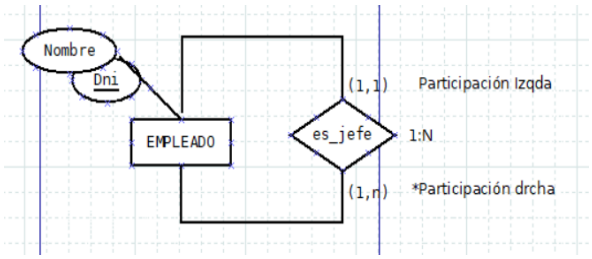
EMPLEADOS	
DNI	Nombre
10D	Ángel
20B	Jose L
30C	Sergio
40C	Albano

PREMIOS		
cod	Nombre	Descripción
1	Emotive	Motivación y moral en el trabajo
2	Purpose	Sentido del propósito en el trabajo
3	Leal	Lealtad en el trabajo
4	MaxVentas	Ventas superiores al millón de euros.
5	Cartera	Fidelización de 100 clientes en un año.
6	Empatía	Empatia en entornos de trabajo.

Esta transformación en cardinalidad 1:N es recomendable cuando la relación tiene muchos atributos y cuando se produce nulidad en claves foráneas.

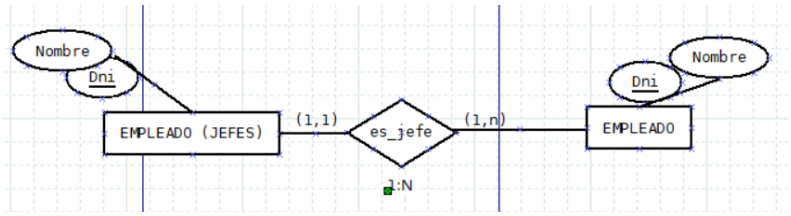
C. Transformación 1:N en REFLEXIVAS

En relaciones de este tipo, se puede optar por la solución A y propagar la clave ó la solución B creando una nueva tabla para la relación. Veámoslo con un ejemplo.



En una empresa, **todos los empleados tienen un único jefe y todos los jefes tienen empleados a su cargo.**

Izqda: ¿Cuántos jefes para un empleado? (1,1).
Drcha: ¿Cúantos empleados para un jefe? (1,n).



Si nos cuesta formularnos las preguntas con la relación reflexiva, siempre podemos dibujarlo de esta forma y posteriormente poner la participación y la cardinalidad en el diagrama original.

Solución A → Propagación de claves:

EMPLEADOS (Nombre, Dni, Dni_jefe)
PK: Dni
FK: Dni_jefe → EMPLEADOS(Dni)
NN:Dni_jefe

La clave foránea Dni_jefe en este caso PUEDE SER NOT NULL porque la participación es (1,1) lo que indica que todos tienen un jefe. En el caso del jefe máximo apuntaría a sí mismo.

EMPLEADOS		
DNI	Nombre	Dni_jefe
10D	Ángel	10D
20B	Jose L	10D
30C	Sergio	10D
40C	Albano	20B
50D	Lidia	20B
60D	Lola	30C

Solución B → Crear una nueva tabla.

EMPLEADOS (Nombre, Dni)
PK: Dni

ES_JEFE_DE(Dni, Dni_jefe)
PK: Dni
FK: Dni → EMPLEADOS(Dni),
Dni_jefe→EMPLEADOS(Dni)
NN:Dni_jefe

EMPLEADOS	
DNI	Nombre
10D	Ángel
20B	Jose L
30C	Sergio
40C	Albano
50D	Lidia
60D	Lola

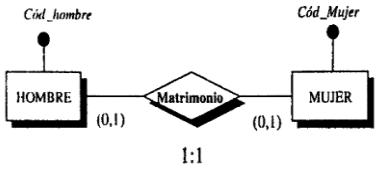
ES_JEFE_DE	
DNI	Dni_jefe
10D	10D
20B	10D
30C	10D
40C	20B
50D	20B
60D	30C

3.3.3. BINARIAS. Cardinalidad 1:1

Una relación 1:1 es un caso particular de una relación N:M, por lo que no hay una única forma de transformación. Los criterios para aplicar una u otra transformación, depende de la participación:

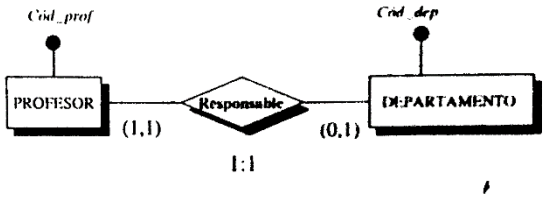
a) **Si las dos entidades que se asocian poseen cardinalidades (0,1)** puede ser conveniente transformarlo en una nueva tabla (relación), utilizando el identificador de uno de los extremos como clave principal, y el otro como identificador alternativo.

ME/R



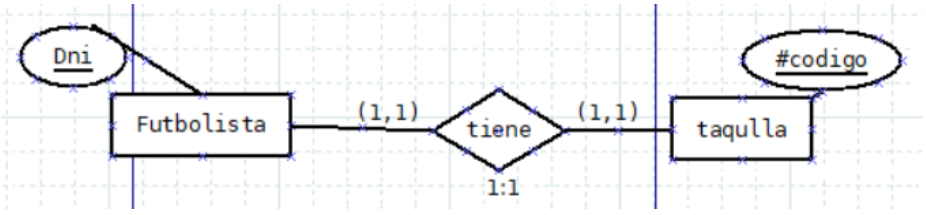
MUJERES(#cod_mujer, nombre)
PK: #cod_mujer
HOMBRES (cod_hombre, Nombre)
PK: cod_hombre
MATRIMONIOS(cod_mujer, cod_hombre)
PK: cod_mujer
FK: cod_mujer → MUJERES(cod_mujer),
Cod_hombre → HOMBRES(cod_hombre)
NN: cod_hombre, cod_mujer

b) **Si una entidad tiene cardinalidad (0,1) y la otra entidad tiene cardinalidad (1,1)** conviene propagar la clave de la entidad con cardinalidad (1,1) a la tabla resultante con cardinalidad (0,1).



PROFESORES(cod_profesor)
PK: #cod_profesor
DEPARTAMENTOS(cod_dep, cod_profesor)
PK: cod_dep
FK: cod_profesor → PROFESORES(cod_profesor),
NN, UK: cod_profesor

c) **Si ambas entidades tienen cardinalidad (1,1)** se puede propagar una de las dos entidades sobre la otra como en el **caso a**. En este caso la clave ajena debe contener la cláusula NOT NULL.



TAQUILLAS(#codigo)

PK:#codigo

FUTBOLISTAS (Dni, #codigo)

PK: Dni.

FK: #codigo → TAQUILLAS(#codigo)

NN:#codigo

FUTBOLISTAS (Dni)

PK: Dni.

TAQUILLAS(#codigo, DNI)

PK:#codigo

FK: DNI → FUTBOLISTAS(Dni)

NN:DNI

Cualquier de las opciones vale. Dependerá de lo que sea más significativo a nivel semántico.

En este caso ((1,1)-(1,1)), se podría valorar hacer una transformación a **una sola tabla**. En el caso de haya pocos atributos y que de la semántica del problema sea adecuada. Supongamos el ejemplo de las taquillas pero en el entorno de los empleados: "Empleado tiene taquilla". Todos los empleados tienen una única taquilla y de estas no queremos guardar información relevante, solo el número de taquilla que le corresponde a cada empleado. Podríamos entonces, hacer esta transformación:

EMPLEADOS(dni, nombre, dirección, teléfono, *num_taquilla*)

PK: dni.

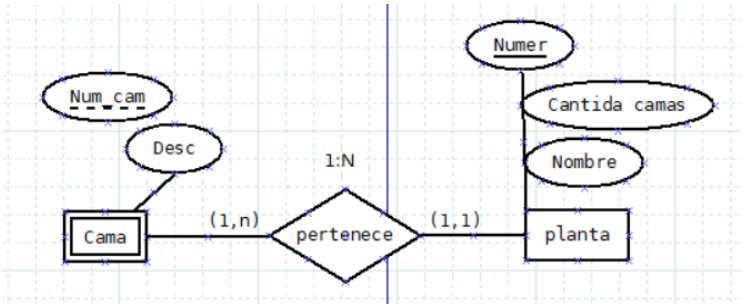
NN, UK: num_taquilla. (*not null, unique*)

Aunque lo más conveniente es crear una tabla por cada entidad y proyectar claves.

3.4.Transformación de Entidades Débiles.

Realizamos una propagación de clave desde la entidad fuerte a la entidad débil. La clave principal de la entidad débil debe contener a la clave foránea propagada.

Ejemplo:



PLANTAS(Numer, Cantidad, Nombre)

PK: Numer

CAMAS(Planta, Num_cam, Desc)

PK: Planta, Num_cam

FK: Planta → PLANTAS(Numer)

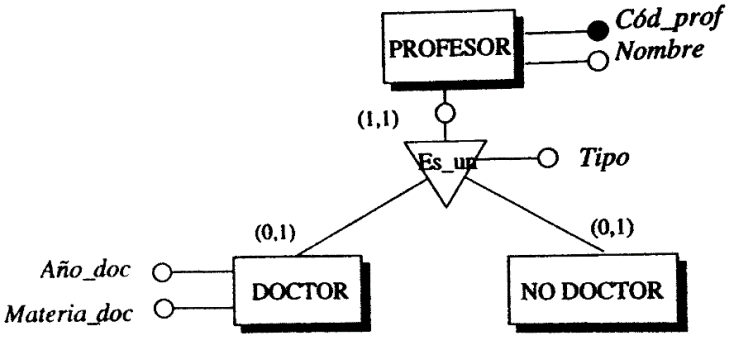
3.5. Transformación de Jerarquías

Existen tres soluciones de transformación:

a) **Englobar todos los atributos de los tipos y de los subtipos en una sola entidad**. Esta es una buena solución cuando lo subtipos se diferencian en muy pocos atributos y no hay relaciones con las entidades de los subtipos.

b) **Crear la relación de tipo padre (supertipo) y tantas relaciones como hijos (subtipos) haya**. Esta es la solución más común, muchos atributos distintos, relaciones con los subtipos, etc.

c) **Crear tantas relaciones como subtipos haya, insertando en ellos los atributos del padre**.



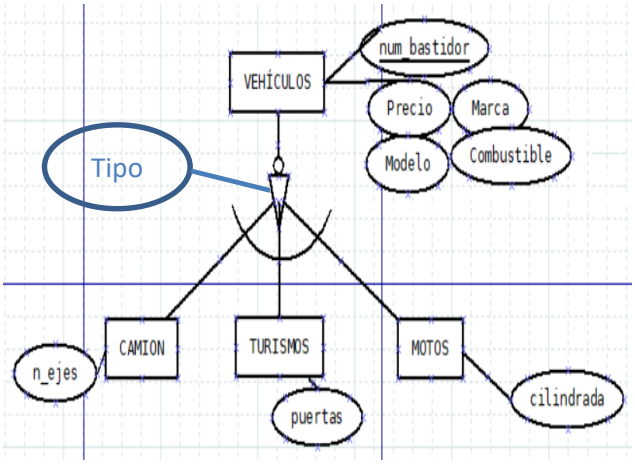
A)
PROFESOR (*Cód_prof*, *Nombre*, ..., *Tipo*)

B)
PROFESOR (*Cód_prof*, *Nombre*, ..)
DOCTOR (*Cód_prof*, ..*Año_doc*, *Materia_doc*)
NO_DOCTOR (*Cód_prof*...)

C)
DOCTOR (*Cód_prof*, *Nombre*,.., *Año_doc*, *Materia_doc*)
NO_DOCTOR (*Cód_prof*, *Nombre*, ...)

En todos los casos hay que crear las aserciones y los disparadores adecuados para cumplir las restricciones que vienen implícitas en la jerarquía.

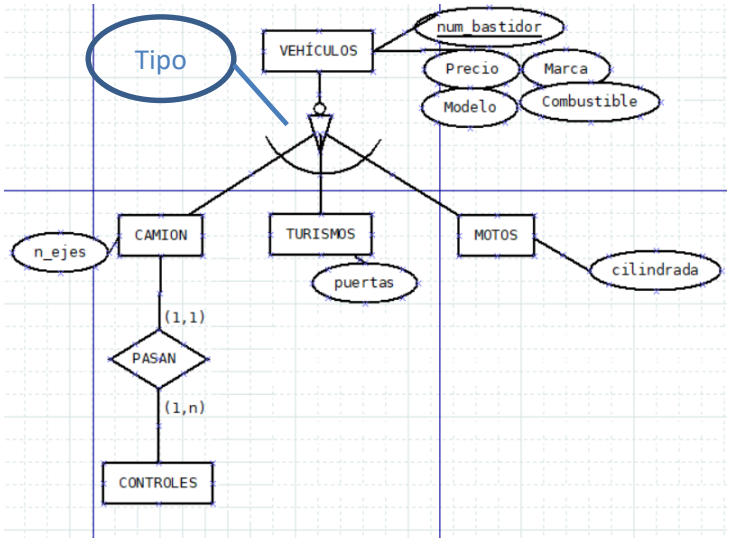
Ejemplo **A** → ***"Todo al padre"***



VEHÍCULOS(num_bastidor, precio, marca, modelo, combustible, tipo, n_ejes, puertas, cilindrada)
PK: num_bastidor
NN: tipo, precio, marca, modelo, combustible
CK: Tipo in ["Cam","Tur","Mot"]

- ✓ Subtipos con pocos atributos.
- ✓ Subtipos sin relaciones.
- ✓ Los atributos n_ejes, puertas y cilindradas admitirán valores NULOS.

Ejemplo **B** → ***"Padres e hijas"***

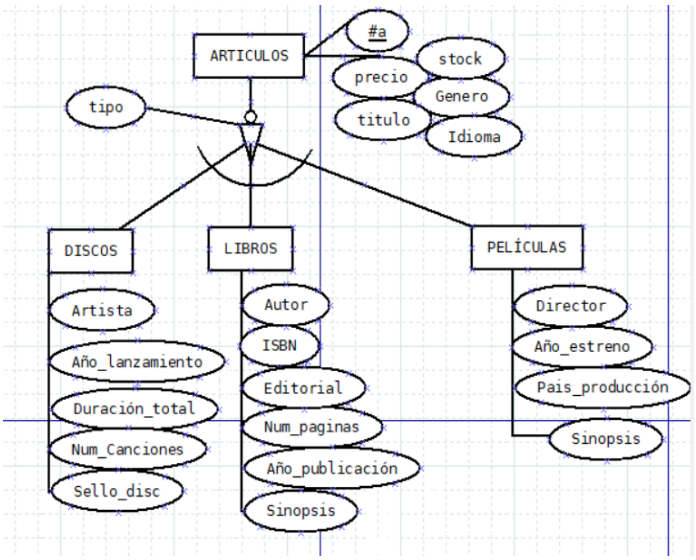


VEHÍCULOS(num_bastidor, precio, marca, modelo, Combustible, tipo)
PK: num_bastidor
NN: tipo, precio, marca, modelo, combustible
CK: Tipo in ["Cam","Tur","Mot"]
CAMIONES (num_bastidor, n_ejes)
PK: num_bastidor.
FK: num_bastidor → VEHÍCULOS
NN: n_ejes
TURISMOS (num_bastidor, puertas)
PK: num_bastidor.
FK: num_bastidor → VEHÍCULOS
NN: puertas
MOTOS (num_bastidor, cilindrada)
PK: num_bastidor.
FK: num_bastidor → VEHÍCULOS
NN: cilindrada

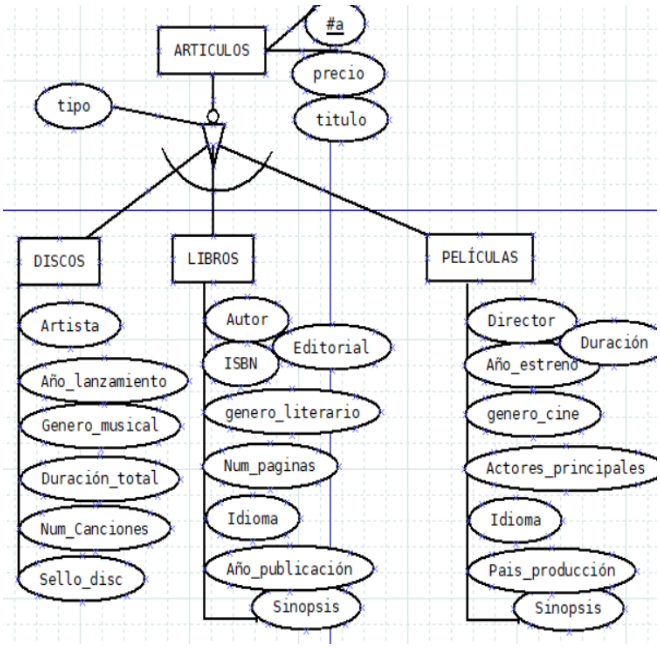
En el ejemplo anterior, se contempla la misma situación que el A) pero en este caso, la entidad CAMION está relacionado con otra entidad del MER, por lo que es recomendable traducir padres e hijas como hemos hecho.

Otra situación es el ejemplo de la derecha, donde vemos que tanto padre como hijas tienen gran cantidad de atributos. Se podría estudiar cuáles son o podrían ser comunes aunque la semántica sea distinta lo cual puede llevar a errores por parte de los usuarios. Vemos que atributos comunes como la sinopsis que no es común a las tres entidades, etc.

Lo más sensato es traducir padres e hijas.



Ejemplo **C** → ***"Todo a las hijas"***



DISCOS (#a, precio, titulo, Artista, Año_lanza …,Sello_disc)
PK: #a

LIBROS (#a, precio, titulo, Autor, ISBN, …., sinopsis)
PK: #a

PELÍCULAS (#a, precio, titulo, Director, Año_est…Sinopsis)
PK: #a

- ✓ Muchos atributos en las hijas, pocos en el padre.
- ✓ Desaparece el tipo.
- ✓ No hay claves foráneas.

3.6.Transformación de relaciones N-ARIAS

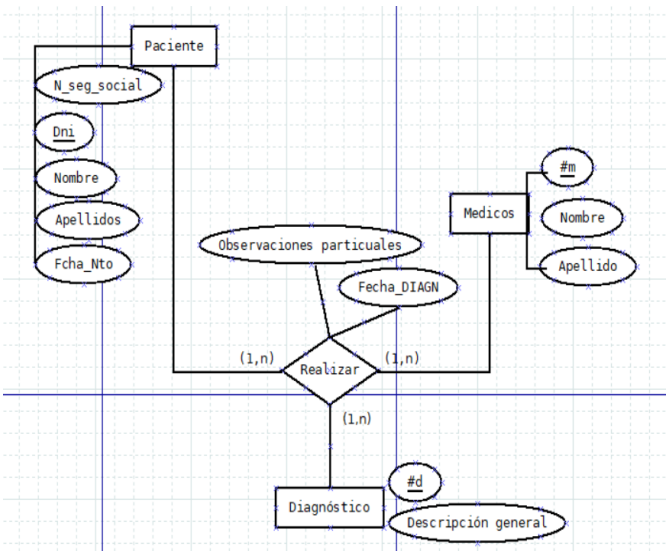
En este tipo de relaciones que agrupan 3 o más entidades, se seguirán las siguientes pautas:

- ✦ Cada entidad se convierte en una tabla siendo la clave primaria el identificador de la misma.
- ✦ La relación también se convierte en una tabla y tendrá como campos los atributos de la relación si los tuviera y los campos clave de las entidades.
- ✦ La clave primaria de la tabla de la relación será la concatenación de las claves aportadas por las entidades con participación N.

Ejemplos:

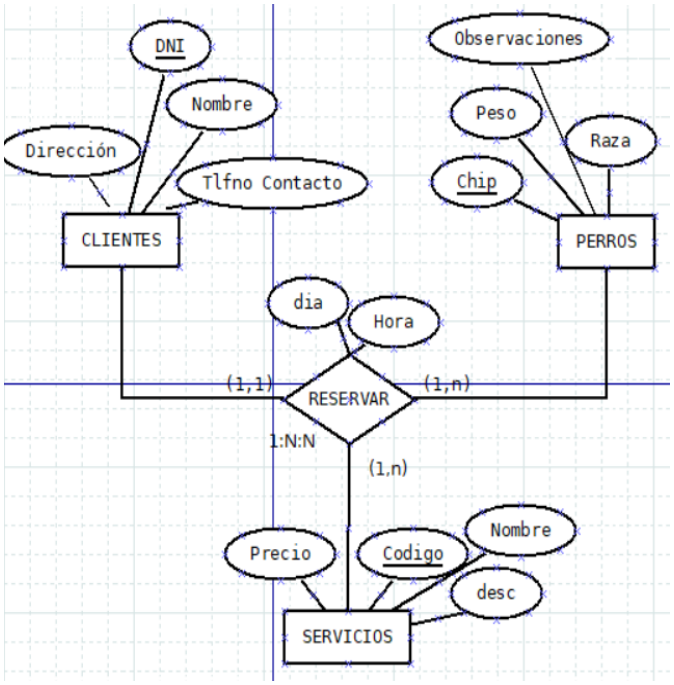
- ✦ **RELACIÓN N:N:N** → *Un paciente puede recibir varios diagnósticos. Un médico puede realizar varios diagnóstico. Los diagnósticos son una descripción general de una enfermedad por lo que varios pacientes pueden ser diagnosticados de "dermatitis".*

PACIENTES(N_seg_social, Dni, Nombre, Apellidos, Fecha_Nto)
PK: Dni
DIAGNOSTICOS(#d, desc)
PK: #d
MEDICOS(#m, Nombre, Apellido)
PK: #m
REALIZAR(Dni, #d, #m, Obs, Fecha_Diag)
PK: (Dni, #d, #m)
FK: Dni → PACIENTES
#d → DIAGNOSTICOS
#m → MEDICOS



RAZONA: ¿podríamos almacenar en la base de datos dos diagnósticos de la misma enfermedad (dermatitis) realizados por el mismo médico (Dtra. Soto) sobre el mismo paciente (Juan López) en dos fechas diferentes (el 12/02/2022 y el 10/12/2022)?

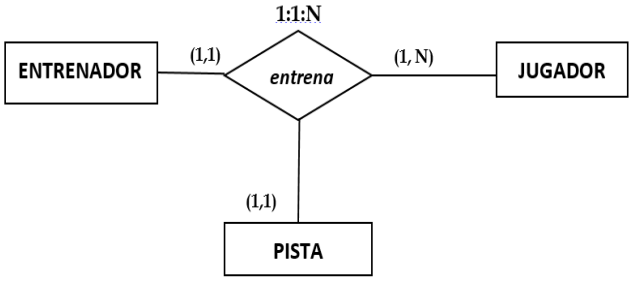
- **Relación 1:N:N** → *Un perro solo pertenece a un cliente. El cliente puede tener 1 o varios perros.*



CLIENTES(Dni, Nombre, Direccion, Tlfno_cto)
PK: Dni
PERROS(Chip, peso, raza, Obs)
PK: Chip
SERVICIOS(Codigo, Nombre, Precio, des)
PK: Codigo
RESERVAR(Dni,Chip,Codigo, dia, hora)
PK: Chip, Codigo
FK: Dni → CLIENTES
Chip → PERROS
Codigo → SERVICIOS

RAZONA: Podríamos almacenar un histórico de citas, Con la clave que hemos elegido. Es decir, podríamos Saber qué días se cortó el pelo Ricki (chip:0001234z)?

- **Relación 1:1:N** → *"Un jugador entrena con un único entrenador siempre en la misma pista. El entrenador puede entrenar en varias pistas. El entrenador puede entrenar a varios jugadores."*



ENTRENADORES (#e)
JUGADORES (#j)
PISTAS(#p)
ENTRENA(#j, #e, #p)
PK: #j
FK: #e → ENTRENADORES,
#j → JUGADORES,
#p → PISTAS

Como solo JUGADOR tiene una participación n, la clave primaria de ENTRENA sería la clave de jugador.

Si quisiéramos llevar un histórico de entrenamientos con día y hora, estos datos se colgarían de la relación entrena quedando ENTRENA de la siguiente forma:

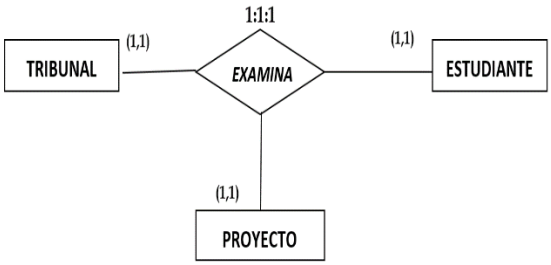
ENTRENA(#e, #j, #p, dia, hora)

En este caso, no podemos elegir como clave primaria solo el código del jugador pues se nos va a repetir tantas veces entrenamientos realice:

ENTRENA				
#j	#e	#p	Dia	hora
100	10	1	1/10/2023	10:00
101	20	2	1/10/2023	10:00
100	10	1	2/10/2023	20:00
100	10	1	3/10/2023	20:20

Tendríamos que añadir al código del jugador, el día del entrenamiento. Incluso, si un jugador puede entrenar a dos horas distintas, la hora.

- **Relación 1:1:1 → "***Un tribunal examina a un estudiante de su proyecto de fin de carrera***"**



TRIBUNAL (id_tribunal)
ESTUDIANTE(id_estudiante)
PROYECTO (id_proyecto)
EXAMINA(id_estudiante, id_proyecto, id_tribunal)
PK: id_estudiante ó id_proyecto ó id_tribunal (UNA)
FK: id_estudiante → ESTUDIANTES
FK: id_proyecto → PROYECTOS
Fk: id_tribunal → TRIBUNALES

4. NORMALIZACIÓN

La Normalización se realiza para optimizar la estructura y por tanto el funcionamiento de la **BD**, centrándose en dos objetivos:

- Eliminar redundancias. Almacenar el dato solo una vez.
- Almacenar hechos distintos en sitios distintos para evitar las anomalías de modificación, de inserción y de borrado.

Se analizarán las tablas, comprobando los requisitos de cada forma normal. Cuando estos no se cumplan, habrá que realizar las transformaciones necesarias para la tabla adopte dicha forma normal. Estas transformaciones suelen consistir en fragmentar tablas complejasen tablas más simples con un número de atributos mínimo que defina la entidad y establecer relaciones entre ellas a través de las claves.

Existen 6 formas normales. Se dice que "***una tabla está en una forma normal, cuando satisface las restricciones impuestas por dicha forma normal***".

4.1. Dependencias funcionales

- **Dependencia funcional parcial.**

Se dice que un atributo Y depende funcionalmente de otro atributo X, si cada valor de X le corresponde un único valor de Y. **X** recibe el nombre de ***determinante*** y podríamos decir

cualquiera de estas expresiones:

- ***"Y depende funcionalmente de X"***.
- ***"X determina a Y"***. *(A también es conocido como determinante).*
- *"X* ***implica a Y"***.

Consideremos la tabla para estudiar las dependencias funcionales:

PRODUCTOS				
Cod	**Nombre**	**Precio**	**Desc**	**Especialidad**
100	**Fideo Cabellín**	**50.34**	**"……."**	**Mantenimiento**
200	**Tallarín huevo**	**60.30**	**"……."**	**Seguridad**
600	**Macarrón n3**	**45.00**	**"……."**	**Seguridad**
300	**Punto de Lluvia**	**67.90**	**"……."**	**Vigilancia**
400	**Macarrón n4**	**45.00**	**"……."**	**Seguridad**
500	**Fideo n4**	**40.30**	**"……."**	**vigilancia**

COD → Nombre, a un código le corresponde un nombre único.

- **Dependencias funcional completa.**

Dada una combinación de atributos X(X1, X2,...), se dice que Y tiene dependencia funcional completa de X, si depende funcionalmente de X, pero no depende de ningún subconjunto del mismo. Por ejemplo:

COMPRAS (CódigoProducto, CódigoProveedor, Cantidad, FechaCompra)
PK: CodigoProducto, CódigoProveedor

(CódigoProducto,CódigoProveedor) → FechaCompra

puesto que la FechaCompra es única para la combinación de CódigoProducto y códigoProveedor (se puede hacer un pedido al día de cada producto a cada proveedor), y sin embargo, se pueden hacer varios pedidos del mismo producto a diferentes proveedores

CodigoProducto	**CodigoProveedor**	**Cantidad**	**FechaCompra**
100	**4001**	**1**	**18/10/2023**
200	**4001**	**6**	**18/10/2023**
100	**4002**	**4**	**18/09/2023**
200	**4002**	**6**	**20/10/2023**
600	**4003**	**5**	**22/10/2023**
500	**4001**	**3**	**24/10/2023**

- **Dependencia funcional transitiva.**

Dados los atributos X, Y y Z de la relación R en la que existen las siguientes dependencias funcionales:
$X \rightarrow Y$;

Y → Z ; se dice que Z tiene una ***dependencia transitiva*** respecto X a través de Y: X → Z.

- **Dependencia multivaluada o de valores múltiples.**

  Sean X e Y dos descriptores, X multidetermina a Y (X→→ Y) si para cada valor de X existe un conjunto bien definido de valores posibles en Y, con independencia del resto de los atributos de la relación.

  No existe dependencia funcional alguna entre los valores de los atributos.

  Este tipo de dependencias produce redundancia de datos.

  Ejemplo:

  En este caso curso →→ profesor

  curso →→ texto

curso	profesor	texto
Bases de Datos	Ullman	Fundamental of DataBase Systems
Bases de Datos	Ullman	A First Course in Data Base Systems
Bases de Datos	Ullman	Matemáticas Discretas
Bases de Datos	Pérez	Fundamental of DataBase Systems
Bases de Datos	Pérez	A First Course in Data Base Systems
Bases de Datos	Pérez	Matemáticas Discretas
Modelos Discretos	Candia	Introduction to Algorithms
Modelos Discretos	Candia	Matemáticas Discretas
Modelos Discretos	Pérez	Introduction to Algorithms
Modelos Discretos	Knuth	Introduction to Algorithms
Modelos Discretos	Knuth	Matemáticas Discretas

4.2. Formas Normales

- **Primera Forma Normal (1FN)**

  Una tabla se encuentra en Primera Forma Normal si y sólo si los valores que componen el atributo de una fila son únicos, es decir, en un atributo ***no deben aparecer valores multievaluados*** y por tanto tienen que ser elementales y únicos.

  Al valor que se repite se le denomina conjunto repetitivo, y diremos que una tabla se encuentra en Primera Forma Normal si no contiene grupos repetitivos.

  Por ejemplo, la siguiente tabla no encuentra en Primera Forma Normal:

Cod_material	Descripción	Medidas
39	Tornillo	5,6-7-23
67	Arandela	4-8-56
461	Broca	2-3-4

Transformación a 1ª FN en caso de campos multievaluados:

1. Localizar atributos clave primaria.
2. Descomponer la tabla en dos proyecciones (dos tablas nuevas).

Tabla 1 (R1): clave primaria y los atributos que no son multievaluados.

Cod_material	**Descripción**
39	Tornillo
67	Arandela
461	Broca

Tabla 2 (R2): Incluir la clave primaria de la tabla e incluir una fila por cada valor del atributo multievaluado. La clave primaria de esta nueva tabla sería la conjunta de su tabla original más el campo multievaluado.

Cod_material	**Medidas**
39	5,6
39	7
39	23
67	4
67	8
67	56

Esta transformación expresada en modelo relacional sería:

R1(cod_material, descripcion)
PK: cod_material
R2(cod_material, medidas)
PK: cod_material, medidas
FK: cod_material → R1

Si suponemos que la tabla original fuera **PRODUCTOS**. R1, podríamos mantenerla con ese nombre "***PRODUCTOS***" y R2, podríamos crearla como "***MEDIDAS***".

✦ Segunda Forma Normal (2FN)

Una tabla se encuentra en Segunda Forma Normal **si está en 1FN y, además**:

- *Todos los atributos secundarios (los que no pertenecen a la clave principal)* ***dependan de la clave principal en su totalidad y no de una parte de ella****, es decir, todos los atributos secundarios deben depender funcionalmente de la clave.*

Esta Forma Normal **sólo se considera si la clave principal es compuesta**. Si la clave es simple, la tabla ya se encuentra en Segunda Forma Normal.

Por ejemplo, para la siguiente tabla:

DNI	**Empresa**	**Nombre_empleado**	**Sueldo**

Hacemos un estudio de las dependencias funcionales con respecto a la clave primaria:

(DNI, Empresa) → Sueldo

DNI → Nombre_empleado

El nombre del empleado solo depende del DNI. El empleado no cambia de nombre cada vez que cambia de empresa….Por tanto, ésta tabla, no se encuentra en Segunda Forma Normal.

Transformación a 2ª FN en caso de atributos que no dependen totalmente de la PK.

1. Descomponer la tabla de en tantas proyecciones como dependencias se hayan encontrado. En este caso, dos proyecciones.

   Tabla 1 (R1): la parte de la clave que tiene depencias y los atributos que determina:

DNI	**Nombre_empleado**

   Tabla 2 (R2): clave primaria y atributos que sí dependen totalmente de la misma.

DNI	**Empresa**	**Sueldo**

   Esta transformación expresada en modelo relacional sería:

R1(DNI, Nombre_Empleado)
PK: DNI
R2(Dni, Empresa, Sueldo)
PK: Dni, Empresa
FK: Dni → R1

Si suponemos que la tabla original fuera **EMPLEADOS**. R1, podríamos mantenerla con ese nombre "**EMPLEADOS**" y R2, podríamos crearla como "***EMPRESAS***".

- **Tercera Forma Normal (3FN)**

Una tabla se encuentra en Tercer Forma Normal si se encuentra en Segunda Forma Normal y, además:

> "**No existen atributos secundarios que dependan funcionalmente de otros atributos secundarios, es decir un atributo sólo debe ser conocido por medio de la clave principal**".

Una tabla que no tiene atributos secundarios se encuentra en Tercera Forma Normal, y una tabla que contenga un solo atributo secundario también.

Por ejemplo la siguiente tabla.

Num_Fact	Total_factura	Fecha	Cod_Cli	Nom_Cli	Dir_Cli

1. Estudiamos las dependencias de los secundarios.
   Total_factura → ¿??
   Fecha → ¿??
   Cod_Cli → Nom_Cli, Dir_Cli (Como existen dependencias entre los secundario, la tabla no se encuentra en 3ª forma normal)
2. **Transformación a 3ª FN en caso de dependecias entre secundarios.**
   Formar una proyección con estos atributos, convirtiendo el determinante en clave:

Cod_Cli	Nobre_Cli	Dir_Cli

Formar otra proyección que contendrá el atributo del cual dependían los otros campos (que será clave principal) y los campos que dependían de él

Num_Fact	Total_factura	Fecha	Cod_Cli

Esta transformación expresada en modelo relacional sería:

R1(Cod_Cli, Nombre_Cli, Dir_Cli)
PK: DNI
R2(Num_fact, Total_factura, Fecha, Cod_Cli)
PK: Num_Fact
FK: Cod_cli → R1

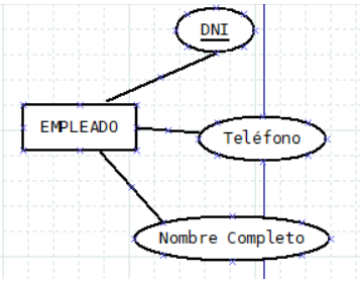
Si suponemos que la tabla original fuera **FACTURAS**. R1, podríamos crearla con el nombre "**CLIENTES**" y R2, podríamos crearla como "***FACTURAS***".

5. Recomendaciones de Diseño

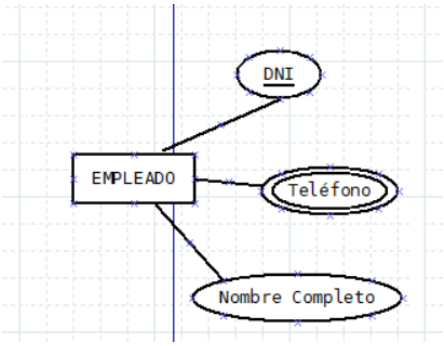
5.1. ¿Entidad o Atributo?

Dependerá de si el objeto que se nombra, tiene características o no. Del número, etc. De la semántica del enunciado en sí. Vamos a ver con un ejemplo, cómo cambiaría el modelo E/R dependiendo de cómo nos pidan almacenar el teléfono.

- ✓ Queremos almacenar el empleado con DNI, nombre completo y **UN** teléfono:

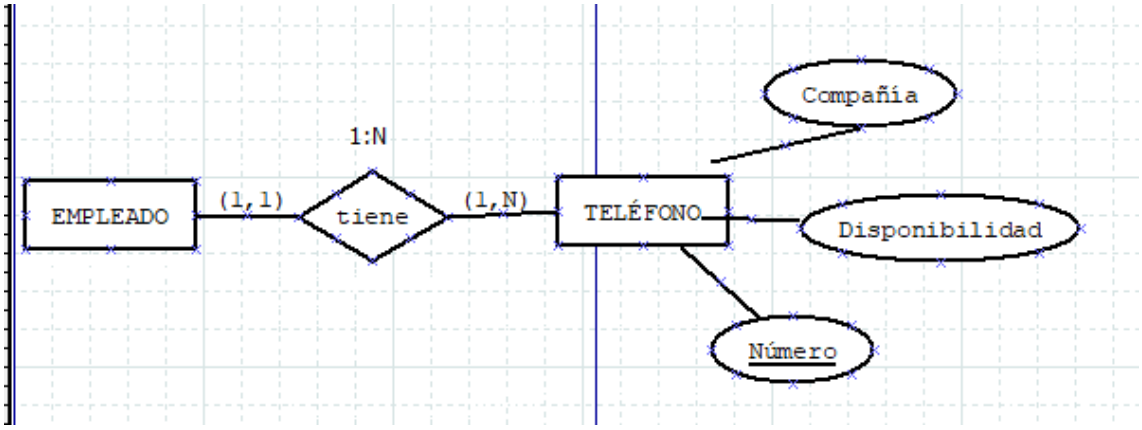


✓ Queremos almacenar el empleado con el DNI, nombre completo y **todos los teléfonos que tenga**:



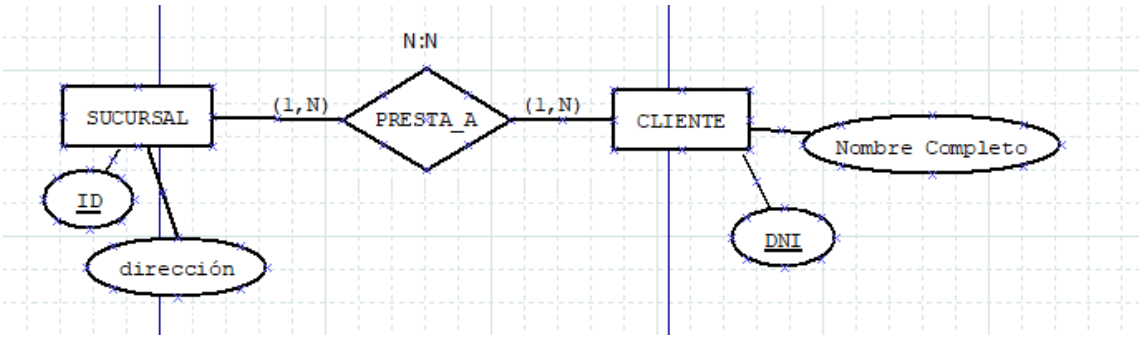
Podemos considerar el mismo esquema pero marcamos el atributo teléfonos como ***MULTIVALOR aunque como ya hemos visto entre las características del modelo Relacional Normalizado, NO PODEMOS TENER EN UNA TABLA ATRIBUTOS DE ESTE TIPO y tendremos que solucionarlo.***

✓ Queremos almacenar el empleado con el DNI, nombre completo y **todos los teléfonos que tenga *y las horas* en las que podemos llamarle a lo largo del día en cada uno de los teléfonos a parte de la compañía telefónica de cada línea**.

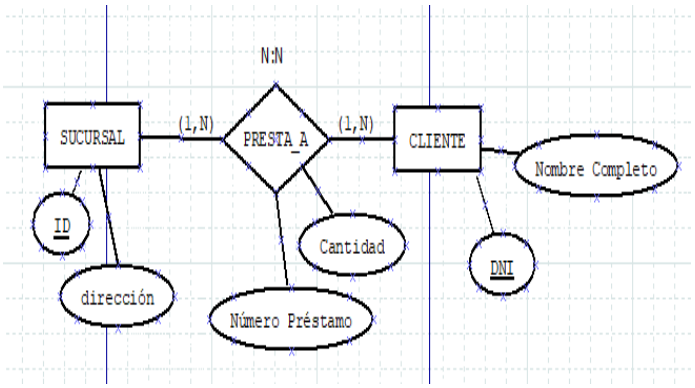


5.2.¿Relación o Entidad?

✓ Queremos tener información de las sucursales que prestan dinero a los clientes.



- ✓ Teniendo en cuenta que un cliente le puede prestar dinero diferentes sucursales y sabiendo que al ser N:N la relación presta_a y que daría lugar a una tabla por lo tanto y se repetirían las claves… podemos solucionarlo añadiendo la fecha a la relación y la cantidad para que sea distinto:



SUCURSALES(id, dirección)
Pk: id
CLIENTES(Dni, Nombre)
Pk: Dni
PRESTA_A(id, Dni, Num_Pres, Cantidad)
Pk:(id, Dni, Num_Pres)
Fk: id → SUCURSALES
Fk: dni → CLIENTES

- ✓ El problema que tiene este tipo de relaciones son los errores derivados de la actualización.

Ejemplo.- Imaginemos que Pablo Lopez y Germán Sánchez son titulares del mismo préstamo; en la tabla de PRESTA_A tendremos:

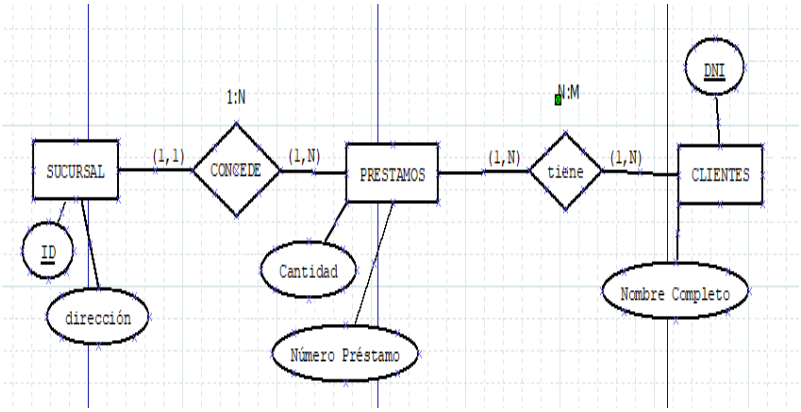
SUCURSALES	
Id	direccion
1	C/SAN FRANCISCO
2	AVDA. JUAN CARLOS I

CLIENTES	
Dni	Nombre
1111111B	PABLO LOPEZ
2222222C	GERMÁN SÁNCHEZ

PRESTA_A			
Dni	Id	NumPrestamo	Cantidad
111111B	1	1000	2345000
222222C	1	1000	2345000

Si tenemos que actualizar el importe y solo lo actualizamos para uno de los clientes, ya habría **una incoherencia**.

- ✓ La forma de solucionar esto, es considerar una entidad préstamo:



SUCURSALES*(id, dirección)*
Pk: id
CLIENTES*(Dni, Nombre)*
Pk: Dni
PRESTAMOS*(Num_Pres, Cantidad, id_sucursal)*
Pk: Num_Pres
Fk: id_sucursal → SUCURSALES
TIENEN*(Dni, Num_prestamo)*
Pk: dni, Num_prestamo
FK: dni → CLIENTES
Num_Prestamo → PRESTAMOS

SUCURSALES	
Id	direccion
1	C/SAN FRANCISCO
2	AVDA. JUAN CARLOS I

PRESTAMOS		
NumPrestamo	Cantidad	IdSucursal
1000	2345000	1
1001	4000	2

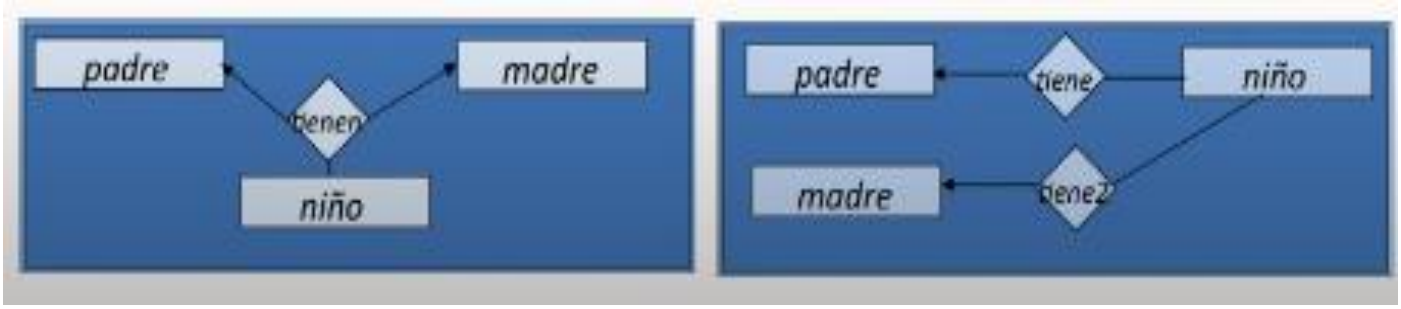
CLIENTES	
Dni	Nombre
1111111B	PABLO LOPEZ
2222222C	GERMÁN SÁNCHEZ
3333333D	LOLA FDEZ.

TIENEN	
Dni	NumPrestamo
111111B	1000
222222C	1000
333333C	1001

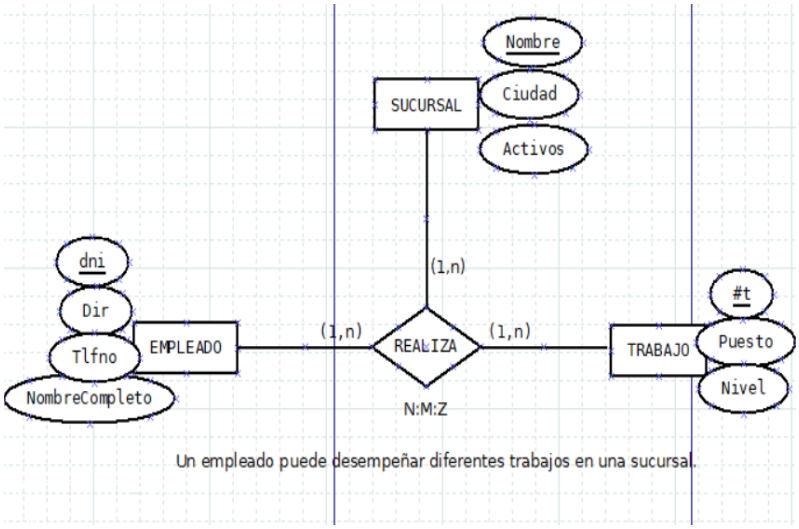
En este caso, solo tendríamos que actualizar la cantidad de la tabla PRESTAMOS.

5.3. ¿ Relación TERNARIAS ó BINARIAS ?

- ✓ Siempre que se pueda relaciones BINARIAS, a no ser que no se pueda….
- ✓ Ejemplo: un niño tiene un padre y una madre → BINARIAS



- ✓ Un empleado puede desempeñar diferentes trabajos en distintas sucursales. De forma natural, se están relacionando las tres entidades, es decir cuando la semántica sugiere un proceso donde se implican tres entidades, se ponen tres:



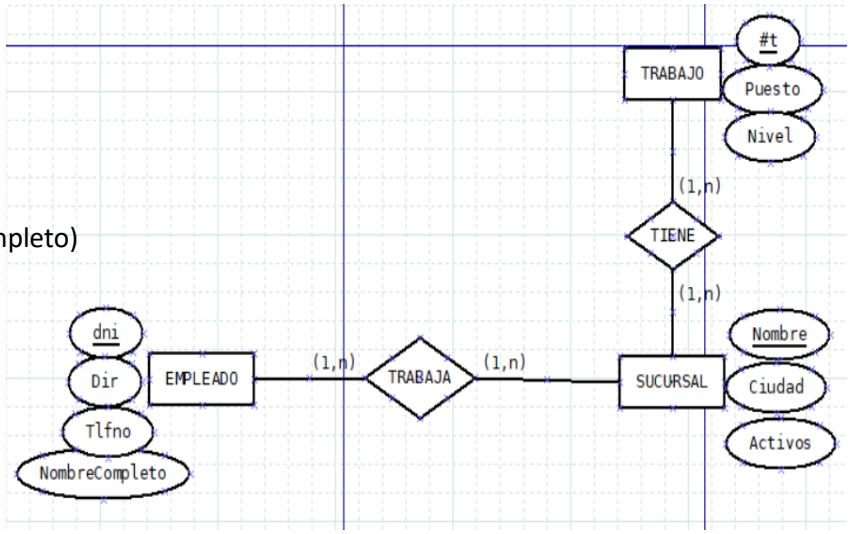
- ✓ Pasamos al Relacional:

SUCURSALES (Nombre, Ciudad, Activos)
PK: Nombre.
TRABAJOS(#t, puesto, nivel)
PK: #t.
EMPLEADOS(dni, Dir, Tlfno, NombreCompleto)
PK: Dni
REALIZA(Nombre,#t,dni)
PK: (Nombre, #t, dni)
FK: Nombre → SUCURSALES
Dni → EMPLEADOS
#t → TRABAJOS

El mismo diseño con binarias sería este:

- ✓ **Pasamos al Relacional:**

TRABAJOS(#t, puesto, nivel)
PK: #t.
SUCURSALES(Nombre, Ciudad, Activos)
PK: Nombre
EMPLEADOS(dni, Dir, Tlfno, NombreCompleto)
PK: Dni
TRABAJA(Nombre,dni)
PK: (Nombre, dni)
FK: Nombre → SUCURSALES
Dni → EMPLEADOS
TIENE(#t,Nombre)
PK:#t, Nombre
FK: #t → TRABAJOS



Imaginemos que nos piden mostrar un listado de empleados y los trabajos que desempeñan en las diferentes sucursales. El listado debe mostrar el NombreCompleto del empleado, el Nombre de la sucursal y el puesto.

Las consultas a realizar serían las siguientes:

MODELO RELACIONAL CON BINARIAS:

```
SELECT EMPLEADOS.NombreCompleto,
        SUCURSALES.Nombre,
        TRABAJOS.puesto
FROM EMPLEADOS JOIN TRABAJA ON EMPLEADOS.dni=TRABAJA.dni
                JOIN SUCURSALES ON TRABAJA.Nombre=SUCURSALES.Nombre
                JOIN TIENE ON TIENE.Nombre=SUCURSALES.Nombre
                JOIN TRABAJOS ON TRABAJOS.#t=TIENE.#t;
```

MODELO RELACIONAL CON TERNARIAS

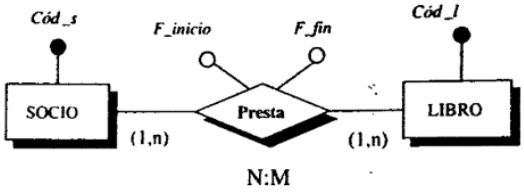
```
SELECT EMPLEADOS.NombreCompleto,
        SUCURSALES.Nombre,
        TRABAJOS.puesto
FROM EMPLEADOS JOIN REALIZA ON EMPLEADOS.dni=REALIZA.dni
                JOIN SUCURSALES ON REALIZA.Nombre=SUCURSALES.Nombre
                JOIN TRABAJOS ON TRABAJOS.#t=REALIZA.#t;
```

La consulta en el modelo relacional con ternarias requiere de un JOIN menos.

5.4. Precaución con los atributos temporales

Si aparecen **atributos fecha en una relación del diagrama Entidad-Relación**, es posible que tengas que añadir dicho atributo a la clave principal para contemplar el hecho que un suceso se puede repetir a lo largo del tiempo.

Ejemplo:



✓ Pasamos al Relacional:

SOCIOS(cod_s)
FK: cod_s
LIBROS(cod_l)
FK: cod_l
PRESTAN(cod_s, cod_l, F_inicio, F_fin)
PK: cod_s, cod_l
FK: cod_s → SOCIOS
cod_l → LIBROS

SOCIOS	
Cod_s	**Nombre**
10D	Rosa
20B	Lidia
30C	Juan
40C	Albano

LIBROS	
Cod_l	**Titulo**
100	Lo que el viento se llevó
101	La historia interminable
102	Viaje al centro de la Tierra
103	Mujercitas

PRESTAN			
Cod_S	Cod_l	F_inicio	F_fin
10D	100	1/09/2023	8/09/2023
10D	101	8/09/2023	15/09/2023
30C	103	1/09/2023	8/09/2023
20B	100	15/09/2023	22/09/2023
40C	102	10/09/2023	17/09/2023
30C	103	20/09/2023	27/09/2023

Si elegimos la Primary Key por defecto, combinación de Primary Key de ambas entidades, vemos que no se podría llevar un histórico de las veces que un mismo socio ha prestado el mismo libro. Habría entonces que incorporar la fecha de inicio, por ejemplo a la PK.

En general, para saber si una clave está bien elegida, podemos plantearnos la siguiente pregunta:

- Para una clave determinada, ¿Cuántas filas obtengo en la tabla? La respuesta siempre debe ser UNA.

En el ejemplo anterior, la Primary Key definitiva debería ser (cod_s, cod_l, F_inicio)