# UNIDAD 3. EL MODELO RELACIONAL .TRANSFORMACIÓN DEL DIAGRAMA E/R AL MODELO RELACIONAL.

## ÍNDICE

# 1.- Introducción

Ya se ha visto en la Unidad anterior lo que es un sistema gestor de bases de datos, y cuáles son sus objetivos. También se vio la arquitectura ANSI para su aplicación en la creación de las bases de datos. Hasta ahora, hemos realizado el diseño conceptual de una base de datos pero aún no hemos realizado su diseño lógico.

Esta Unidad trata del que actualmente es el principal modelo para las aplicaciones de procesamiento de datos, el **modelo relacional**, que presenta una forma muy simple y potente de representar los datos. A lo largo de la Unidad, se exponen los fundamentos del modelo de datos relacional y su aplicación para el diseño lógico de datos y de bases de datos relacionales.

## 2. El Modelo Relacional

En 1970 Codd creó el Modelo RELACIONAL, con una base matemática muy sólida (la de las relaciones), donde los datos se estructuran en forma de ***relaciones*** (tablas).

Fue a partir de los años 80, cuando la tecnología lo permitió, con la salida de mejores productos, como por ejemplo el Oracle (1979), cuando su implantación se volvió aplastante.

Objetivos del Modelo Relacional:

- **Independencia física**, para que la manera de guardar los datos no influya en su manipulación lógica.
- **Independencia lógica**, para que las vistas externas no se vean afectadas por cambios en el esquema conceptual de la B.D.
- **Flexibilidad**, para poder ofrecer los datos a cada usuario de la forma más adecuada a su aplicación.
- **Uniformidad**, las estructuras lógicas de los datos presentan un aspecto simple y uniforme (tablas).
- **Sencillez**, las características anteriores, unidas a unos lenguajes de usuario sencillos, hacen que el Modelo Relacional sea fácil de entender y de

utilizar por el usuario final.

## 3- Estructura del Modelo Relacional

Como ya se ha indicado anteriormente, la **relación** es el elemento básico del modelo relacional y se representa como una tabla, en la que se puede distinguir el *nombre de la tabla*, el conjunto de columnas que representan las propiedades de la tabla y que se les llama **atributos**, y el conjunto de filas llamadas **tuplas**, que contienen los valores que toma cada uno de los atributos para cada elemento de la relación.

En la figura que se muestra a continuación se representa una relación llamada ALUMNO en forma de tabla.

**RELACIÓN ALUMNOS**

| NUM_MAT | NOMBRE | APELLIDOS | CURSO |
|---|---|---|---|
| 5467 | JUAN | CABELLO | 1BACH-A |
| 3421 | DOLORES | GARCÍA | 1BACH-C |
| 7622 | JESÚS | SÁNCHEZ | 2BACH-C |

ATRIBUTOS

TUPLAS

A continuación, se exponen los elementos que constituyen el modelo relacional.

### 3.1. Dominios y atributos

Cada una de las propiedades o características que tiene un tipo de entidad o una relación se denomina **atributo**. Se corresponden con las columnas de la relación.

Se define **dominio** como el conjunto finito de valores homogéneos (todos del mismo tipo) y atómicos (son indivisibles), que puede tomar cada atributo.

Los valores contenidos en una columna pertenecen a un dominio que previamente se ha definido. Todos los dominios tienen un **tipo de datos** asociado.

Existen dos tipos de dominios:

– Dominios **generales**. Son aquellos cuyos valores están comprendidos entre un máximo y un mínimo. Por ejemplo, el Código-postal, que está formado por todos los números enteros positivos de 5 cifras, o el nombre de una persona, formado por las letras del alfabeto con una longitud determinada.

– Dominios **restringidos**. Son los que pertenecen a un conjunto de valores específico. Por ejemplo, Sexo, que puede tomar los valores H o M.

Por ejemplo, en la relación ALUMNO podemos considerar los siguientes atributos y dominios:

– Atributo NUM-MAT, dominio: conjunto de enteros formados por 4 dígitos.

– Atributo NOMBRE, dominio: conjunto de 15 caracteres.

– Atributo APELLIDOS, dominio: conjunto de 20 caracteres.

– Atributo CURSO, dominio: conjunto de 7 caracteres

– Atributo REPITE, dominio restringido a valores 'Si' o 'No'.

## 3.2. Relaciones

La relación se representa mediante una tabla con filas y columnas.

En el modelo relacional, las relaciones se utilizan para almacenar información sobre los objetos que se representan en la base de datos. Se representa gráficamente como una **tabla bidimensional** en la que las filas corresponden a registros individuales y las columnas a los campos de información de esos registros.

La relación está formada por:

• **Atributo** (columna). Se trata de cada una de las columnas de la tabla, identificadas por un nombre. Se denomina **grado** al número de columnas de la tabla.

• **Tupla** (fila). Representa una fila de la tabla.

En la imagen más abajo aparece la tabla EMPLEADO con dos filas o tuplas y 5 atributos o columnas (tabla de grado 5).

• **Valor**. Viene representado por la intersección entre una fila y una columna. Por ejemplo, son valores de la tabla EMPLEADO: 13407877B, Milagros Suela Sarro, 1500… Se conoce como **valor Null** la ausencia de información en una de las intersecciones.

| Nº Emple | Apellidos | Salario | Numdepart | FechaAlta |
|---|---|---|---|---|
| 13407877B | Milagros Suela Sarro | 1.500 | 10 | 18/11/90 |
| 41667891C | José María Cabello | 2.000 | 20 | 29/10/92 |

**Propiedades de las relaciones**

Las relaciones tienen las siguientes características:

– Cada relación tiene un **nombre** y éste es distinto de los demás.

– Cada atributo tiene un **nombre** y éste es distinto de los demás.

– Los valores de los atributos son **atómicos**: en cada tupla cada atributo toma un solo valor.

– El **orden** de los atributos es irrelevante, no están ordenados.

– Cada **tupla** en la relación es **distinta** de las demás, no hay tuplas duplicadas, al menos deben diferir en un valor.

– Al igual que en los atributos, el **orden** de las tuplas es irrelevante, las tuplas no están ordenadas.

**Tipos de relaciones**

En un SGBD relacional pueden existir varios tipos de relaciones, aunque no todos manejan todos los tipos. Unas relaciones permanecen en la base de datos y otras son el resultado de una selección de información:

– **Relaciones base**. Son relaciones reales que tienen nombre y forman parte directa de la base de datos almacenada. Representan el nivel conceptual de la arquitectura ANSI.

– **Vistas**. Se corresponden con el nivel externo de la arquitectura ANSI. Son relaciones con nombre que se definen a partir de una consulta. No tienen datos almacenados, lo que se almacena es la definición de la consulta. Se las llama también virtuales.

– **Resultados de consultas**. Son las resultantes de las consultas de usuario. No persisten en la base de datos.

## 3.3. Claves

**En una relación no hay tuplas repetidas**. Cada tupla de una tabla tiene que estar asociada a una **clave única** que permita identificarla.

Una clave puede estar compuesta por uno o más atributos. Si es necesario utilizar más de un atributo para tener una identificación única de la tupla, no podemos descartar ninguno de los atributos que componen la clave.

Se define **clave candidata** de una relación como uno o varios atributos que identifican unívoca y mínimamente (necesarios para identificar la tupla) cada tupla de la relación. **Siempre hay al menos una clave candidata pues por definición no puede haber dos tuplas iguales.**

Según el concepto de clave única, la clave candidata puede ser **compuesta:**

| **ID-Cliente** | **ID-Pedido** | **Fecha** |
|---|---|---|
| 28123456T | 0001 | 12/12/2019 |
| 31999888S | 0001 | 10/01/2020 |
| 28123456T | 0002 | 12/02/2020 |
| 52333555H | 0001 | 13/02/2020 |

(ID_cliente+ID_Pedido), en relación de pedidos de una pizzería

Una relación puede tener más de una clave candidata entre las cuales se distinguen:

• **Clave primaria o principal** (**Primary Key/PK**): aquella clave candidata que el usuario escoge para identificar las tuplas de la relación. No puede tener valores nulos. Si sólo existe una clave candidata, ésta se elegirá como clave primaria.

• **Clave alternativa**: aquellas claves candidatas que no han sido escogidas como clave primaria.

Ahora introduciremos un nuevo concepto: se denomina **Clave Ajena (Foreign Key/FK)** de una relación R1 al atributo o conjunto de atributos cuyos valores han de coincidir con los valores de la clave primaria de otra relación R2. Ambas claves estarán definidas sobre el mismo dominio y son muy importantes en el estudio de la integridad de datos del modelo relacional.

**Caso práctico**

Disponemos de las tablas T-DEPART y T-EMPLE.

- Las **columnas** de la tabla T-DEPART son: Nº. de departamento (NUM-DEPT), Nombre de departamento (NOM-DEPT), Presupuesto (PRESUPUESTO).

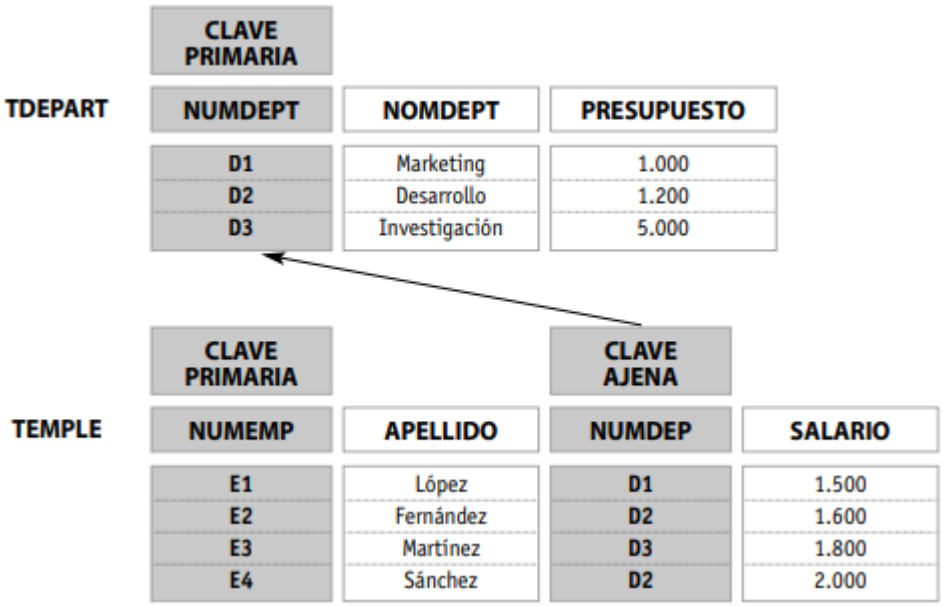
- Las **claves candidatas** son Nº. de departamento y Nombre de departamento, pues son únicos y no se van a repetir.

- Podemos escoger como **clave primaria** el Nº. de departamento. Será preferible normalmente elegir como claves primarias los campos de tipo numérico o de tipo código que identifiquen de manera única las tuplas, siempre que sea posible.

- Las **columnas** de la tabla T-EMPLE son: Nº. de empleado (NUMEMP), Apellido (APELLIDO), Nº. de departamento (NUM-DEP), Salario (SALARIO).

- La **clave candidata** es el Nº. de empleado. El apellido se puede repetir, así que no se la considera candidata. Como sólo hay una se escoge **primaria** el Nº. de empleado.

- El atributo Nº. de departamento es **clave ajena**; relaciona las tablas T-EMPLE y T-DEPART. La Figura siguiente muestra el contenido de las tablas T-EMPLE y T-DEPART con sus claves primarias y ajenas.




El nombre del atributo que es clave ajena en una relación R2, no tiene por qué coincidir con el nombre del atributo que es clave primaria en la relación R1. Sin embargo, sí es necesario que ambos estén definidos con el mismo dominio (mismo tipo de datos y longitud).

## 3.4. Esquema de la base de datos

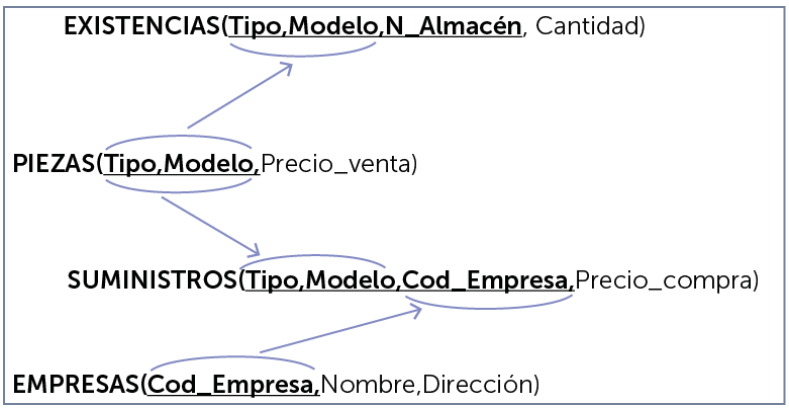
Para representar un esquema de una base de datos relacional se debe dar el nombre de sus relaciones, los atributos de éstas, las claves primarias y las claves ajenas. Posteriormente veremos cómo completar el esquema relacional con los dominios sobre los que se definen estos atributos y las restricciones que necesitemos definir.

En el esquema los nombres de las relaciones aparecen seguidos de los nombres de los atributos encerrados entre paréntesis. Las claves primarias son los atributos subrayados, y las claves ajenas se representan mediante diagramas referenciales.

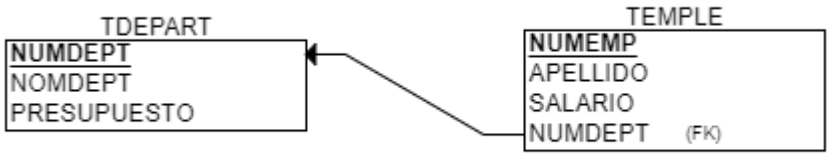
El esquema de la base de datos de empleados y departamentos es el siguiente:

| | |
|---|---|
| TDEPART | (NUMDEPT, NOMDEPT, PRESUPUESTO) |
| TEMPLE | (NUMEMP, APELLIDO, NUMDEP, SALARIO) |
| TEMPLE $\xrightarrow{\text{NUMDEP}}$ TEDEPART: | Departamento al que pertenece el empleado |

Otro ejemplo, utilizando el llamado **grafo relacional**:




También podemos utilizar los diagramas de Bachmann para representar el esquema lógico de una base de datos. Por ejemplo, utilizando la herramienta online ERDPLUS u otra similar:




*Practica: Utilizando ERDPLUS u otra herramienta online, representa el esquema relacional del grafo descrito unas líneas más arriba.*

**Ejercicios**: En Classroom tienes una hoja de ejercicios de esquemas relacionales para que practiques su diseño con las herramientas online.

## 4.- Restricciones

En todos los modelos de datos existen **restricciones** que a la hora de diseñar una base de datos se tienen que tener en cuenta. Los datos almacenados en la base de datos han de adaptarse a las estructuras impuestas por el modelo y deben cumplir una serie de reglas para garantizar que son correctos. El modelo relacional impone dos tipos de restricciones.

***Tipos** de restricciones*:

- **Inherentes**: impuestas por el propio modelo.
- **De usuario** (semánticas): en las cuales es el usuario quien prohíbe o restringe, porque el modelo se lo permite en determinadas circunstancias.

## 4.1.- Restricciones Inherentes

Impuestas por el propio modelo. Son aquellas que no son determinadas por los usuarios, sino que son definidas por el hecho de que la base de datos sea relacional. Se comentaron previamente en la unidad al hablar de las propiedades de las relaciones:

- No puede haber dos tuplas iguales
- El orden de las tuplas no es significativo
- No puede haber dos atributos con el mismo nombre
- El orden de los atributos no es significativo
- Cada atributo sólo puede tomar un valor en la tupla y dominio en el que está inscrito (valor atómico)

## 4.2.- Restricciones de Usuario (Semánticas)

Representan la semántica del mundo real. Éstas hacen que las ocurrencias de los esquemas de la base de datos sean válidos. He aquí las más importantes:

a) La **restricción de clave primaria** (PRIMARY KEY) permite declarar uno o varios atributos como clave primaria de una relación.

b) La **restricción de unicidad** (UNIQUE) impide que los valores de los atributos marcados de esa forma, puedan repetirse (con ello pueden crearse claves alternativas).

c) La **restricción de obligatoriedad** (NOT NULL) permite declarar si uno o varios atributos no pueden tomar valores nulos.

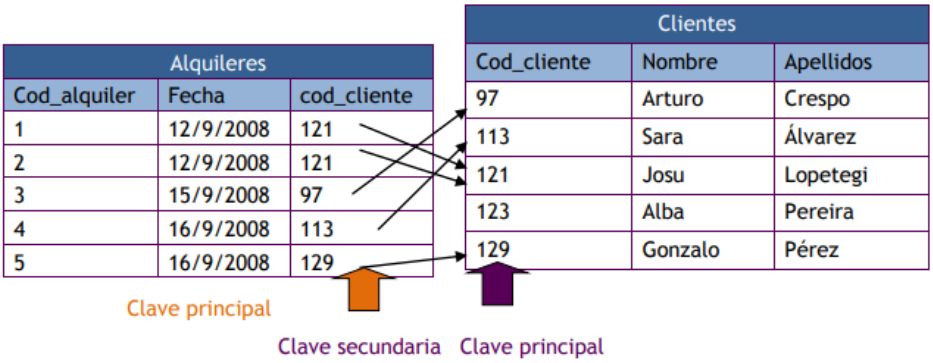
d) La **restricción de clave ajena** (FOREIGN KEY). Las claves ajenas, como hemos visto, enlazan relaciones de una base de datos.

**Integridad referencial**

La integridad referencial garantiza que los valores de la clave ajena en la relación “hijo” se corresponden con los de la clave primaria en la relación “padre”.

Respecto a la integridad referencial, si por ejemplo hay una tabla de alquileres en la que cada fila es un alquiler, existirá un atributo cod_cliente que indicará el código del cliente y que estará relacionado con una tabla de clientes, en la que dicho atributo es la clave principal. De hecho no se podrá incluir un código que no esté en la tabla clientes; eso es lo que prohíbe la integridad

referencial.




Pero, ¿qué ocurre si borramos o modificamos valores de códigos de cliente en la relación padre? Podríamos causar incoherencias si, por ejemplo, quedasen filas en la tabla secundaria con la clave externa haciendo referencia a un valor que ya no existe en la tabla principal.

Para solventar esta situación, cuando definamos la restricción de clave ajena **en el diseño físico** (lo veremos más adelante) se puede hacer uso de estas opciones:

1) Prohibir la operación.
2) Transmitir la operación en cascada. Es decir si se modifica o borra un cliente; también se modificarán o borrarán los alquileres relacionados con él.
3) Colocar nulos (set null) en las referencias de la tabla secundaria (si en la tabla secundaria la columna no está actuando como clave principal). Es decir, quedarían alquileres sin cliente.
4) Usar el valor por defecto (default). Se colocan un valor por defecto en las claves externas relacionadas, por ejemplo, el valor 999. Este valor se indica al crear la tabla (opción default).

e) **Restricción de Valor por Defecto** (DEFAULT). Permite que cuando se inserte una tupla o registro en una tabla, para aquellos atributos para los cuales no se indique un valor concreto se les asigne un valor por defecto.

f) **Restricciones o reglas de validación** (CHECK) Condición lógica que debe de cumplir un dato concreto para darlo por válido. Por ejemplo restringir el campo sueldo para que siempre sea mayor de 1000, sería una regla de validación.

g) **Disparadores o triggers**. Se trata de pequeños programas grabados en la base de datos que se ejecutan automáticamente cuando se cumple una determinada condición. Sirven para realizar una serie de acciones cuando ocurre un determinado evento (cuando se añade una tupla, cuando se borra un dato, cuando un usuario abre una conexión…) Los triggers permiten realizar restricciones muy potentes; pero son las más difíciles de crear.

**Las restricciones e) f) y g) las podremos en práctica en el diseño físico de la base de datos.**

*Practica: Para poner en práctica las restricciones, puedes modificar los esquemas relacionales que diseñaste, añadiendo los dominios de los atributos y sus restricciones.*

## 5.- Transformación Diagrama E/R a Modelo Relacional.

El proceso de diseño de bases de datos seguido en el curso nos ha llevado, como primer paso, a la realización del **diseño conceptual** de nuestra base de datos utilizando el Diagrama Entidad/Relación.

Ahora vamos a continuar en nuestro proceso de diseño realizando el **diseño lógico**, es decir, eligiendo un modelo lógico de bases de datos para implementar nuestro diseño conceptual. Como imaginarás, el modelo lógico elegido es el **modelo relacional**, que hemos analizado en profundidad en la unidad anterior.

Para realizar la transformación de nuestro diagrama en las estructuras básicas del modelo relacional, las tablas, seguiremos una serie de reglas

generales que establecen cómo los diferentes tipos de relaciones se convierten en tablas o relaciones de nuestro esquema. Analizaremos uno a uno todos los elementos del diagrama E/R observando cómo se transforma cada uno de ellos.

Para las explicaciones se utilizará la notación clásica de tabla, con lista de atributos entre paréntesis, pero puede utilizarse cualquier herramienta gráfica, como ERDplus, para representar este diseño lógico.

## 5.1.- Entidades.

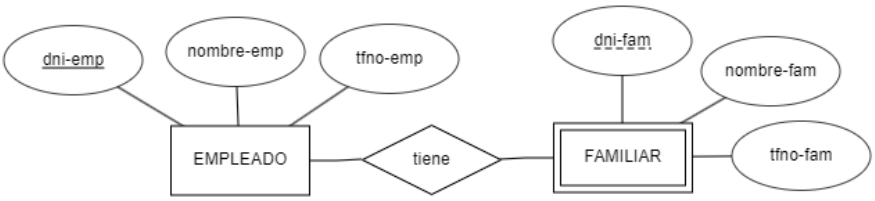
**Entidades Normales:**

*Toda entidad normal se transformará en una tabla*, con todos sus atributos, que se consideran como simples. Se tendría uno o varios como clave principal, y lo denotaremos subrayándolo.

EMPLEADO (Dni, Nombre, Dirección, Teléfono, Sueldo, Fecha-n)
DEPARTAMENTO (Num-d, Nom-d)
PROYECTO (Num-p, Nom-p)

**Entidades Débiles (diagrama E/R extendido):**

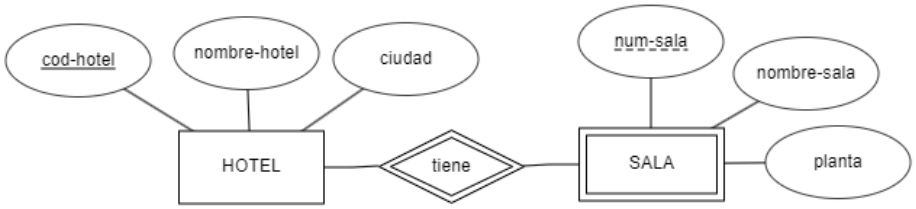
Toda entidad débil que depende de otra, se transformará en una tabla con todos sus atributos, y se añade el o los atributos de la clave principal de la entidad fuerte. Este atributo será **clave ajena** en la tabla de la débil.



EMPLEADO (**dni-emp**, nombre-emp, tfno-emp)
FAMILIAR (**dni-fam**, nombre-fam, tfno-fam, dni-emp (FK))

**(*)** Véase además el punto 5.7 sobre restricciones no plasmables en el diseño lógico.

Si además, la dependencia lo es en Identificación, la clave externa formará parte de la clave principal, creándose así una **clave compuesta**, y aplicándosele además la restricción de **clave ajena** a la parte de la clave que proviene de la entidad fuerte.



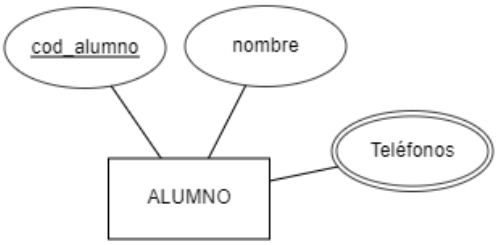

HOTEL (**cod-hotel**, nombre-hotel, ciudad)
SALA (**cod-hotel (FK), num-sala**, nombre-sala, planta)

## 5.2.- Atributos multivaluados y compuestos

Por cada **atributo multivaluado** (ya sea de entidad o de relación) se crea una nueva tabla, a la que podemos llamar con el nombre del atributo. Dicha tabla tendrá **dos** atributos:

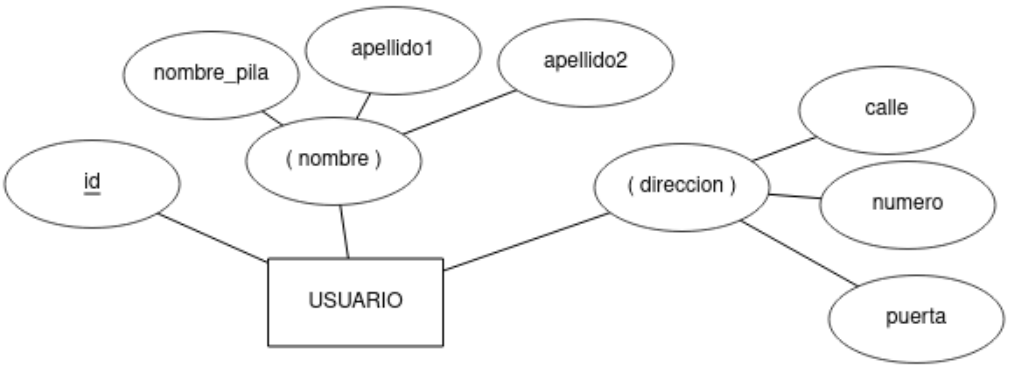
– La clave de la entidad a la que pertenece, que actuará como clave ajena
– Un nuevo atributo que representará el atributo multivaluado.

La **clave primaria** de esta nueva tabla será **la clave compuesta** formada por sus dos atributos.




ALUMNOS (**cod_alumno**, nombre)
TELÉFONOS (**cod_alumno(FK), teléfono**)

Si tuviéramos algún atributo **compuesto** como los que se muestran en la figura:




tomaremos directamente los atributos componentes:

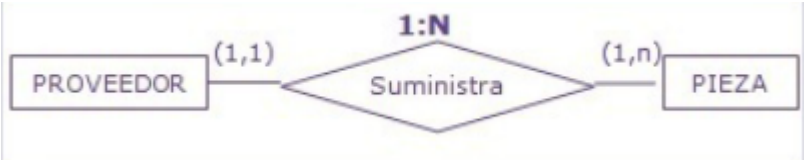
USUARIO **(id**, nombre_pila, apellido1, apellido2, calle, numero, puerta)

## 5.3.- Relaciones

### Relaciones 1:N

A la entidad del lado N de la relación se le añade la clave primaria de la entidad del lado 1, creando una **clave ajena**, de modo que se puedan relacionar

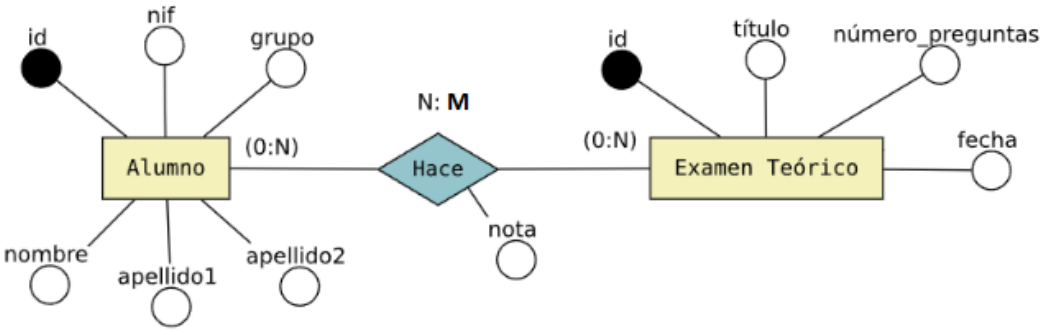
ambas tablas mediante operadores relacionales. El nombre de la relación desaparece.




PROVEEDOR (**id-proveedor**, nombre)
PIEZA **(id-pieza**, nombre-pieza, id-proveedor (FK))

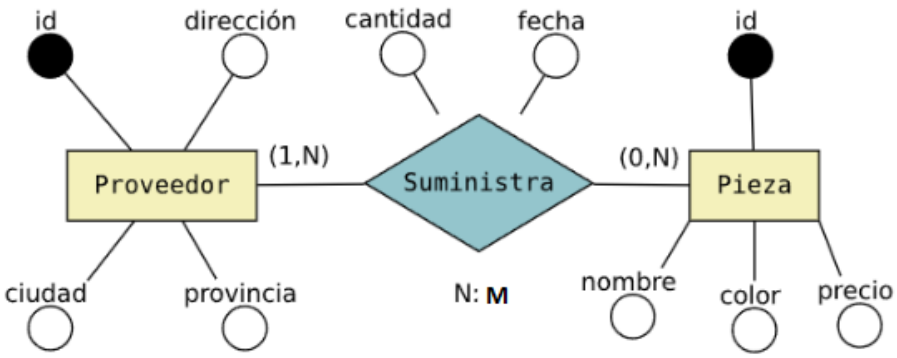
**Relaciones N:M**

Se crea una nueva tabla que tendrá como clave primaria la **concatenación de los atributos clave de las entidades que asocia** y con los atributos propios de la relación si los hay. Los componentes de la clave primaria son a su vez **clave ajena** de sus respectivas tablas.




- ALUMNO(**id**, nombre, apellido1, apellido2, nif, grupo)
- EXAMEN_TEÓRICO**(id**, título, número_preguntas, fecha)
- HACE (**id_alumno (FK), id_examen (FK)**, nota)

Pero ahora veamos este ejemplo:




Las reglas de transformación de E/R al modelo relacional nos dicen que la relación **Suministra** genera, además de una tabla por cada entidad, una nueva tabla porque es una relación de cardinalidad **N:M**. Esta nueva tabla recibe las claves primarias de las dos entidades que participan en la relación y además participan como **clave ajena**. La solución teórica sería la siguiente:

- PROVEEDOR **(id**, dirección, ciudad, provincia)
- PIEZA (**id**, nombre, color, precio)
- SUMINISTRA **(id_proveedor (FK), id_pieza (FK)**, fecha, cantidad)

Pero con esta solución podemos tener un problema en el caso de que un proveedor nos suministre piezas con el mismo id en fechas diferentes. En este caso no podríamos almacenar esta información en la tabla porque se produciría un error de claves primarias duplicadas.

| **#id_proveedor** | **#id_pieza** | fecha | cantidad |
|---|---|---|---|
| 1 | 1 | 01/01/2018 | 100 |
| 1 | 1 | 20/01/2018 | 100 |

Para solucionarlo podemos incluir el atributo *fecha* como parte de la clave primaria de la tabla, de modo que la clave primaria estaría compuesta por id_proveedor, id_pieza y fecha. La solución sería la siguiente:

- SUMINISTRA **(id_proveedor (FK), id_pieza (FK)**, **fecha**, cantidad)

En este caso ya no habría ningún problema para almacenar que un proveedor nos suministra piezas con el mismo id en fechas diferentes.

| **#id_proveedor** | **#id_pieza** | **#fecha** | cantidad |
|---|---|---|---|
| 1 | 1 | 01/01/2018 | 100 |
| 1 | 1 | 20/01/2018 | 100 |

Si fuese necesario registrar que el mismo proveedor puede suministrar piezas con el mismo código en diferentes horas del mismo día, habría que reemplazar la columna fecha por fecha_hora, por ejemplo, de manera que se evitasen duplicados.

| **#id_proveedor** | **#id_pieza** | **#fecha_hora** | cantidad |
|---|---|---|---|
| 1 | 1 | 01/01/2018 08:00:00 | 100 |
| 1 | 1 | 20/01/2018 10:00:00 | 100 |
| 1 | 1 | 20/01/2018 17:00:00 | 100 |

Otra forma de resolver este problema puede ser creando un nuevo atributo **id** que sea un valor numérico con autoincremento y que éste sea la única clave primaria de la tabla. La solución sería la siguiente:

- SUMINISTRA **(id_suministro,** id_proveedor (FK), id_pieza (FK), fecha, cantidad)

En este caso tampoco habría ningún problema para almacenar que un proveedor nos suministra piezas con el mismo id en fechas diferentes.

| **#id** | id_proveedor | id_pieza | fecha_hora | cantidad |
|---|---|---|---|---|
| 1 | 1 | 1 | 1/01/2018 08:00:00 | 100 |
| 2 | 1 | 1 | 20/01/2018 10:00:00 | 100 |
| 3 | 1 | 1 | 20/01/2018 17:00:00 | 100 |

*ACTIVIDAD: Realiza la transformación de estos otros ejemplos:*

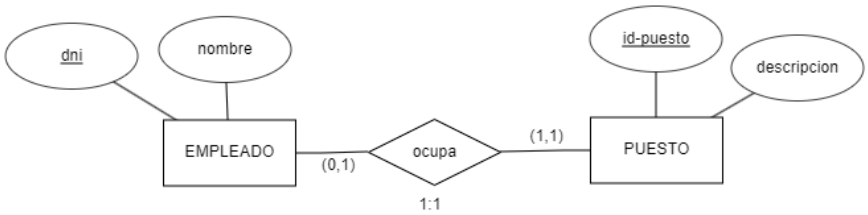
*EMPLEADO trabaja en PROYECTO (tipo N:M, además con un atributo de relación: Num_Horas)*

*CLIENTE alquila VEHÍCULO (tipo N:M, además con un atributo de relación: Mes)*

## Relaciones 1:1

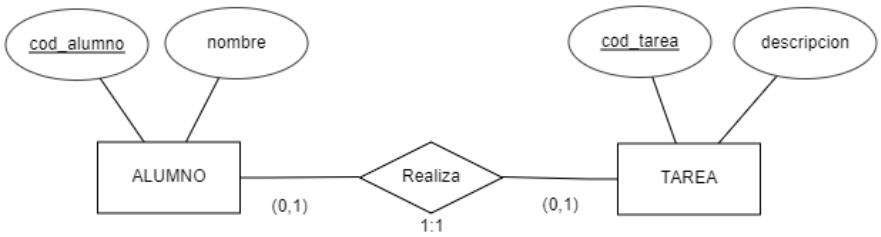
Se transforman en función de las cardinalidades:

- Cardinalidades individuales (0,1) y (1,1), Crear una tabla para cada entidad y propagar la clave de la entidad con cardinalidad (1,1) a la entidad que tenga (0,1).




EMPLEADO (**dni**, nombre, id-puesto(FK))
PUESTO **(id-puesto,** descri)

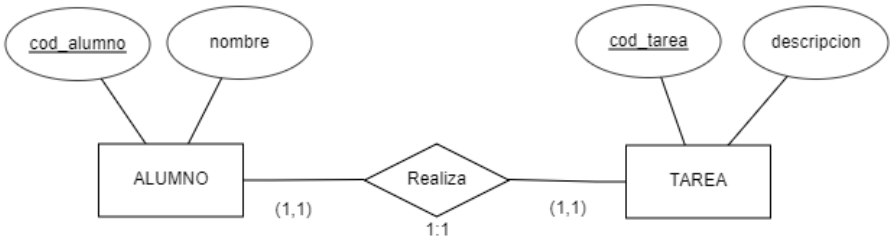
- Cuando ambas tablas tienen cardinalidades (0,1), además de crear una tabla para cada entidad, se creará una nueva tabla a partir de la relación con las dos claves de ambas, para evitar los valores nulos, más los atributos de la relación, si los hubiese:




ALUMNO (**cod-alumno**, nombre)
TAREA (**cod-tarea**, descripcion)
TRABAJO (**cod-alumno (FK), cod-tarea (FK)**, + atributos de la relación)

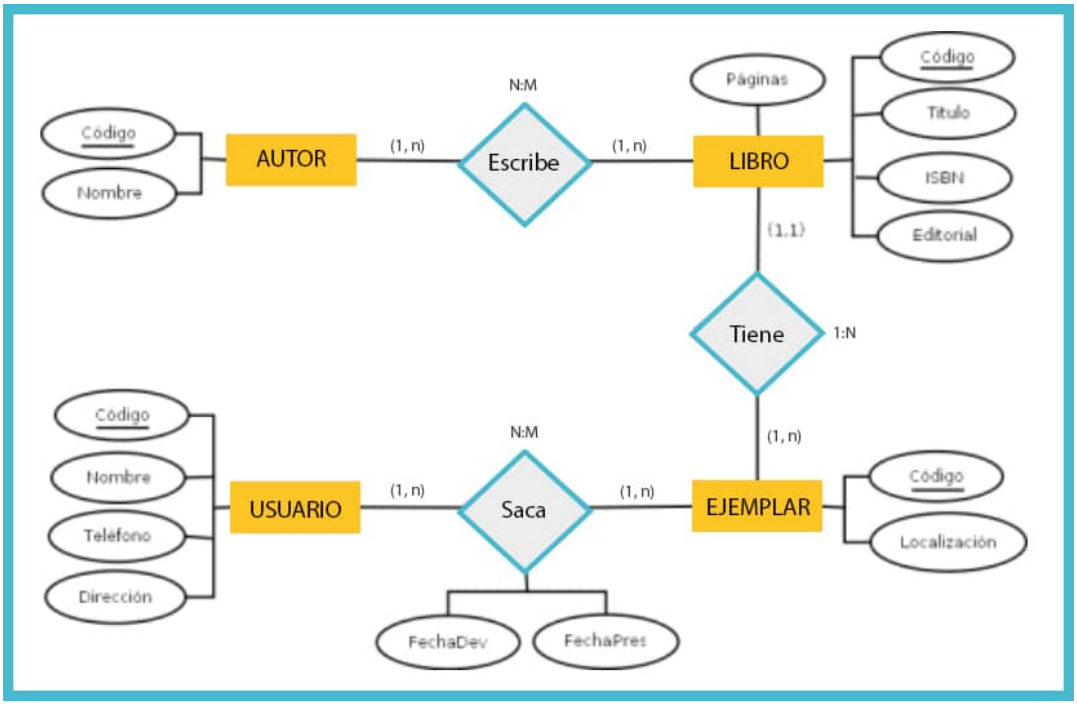
- Cuando ambas entidades participan con cardinalidades (1,1), propagaremos la clave de cualquiera de ellas a la otra tabla como **clave**

**ajena**, generalmente llevando esta clave ajena a la tabla a la que se le efectúan los accesos más frecuentes.




ALUMNO (**cod-alumno**, nombre, cod_tarea(FK))
TAREA (**cod-tarea**, descripcion)

**Practica con este ejemplo:**

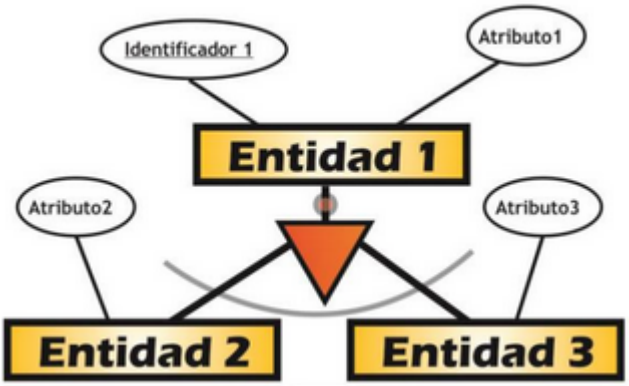


## 5.4.- Generalizaciones/Especializaciones

A tener en cuenta: **Tanto las superentidades como las subentidades generarán tablas en el modelo relacional.** Los atributos se colocan en la tabla de la entidad correspondiente.

Además:

Para las subentidades se colocará **como clave primaria** la clave de su superentidad. Asimismo, dicha clave primaria tendrá la restricción de **clave ajena,** o, en otras palabras, cuando agreguemos información en la tabla de la subentidad, la clave utilizada debe existir previamente en la tabla de la superentidad.

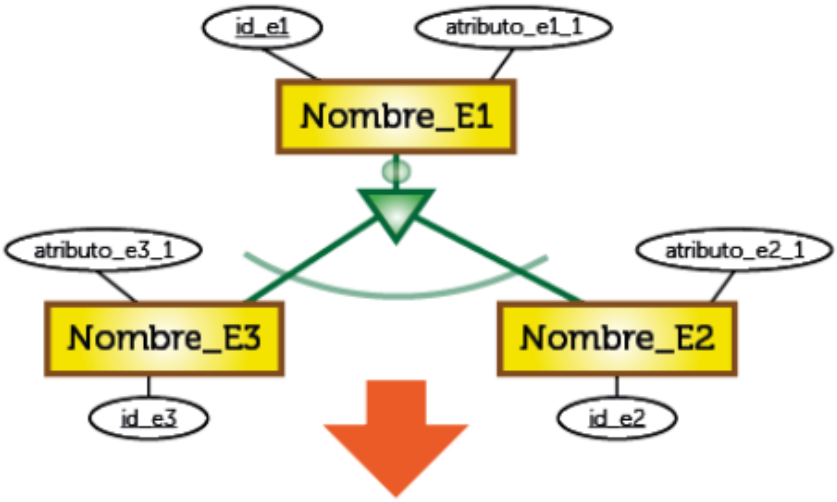



ENTIDAD_1 (**Identificador_1**, atributo1)
ENTIDAD_2 (**Identificador_1 (FK)**, atributo2)
ENTIDAD_3 (**Identificador_1 (FK)**, atributo3)

La forma de proceder es la misma si la relación de especialización lo es en **exclusividad** o **sin exclusividad**.

**(*)** Véase además el punto 5.7 sobre restricciones no plasmables en el diseño lógico.

Si alguna de las entidades inferiores tuviera su clave primaria propia, la clave de la superentidad pasaría a la tabla de la subentidad como **clave ajena**:




CLIENTE (Cod-cliente, Nombre)
SOCIO (Num-Socio, Descuento, Cod-cliente)
NO-SOCIO (Num-Solicitud, Fecha-Solic, Cod-cliente)

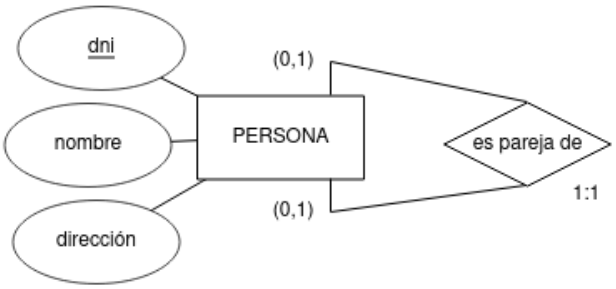
Nombre_E1 (**id_e1**,atributo_e1_1)
Nombre_E2 (**id_e2**,atributo_e2_1,id_e1(FK))
Nombre_E3 (**id_e3**,atributo_e3_1,id_e1(FK))

## 5.5.- Relaciones reflexivas.

Aquí tenemos tres posibilidades: relación 1:1, 1:N y N:M

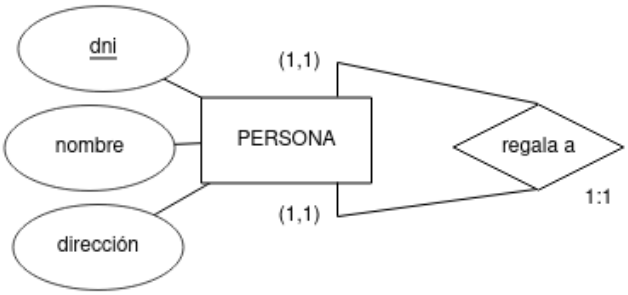
**Caso 1:1**

Tendremos la misma casuística que con las relaciones binarias 1:1. Sólo crearemos **una nueva tabla en caso que las cardinalidades individuales sean (0,1) y (0,1),** creando en ella una clave compuesta con un duplicado con diferentes nombres del atributo clave de la relación.



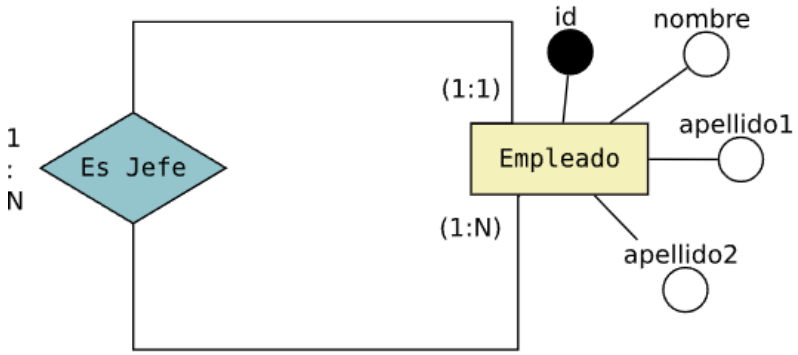

PERSONA (**dni**, nombre, direccion)
PAREJAS (**dni_1, dni_2**)

Si la relación es de (0,1) a (1,1), caso muy raro, o de (1:1) a (1:1) agregamos en la misma tabla un duplicado del atributo clave que **referenciará** como clave ajena a la clave primaria. Un ejemplo con una relación de regalos en el “amigo invisible”:



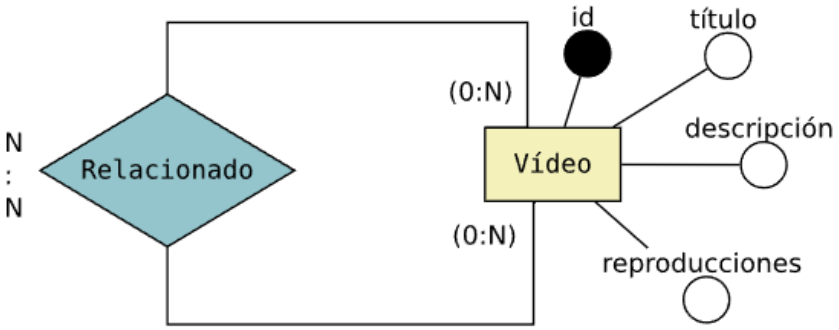

PERSONA (**dni**, nombre, dirección, regala_a_dni *(FK)*)

**Caso 1:N**




- EMPLEADO(**id**, nombre, apellido1, apellido2, *id_jefe (FK)*)

**Caso N:M**



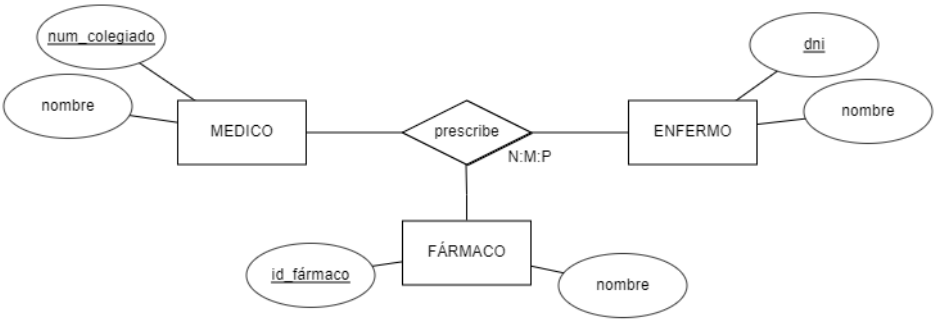

VÍDEO(**id**, título, descripción, reproducciones)
VÍDEOS_RELACIONADOS (**id_video (FK), id_video_relacionado (FK)**)

## 5.6.- Relaciones Ternarias.

En una relación ternaria, se generará, además de las tablas propias de las entidades participantes, **una nueva tabla desde el momento en que se combinen al menos 2 entidades con una cardinalidad múltiple (es decir, en los casos 1:N:M y N:M:P)**. Veamos cada caso por separado.

**Caso N:M:P**

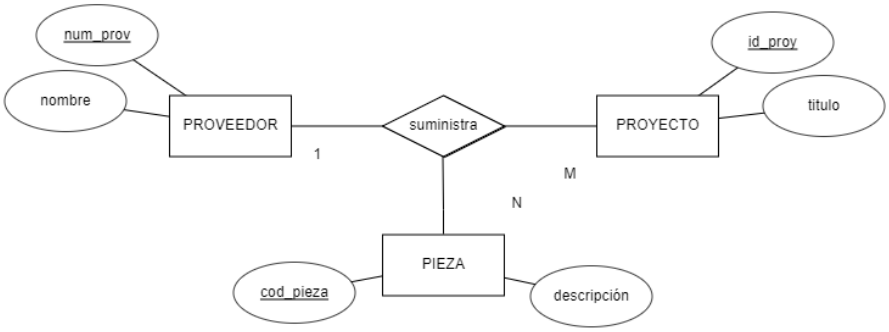
Se crea una nueva tabla con las claves primarias de las 3 tablas (y atributos de la relación, si los hubiera). Todas las claves agregadas funcionarán como una **clave primaria compuesta** de las 3.




MÉDICO (**num_colegiado**, nombre)
ENFERMO (**dni**, descripción)
FÁRMACO (**id_fármaco**, nombre)
SUPERVISA (**num_colegiado(FK), dni(FK), id_fármaco(FK)**)

**Caso 1:N:M**

Se crea una nueva tabla con las claves primarias de las 2 tablas pertenecientes a las entidades con cardinalidad múltiple actuando como clave primaria compuesta, más la clave de la otra entidad, sólo como clave ajena.

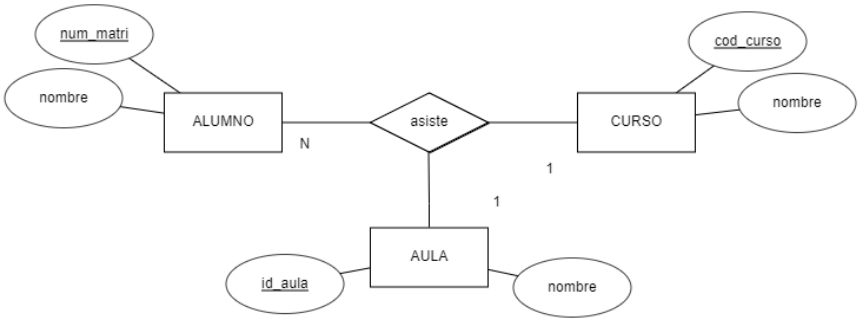



PROVEEDOR (**num_prov**, nombre)
PROYECTO (**id_proy**, titulo)
PIEZA (**cod_pieza**, descripción)
SUMINISTRA (**cod_pieza (FK), id_proy (FK),** num_prov(FK)**)**

Para los 3 casos, sólo **dos** de las claves agregadas funcionarán como una clave primaria compuesta. En todos los casos siempre formarán parte de la clave seleccionada aquella o aquellas que correspondan a entidades del lado de una cardinalidad N:

**Caso 1:1:N**

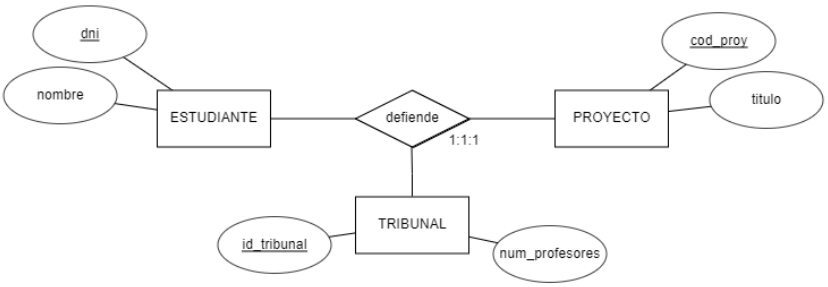
No es necesario crear una nueva tabla. En la tabla de la entidad de cardinalidad N se agregan las claves primarias de las otras dos entidades como claves ajenas (sin marcar como claves primarias).




ALUMNO (**num_matri**, nombre, cod_curso (FK), id_aula (FK))
CURSO ( **cod_curso**, nombre)
AULA (**id_aula**, nombre)

**Caso 1:1:1**

No necesitamos crear una tabla nueva. Elegimos una de las 3 entidades, por ejemplo ESTUDIANTE, y agregamos como claves ajenas (sin marcar como primarias) las claves de las otras dos entidades:




ESTUDIANTE (**dni**, nombre,cod_proy(FK), id_tribunal(FK))
PROYECTO (**cod_proy**, descripción)
PLANTA (**id_tribunal**, nombre)

## 5.7. Restricciones que no pueden plasmarse en el diseño lógico

Algunas restricciones del diseño conceptual no pueden plasmarse en el diseño lógico, por lo que será necesario **agregar una nota aclaratoria** con dichas restricciones para que sean tomadas en cuenta en el diseño físico, por lo general, para ser implementadas mediante *triggers* (disparadores) o con operaciones en *cascada*, como veremos en la próxima unidad:

- **Especializaciones/generalizaciones exclusivas** (por ejemplo, que un profesor no puede ser conserje y viceversa en la especialización de una entidad “EMPLEADO”)
- **Cardinalidades fuera de las habituales** (por ejemplo, que en tabla ALUMNOS, para un determinado curso no se sobrepasen las 40 ocurrencias)
- **La dependencia en existencia** (no en identificación) necesitará que el diseño físico tenga en cuenta que la eliminación o la modificación de la ocurrencia de la entidad fuerte se transmita a todas las de la débil que dependan de la misma.

- **Relaciones de Exclusividad, Inclusividad, exclusión e inclusión,** que implican que una ocurrencia de una entidad, si aparece en una tabla no aparezca en otra, o que aparezca en la segunda bajo determinadas condiciones (por ejemplo, si una persona se incluye en la tabla IMPARTE-CURSO, no debe aparecer en la tabla RECIBE-CURSO, o que no aparezca en RECIBE-CURSO a menos que aparezca al menos 3 veces en IMPARTE-CURSO.
  *(**NOTA**: ESTA ÚLTIMA VARIEDAD DE RELACIONES NO FORMA PARTE DEL TEMARIO DE ESTE MÓDULO)*

Y además, cualquier otra restricción que venga especificada en el diseño conceptual y que no haya podido ser representada en el Diagrama Entidad/Relación, como por ejemplo:

- Restringir el valor de un determinado campo (por ejemplo, el atributo “Tipo” sólo puede ser “A”, “B” o “C”
- Condición que han de cumplir los campos de una determinada tabla (por ejemplo, que el valor de un atributo “Importe” no sobrepase el valor 1000).