TEMA 3.
FUNDAMENTOS DEL MODELO RELACIONAL:
Estructura de las bases de datos relacionales

1º CFGS – Desarrollo de Aplicaciones Multiplataforma

IES Alonso de Ercilla, Ocaña (Toledo)

Gaspar Panadero Blanco

Índice

1. Introducción

- En 1970 **Codd** publicó un artículo proponiendo un nuevo modelo de datos que tenía como objetivo fundamental aislar al usuario de las estructuras físicas de los datos, consiguiendo así la **independencia** de las aplicaciones respecto de los datos.
- Este objetivo fundamental es expresado explícitamente por Codd:
  - *"... se propone un modelo relacional de datos como una base para proteger a los usuarios de sistemas de datos formateados de los cambios que potencialmente pueden alterar la representación de los datos, causados por el crecimiento del banco de datos y por los cambios en los caminos de acceso"*.
- El nuevo modelo se basa en la teoría matemática de las relaciones. Los datos se estructuran lógicamente en forma de relaciones (muy parecido al concepto de tabla).

1. Introducción

- Los avances más importantes que el modelo de datos relacional incorpora respecto a los modelos de datos anteriores son:
  - **Sencillez y uniformidad**: los usuarios ven la base de datos relacional como una colección de tablas, y al ser la tabla la estructura fundamental del modelo, éste goza de una gran uniformidad, lo que unido a unos lenguajes no navegacionales y muy orientados al usuario final, da como resultado la sencillez de los sistemas relacionales.
  - **Sólida fundamentación teórica**: al estar el modelo definido con rigor matemático, el diseño y la evaluación del mismo puede realizarse por métodos sistemáticos basados en abstracciones.
  - **Independencia de la interfaz de usuario**: los lenguajes relacionales, al manipular conjuntos de registros, proporcionan una gran independencia respecto a la forma en la que los datos están almacenados.

2. Elementos básicos

- **Relación**: es la estructura básica del modelo relacional. Se representa mediante una tabla.

- **Atributo**: son las propiedades de la relación. Se representan mediante columnas en las tablas.

- **Dominio**: conjunto de valores sobre los que se define el tipo de un atributo.

- **Tupla**: ocurrencia de la relación. Se representa mediante filas dentro de las tablas.

2. Elementos básicos: dominios y atributos

- El UD (Universo del Discurso) de una BDD relacional está compuesto por un conjunto de dominios {Di} y un conjunto de relaciones {Ri} definidas sobre esos dominios.
- Un **dominio** es un conjunto homogéneo de valores identificado por un nombre.

- Un dominio puede definirse de dos formas:
  - Por **extensión** (dando sus posibles valores): Ejemplo: días de la semana = {lunes, martes, miércoles, jueves, viernes, sábado, domingo}
  - Por **intensión** (usando tipos de datos): Ejemplo: edad : entero

2. Elementos básicos: dominios y atributos

- Un **atributo** (A) es la interpretación de un determinado dominio en una relación, es decir, el "papel" que juega en la misma.
- Notación: **D = Dom(A) → D es el dominio de A**
- Un atributo y un dominio pueden llamarse igual, pero:
  - Un atributo está siempre asociado a una relación, mientras que un dominio tiene existencia propia con independencia de las relaciones.
  - Un atributo representa una propiedad de una relación.
  - Un atributo toma valores de un dominio.
  - Varios atributos distintos (de la misma o de diferentes relaciones) pueden tomar sus valores del mismo dominio.

2. Elementos básicos: relaciones

- La **relación** es el elemento fundamental del modelo relacional, y se puede representar en forma de tabla:

NOMBRE

atributo 1	atributo 2	.....	atributo n	
XXX	XXX	.....	XXX	→ tupla 1
XXX	XXX	.....	XXX	→ tupla 2
.....	.....	.....	.....	.....
XXX	XXX	.....	XXX	→ tupla m

- Pero, ¡CUIDADO!, una relación **NO** es una tabla, existen diferencias entre ambas estructuras.

2. Elementos básicos: relaciones

Comparación de la terminología relación, tabla, fichero

RELACIÓN	~	TABLA	~	FICHERO
TUPLA		FILA		REGISTRO
ATRIBUTO		COLUMNA		CAMPO
GRADO		Nº DE COLUMNAS		Nº DE CAMPOS
CARDINALIDAD		Nº DE FILAS		Nº DE REGISTROS

Modelo Relacional (teoría)	SGBD Relacionales (implementación)	Sistemas de Ficheros Clásicos

2. Elementos básicos: relaciones

- Matemáticamente, una relación definida sobre un conjunto de dominios $D_1 \ldots D_n$ (no necesariamente distintos) es un subconjunto del producto cartesiano de los n dominios, donde cada elemento de la relación (tupla) es una serie de n valores ordenados:
  - $R \subseteq D_1 \times D_2 \times \ldots \times D_n$, siendo n el grado de la relación.

- Está definición no considera el concepto de *atributo*, por eso en bases de datos se utiliza otra definición que incluye los siguientes elementos:
  1. Un **nombre** que la identifica (algunos resultados intermedios no lo necesitan).
  2. Una **cabecera de relación** que contiene n pares atributo-dominio donde toma valores el atributo. Donde n es el grado de la relación.
  3. El **cuerpo de la relación** contiene m tuplas. Cada tupla estará compuesta de n pares atributo-valor. Donde m se denomina cardinalidad de la relación.
  4. El **esquema de la relación** está formado por el nombre R de la relación (si existe) y la cabecera de la relación. $R(\{A_i:D_i\}^n_{i=1})$. Es similar al concepto de *Entidad* del modelo Entidad/Relación.
  5. El **estado r de una relación de esquema R** (se suele denominar simplemente *relación*) se representa como r(R) y está constituido por el esquema y el cuerpo de la relación:
     - r(R) = <esquema, cuerpo>

2. Elementos básicos: relaciones

- ESQUEMA DE RELACIÓN (intensión):

AUTOR (*Nombre*: ***Nombres***, *Nacionalidad*: ***Nacionalidades***, *Institución*: ***Instituciones***)

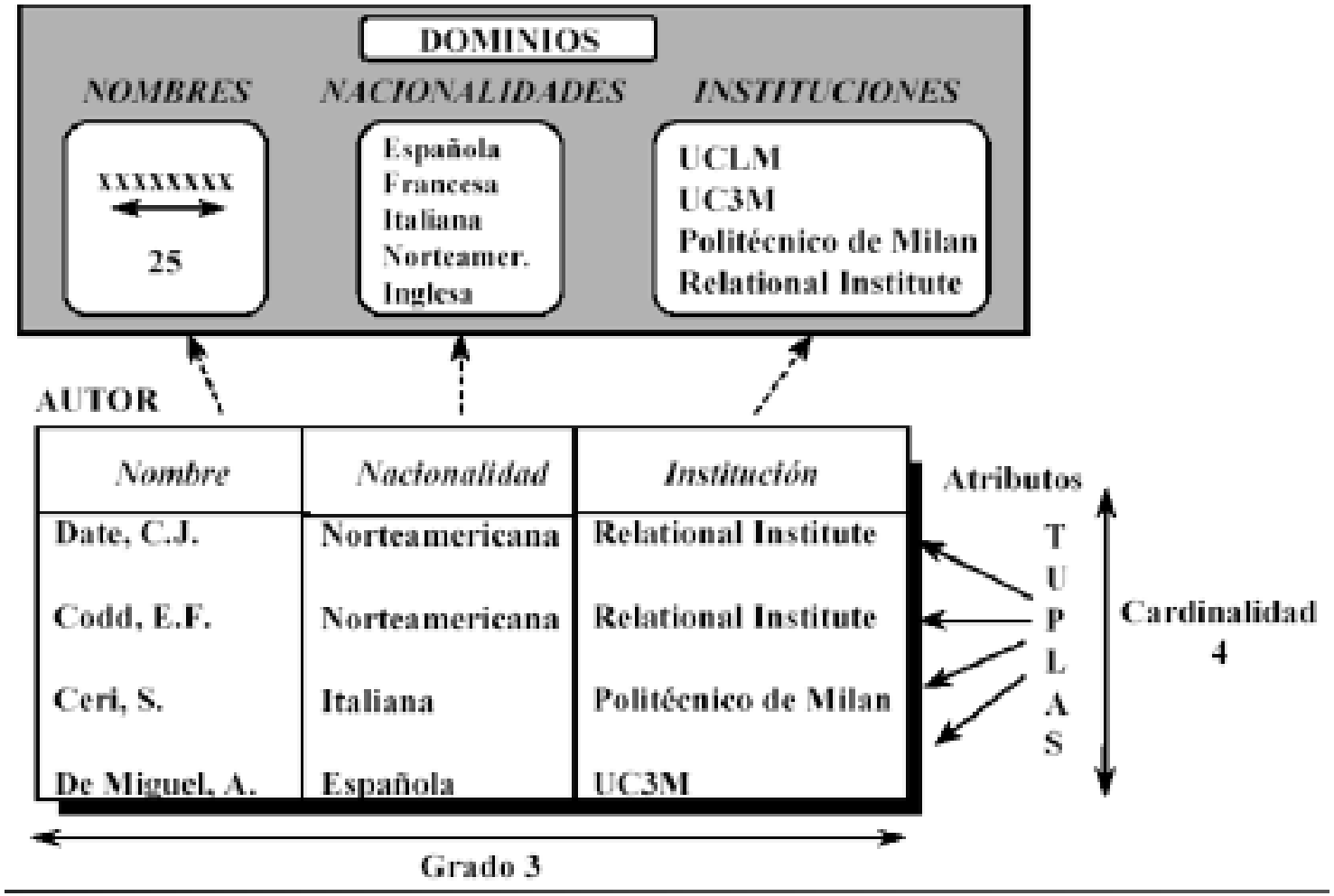
- RELACIÓN (EXTENSIÓN, ESTADO U OCURRENCIA):

AUTOR

Nombre	*Nacionalidad*	*Institución*
Date, C.J.	**Norteamericana**	**Relational Institute**
De Miguel, A.	**Española**	**UC3M**
Ceri, S.	**Italiana**	**Politécnico de Milan**

2. Elementos básicos: relaciones

- Ejemplo de relación:



2. Elementos básicos: relaciones

- Relaciones vs Tablas:

Relación (modelo teórico)	Tabla (implementación)
Tupla	Fila
Atributo	Columna
Grado	Nº de columnas
Cardinalidad	Nº de filas

Relación ≠ Tabla

3. Tipos de relaciones

- Con nombre:
  a) Persistentes:
     - Base
     - Vistas
     - Instantáneas
  b) Temporales:
     - Autónomas (o base temporales)
     - Vistas Temporales
     - Instantáneas temporales
- Sin nombre (son siempre temporales):
  a) Resultado final de una consulta
  b) Resultados intermedios de una consulta

3. Tipos de relaciones

- En el modelo relacional los datos siempre se manejan en forma de relaciones.
- **Persistentes**: su definición (esquema) permanece en la base de datos, borrándose solamente mediante una acción explícita del usuario.
  - *Relaciones base*: existen por sí mismas, no en función de otras relaciones. Se crean especificando explícitamente su esquema de relación (nombre y conjunto de pares atributo/dominio).
  - *Vistas (view):* son relaciones derivadas que se definen dando un nombre a una expresión de consulta. Lo único que se almacena es su definición en términos de otras relaciones con nombre.
  - *Instantáneas (snapshots)*: son relaciones derivadas al igual que las vistas, pero tienen datos propios almacenados, que son el resultado de ejecutar la consulta especificada. Las instantáneas no se actualizan cuando cambian los datos de las relaciones sobre las que están definidas, pero se renuevan cada cierto tiempo, de acuerdo con lo indicado por el usuario en el momento de su creación. No pueden ser actualizadas por el usuario.

3. Tipos de relaciones

- **Temporales**: a diferencia de las persistentes, una relación temporal desaparece de la BDD en un cierto momento sin necesidad de una acción de borrado específica del usuario; por ejemplo, al terminar una sesión o una transacción.

4. Restricciones

- Las **restricciones** de integridad proporcionan un medio de asegurar que las modificaciones realizadas a la base de datos no provoquen la pérdida de la consistencia de los datos.

- Tenemos dos tipos de restricciones:

    1. **Restricciones inherentes**

    2. **Restricciones semánticas**

4.1 Restricciones Inherentes

- **Restricciones inherentes**: aquellas que impone el modelo de datos al no admitir ciertas estructuras. No son definidas por el usuario.
  1. No hay dos tuplas iguales (obligatoriedad de la clave primaria).
  2. El orden de las tuplas no es significativo.
  3. El orden de los atributos no es significativo.
  4. Cada atributo sólo puede tomar un único valor del dominio sobre el que está definido; no se admiten grupos repetitivos como valores de los atributos de una tupla. (Primera forma normal)
  5. Regla de integridad de entidad: ningún atributo que forme parte de la clave primaria de una relación puede tomar el valor nulo.

4.2 Restricciones Semánticas

- **Restricciones semánticas**: también llamadas de usuario, son facilidades que el modelo ofrece a los usuarios con el fin de estos puedan reflejar en el esquema, la semántica del mundo real.
  1. Clave primaria (PRIMARY KEY)
  2. Unicidad (UNIQUE)
  3. Obligatoriedad (NOT NULL)
  4. Integridad Referencial (FOREIGN KEY)
  5. Verificación (CHECK)
  6. Aserción (ASSERTION)
  7. Disparador (TRIGGER)

4.2 Restricciones Semánticas

- **Clave primaria (PRIMARY KEY)**: de la definición de relación se deduce que siempre existe, como mínimo, un conjunto de atributos que identifican de forma unívoca cada una de las tuplas de una relación, al que se denomina **clave candidata**.
- La **clave primaria** es la clave candidata que el usuario escoge por motivos ajenos al modelo relacional.
  - Los valores del atributo/s que componen la clave primaria no pueden repetirse.
  - Los valores del atributo/s que componen la clave primaria no admiten valores nulos.

4.2 Restricciones Semánticas

- **Unicidad (UNIQUE)**: permite la definición de conjuntos de atributos cuyos valores no pueden repetirse dentro de la relación (*claves alternativas*).
  - Permite valores nulos si no se especifica lo contrario.
- **Obligatoriedad (NOT NULL)**: indica que un conjunto de atributos no admite valores nulos.
- **Integridad referencial (FOREIGN KEY)**: se llama clave foránea/ajena (*foreign key*) de una relación R2 a un conjunto de atributos cuyos valores deben coincidir con los valores de una clave candidata de una relación R1 (donde R1 y R2 no necesitan ser necesariamente distintas) o contener el valor nulo.
  - La clave candidata y la clave foránea implicadas deben estar definidas sobre el mismo dominio.

4.2 Restricciones Semánticas

- La restricción "*foreign key*" tiene las cláusulas "***on delete***" y "***on update***" que son opcionales.
- Estas cláusulas especifican cómo se debe actuar frente a eliminaciones y modificaciones de las tablas referenciadas en la restricción.
- Las opciones para estas cláusulas son las siguientes:
  - **Operación restringida (NO ACTION)**: el borrado o modificación de tuplas de la relación con la clave referenciada sólo se permite si no existen tuplas con este valor en la relación que contiene la clave foránea (opción predeterminada).
    - Dicho de otra forma, indica que si intentamos eliminar o actualizar un valor de la clave primaria de la tabla referenciada (TABLA2) que tengan referencia en la tabla principal (TABLA1), se genere un error y la acción no se realiza.

4.2 Restricciones Semánticas

- Las opciones para estas cláusulas son las siguientes:
  - **Operación con transmisión en cascada (CASCADE):** el borrado de tuplas de la relación con la clave referenciada (o la modificación de la clave) lleva consigo el borrado (o modificación) en cascada de las tuplas de la relación que contiene la clave foránea.
  - **Operación con puesta a nulos (SET NULL):** el borrado de tuplas de la relación con la clave referenciada (o la modificación de la clave) lleva consigo poner a nulos los valores de las claves foráneas de la relación que referencia.
  - **Operación con puesta a valor por defecto (SET DEFAULT):** el borrado de tuplas de la relación con la clave referenciada (o la modificación de la clave) lleva consigo poner el valor por defecto a la clave foránea de la relación que referencia.
- NOTA: los modos de borrado y modificación son independientes, es decir, cada uno tomará una de las 4 opciones por separado.

4.2 Restricciones Semánticas

- **Verificación (CHECK):** se usa para limitar el rango de valores que se puede colocar en una columna. Esta condición se comprueba siempre que se actualiza o se añade una nueva tupla. CHECK (N_HORAS > 30).
- **Aserción (ASSERTION):** funciona de forma similar a la verificación, pero en este caso la condición puede afectar a varios elementos, incluso a varias relaciones.
- **Disparador (TRIGGER):** permite que el usuario especifique cómo debe reaccionar el sistema cuando se cumpla una condición.
  - Los disparadores son procedimentales, siendo preciso que el usuario escriba el procedimiento que ha de aplicarse en caso de que se cumpla la condición.

5. De modelo E/R a modelo relacional

- **Entidades fuertes:** cada entidad fuerte se transforma en una tabla con los atributos de la entidad.
- **Entidades débiles:** cada entidad débil se transforma en una tabla con los atributos de la entidad débil más los atributos que forman la clave de la entidad fuerte de la que depende.
- **Relaciones:** dependerán de la cardinalidad de la relación:
  - ***Relaciones 1:1*** → se añade a cualquier de las dos tablas la clave de la otra tabla y los atributos de la relación, o bien se hace una sola tabla con todos los atributos.
  - ***Relaciones 1:N*** → se añaden a la tabla con participación N la clave de la tabla con participación 1 y los atributos de la relación.
  - ***Relaciones N:M*** → la relación se transforma en una tabla con los atributos clave de cada una de las entidades que participan en la relación (que formarán parte también de la clave de la nueva tabla) y los atributos de la relación.

5. De modelo E/R a modelo relacional

- **Generalizaciones y especializaciones:** existen 4 opciones:

1. Crear una tabla para la superclase y otra para cada subclase, en la que se incorporaría la clave de la superclase.
2. Crear una tabla para cada subclase con todos los atributos de la superclase y no crear una tabla para la superclase.
3. Crear solo una tabla para la superclase con todos los atributos de las subclases y añadir un campo "tipo" que permitiría identificar de qué tipo de subclase se trata. Esto solo sería válido si la relación es **exclusiva**.
4. Crear solo una tabla para la superclase con todos los atributos de las subclases y añadir un campo condicional por cada subclase que indique si es o no de esa subclase. Esto aplicaría para relaciones **solapadas**.

5. De modelo E/R a modelo relacional

- **Atributos:** por regla general, los atributos se transformarán en atributos (columnas) de la tabla a la que da lugar la entidad, pero hay 2 excepciones:

1. ***Multivaluados***: se creará una tabla con las claves de la entidad y el atributo multivaluado (todos ellos formarán la clave de la nueva tabla).

2. ***Compuestos***: se elimina el atributo compuesto y se añaden a la tabla cada uno de los atributos simples.

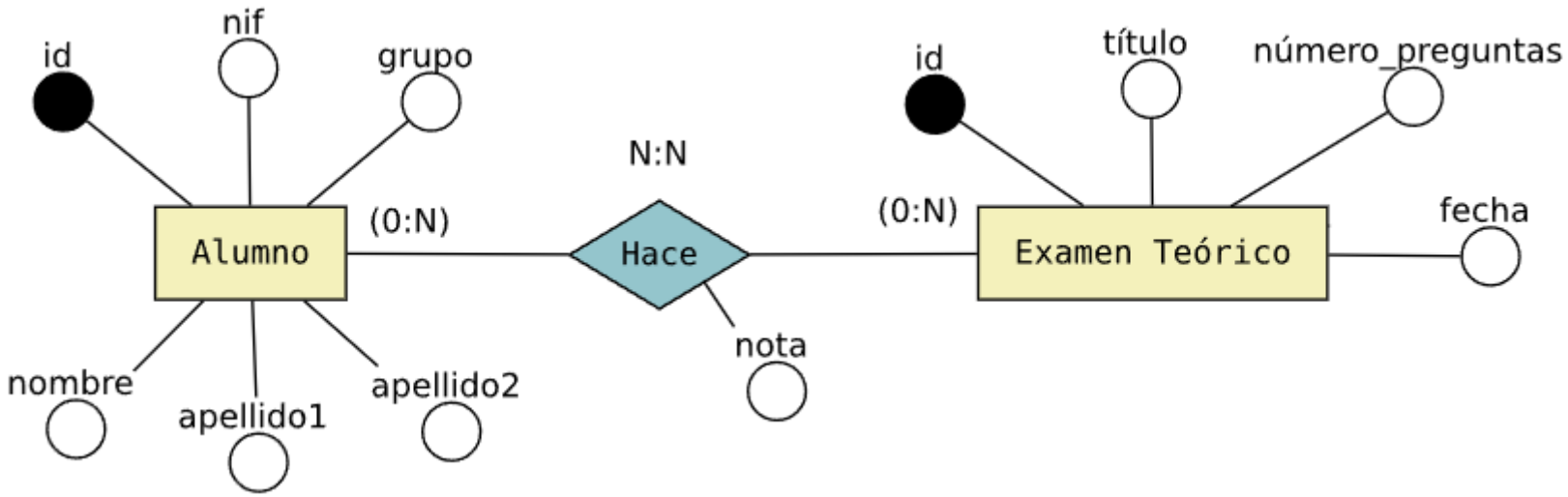
5. De modelo E/R a modelo relacional

- Para representar gráficamente el modelo relacional tenemos 2 opciones:

1. **El grafo relacional clásico**:
   - ***a)*** ***Claves primarias (PK)***: se representan subrayadas (y ponemos debajo PK).
   - ***b)*** ***Claves alternativas***: se representan con un subrayado discontinuo.
   - ***c)*** ***Claves foráneas/ajenas (FK)***: se representan con un subrayado discontinuo (y ponemos debajo FK). Además, pondremos un número de para relacionar la FK con la PK correspondiente.
   - d) Los atributos que pueden ***tomar valores nulos*** aparecen con un asterisco "*".

5.1 Entidades (fuertes y débiles)

- **Entidades fuertes**:



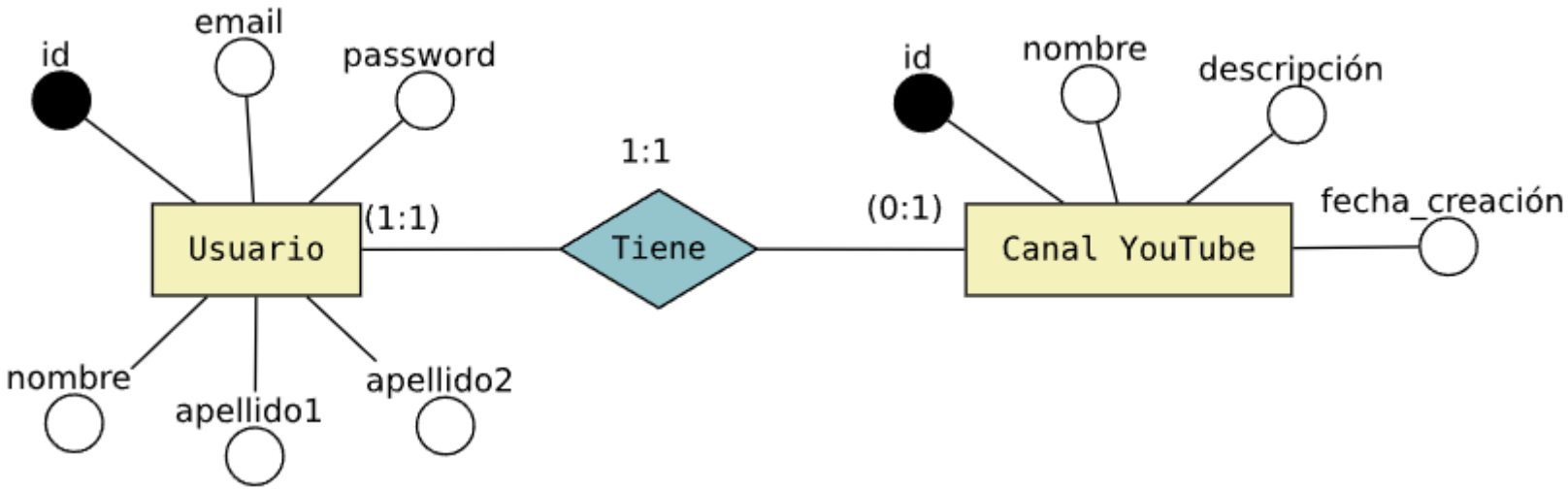
- En este ejemplo las entidades fuertes Alumno y Examen Teórico generan una tabla en el modelo relacional con las siguientes columnas.
  - ALUMNO(<u>id</u>, nombre, apellido1, apellido2, nif, grupo)
  - EXAMEN_TEÓRICO(<u>id</u>, título, número_preguntas, fecha)

5.2 Relaciones

- **Cardinalidad 1:1**:
  - Como norma general, las relaciones con cardinalidad 1:1 no generan una tabla, lo que haremos será que la clave primaria de una entidad pasará a formar parte de la tabla de la otra entidad, y pasará como un atributo (que será clave foránea/ajena con respecto a la clave primaria de la otra tabla).
  - La participación de cada una de las entidades será lo que nos ayude a decidir cuál será la entidad que pasará su clave primaria a la otra entidad.
  - ***Excepción***: Sólo existe un caso donde una relación con cardinalidad 1:1 genera una nueva tabla, y será cuando la participación de las dos entidades sea de tipo (0,1)..(0..1).

5.2 Relaciones

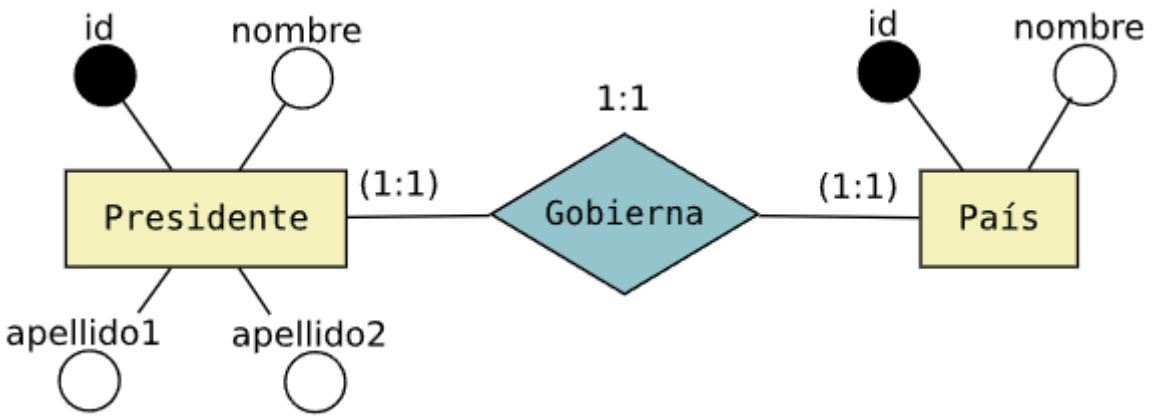
- **Cardinalidad 1:1 → *Participación (1,1)..(0,1)***:



- Como la participación de Usuario es de (1,1) y la de Canal YouTube es de (0,1), la clave primaria de Usuario se almacena en la tabla de Canal YouTube como un atributo. Se dice que el atributo id_usuario que se añade en la tabla Canal_YouTube es una clave ajena o foreign key (FK) de la tabla Usuario.
- Las tablas del modelo relacional quedarían así:
  - USUARIO(<u>id</u>, email, password, nombre, apellido1, apellido2)
  - CANAL_YOUTUBE(<u>id</u>, nombre, descripción, fecha_creación, *id_usuario*)
    - id_usuario: FOREIGN KEY de USUARIO(id)

5.2 Relaciones

- **Cardinalidad 1:1 → *Participación (1,1)..(1,1)*:**

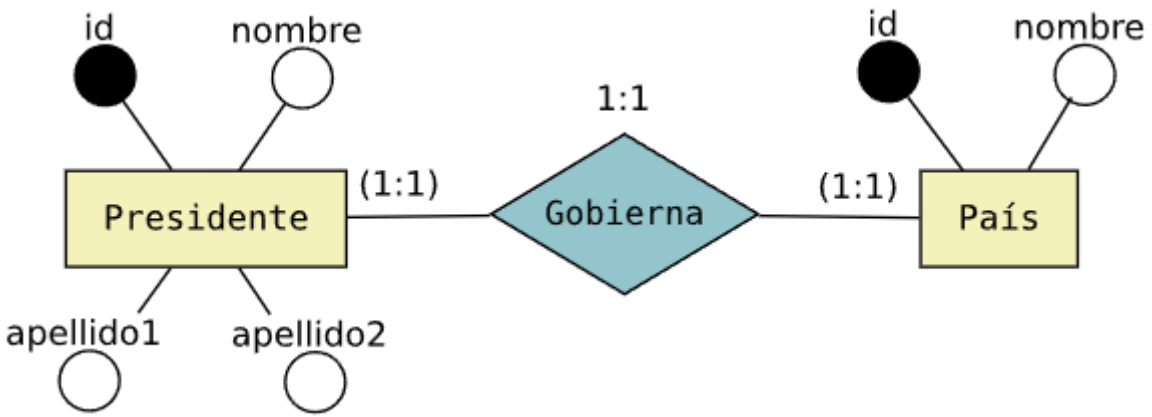


- En este caso, como la participación de las dos entidades es de (1,1) podemos resolverlo de tres formas.

1. La clave primaria de Presidente se almacena en la tabla País como un atributo (id_presidente). Se dice que id_presidente es una clave ajena o foreign key (FK) de la tabla Presidente.
   - PRESIDENTE(<u>id</u>, nombre, apellido1, apellido2)
   - PAÍS(<u>id</u>, nombre, *id_presidente*)
     - id_presidente: FOREIGN KEY de PRESIDENTE(id)

5.2 Relaciones

- **Cardinalidad 1:1 → *Participación (1,1)..(1,1)*:**

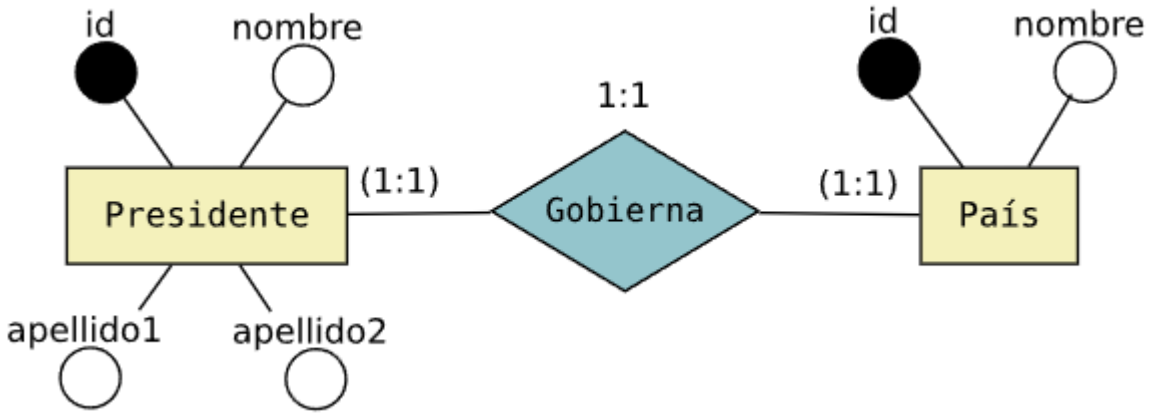


2. La clave primaria de País se almacena en la tabla Presidente como un atributo (id_pais). Se dice que id_pais es una clave ajena o foreign key (FK) de la tabla País.

   - PAÍS(<u>id</u>, nombre)
   - PRESIDENTE(<u>id</u>, nombre, apellido1, apellido2, *id_país*)
     - id_país: FOREIGN KEY de PAÍS(id)

5.2 Relaciones

- **Cardinalidad 1:1 → *Participación (1,1)..(1,1)***:

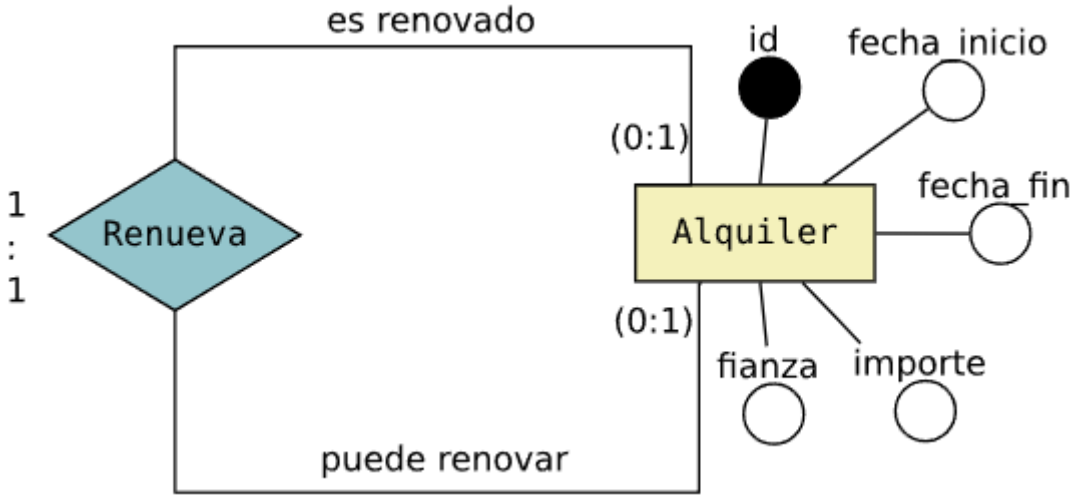


3. Las claves primarias de ambas entidades se guardan en la tabla de la otra entidad. Es decir, la tabla Presidente guardaría la clave primaria de País y la tabla País guardaría también la clave primaria de Presidente. Esta solución puede presentar redundancia, pero puede ser interesante en algunas ocasiones, dependiendo de las consultas que se vayan a realizar sobre estas tablas a nivel de aplicación. En este caso los atributos id_país y id_presidente serían claves_ajenas o foreign key (FK).

- PRESIDENTE(<u>id</u>, nombre, apellido1, apellido2, *id_país*)
  - id_país: FOREIGN KEY de PAÍS(id)
- PAÍS(<u>id</u>, nombre, *id_presidente*)
  - id_presidente: FOREIGN KEY de PRESIDENTE(id)

5.2 Relaciones

- **Cardinalidad 1:1 → *Participación (0,1)..(0,1)***:

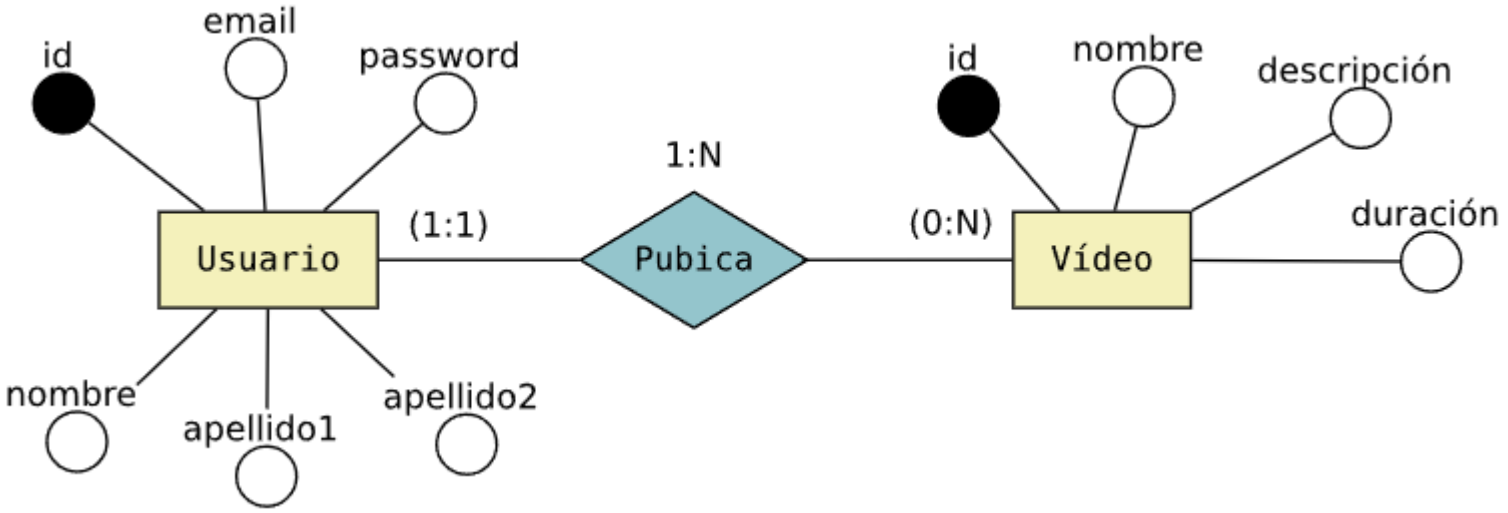


3. Cuando la participación de las dos entidades es de (0,1), se puede crear una nueva tabla donde se almacenan las claves primarias de las dos entidades que participan en la relación. La clave primaria de la nueva tabla será una de las dos claves ajenas que se reciben.

- ALQUILER(id, fecha_inicio, fecha_fin, importe, fianza)
- ALQUILER_RENUEVA_ALQUILER(*id_alquiler*, *id_alquiler_anterior*)
  - id_alquiler: FOREIGN KEY de ALQUILER(id)
  - id_alquiler_anterior: FOREIGN KEY de ALQUILER(id)

5.2 Relaciones

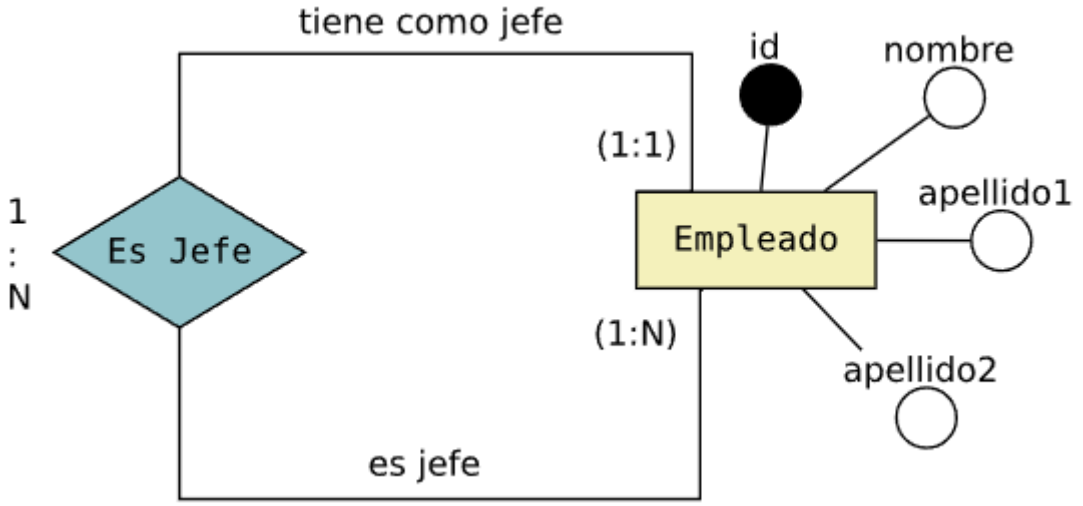
- **Cardinalidad 1:N**:



- En este caso la clave primaria de la entidad que participa en la relación con cardinalidad 1 se guarda en la tabla de la entidad que participa con cardinalidad N.
  - USUSARIO(<u>id</u>, email, password, nombre, apellido1, apellido2)
  - VÍDEO(<u>id</u>, nombre, descripción, duración, *id_usuario*)
    - id_usuario: FOREIGN KEY de USUARIO(id).

5.2 Relaciones

- **Cardinalidad 1:N → Relaciones reflexivas con cardinalidad 1:N**

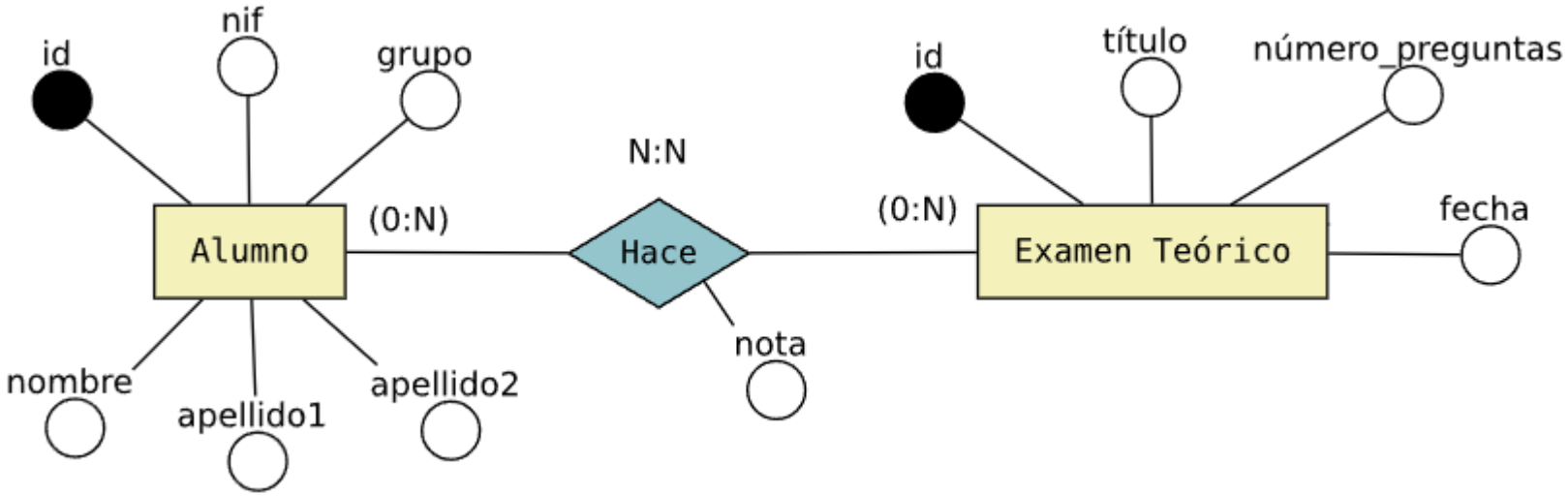


- En este caso la clave primaria se almacena en la misma tabla como atributo. La tabla Empleado vuelve a guardar su clave primaria como atributo haciendo referencia al id del jefe, le llamaremos id_jefe.

  - EMPLEADO(id, nombre, apellido1, apellido2, *id_jefe*)
    - id_jefe: FOREIGN KEY de EMPLEADO(id)

5.2 Relaciones

- **Cardinalidad N:M:**

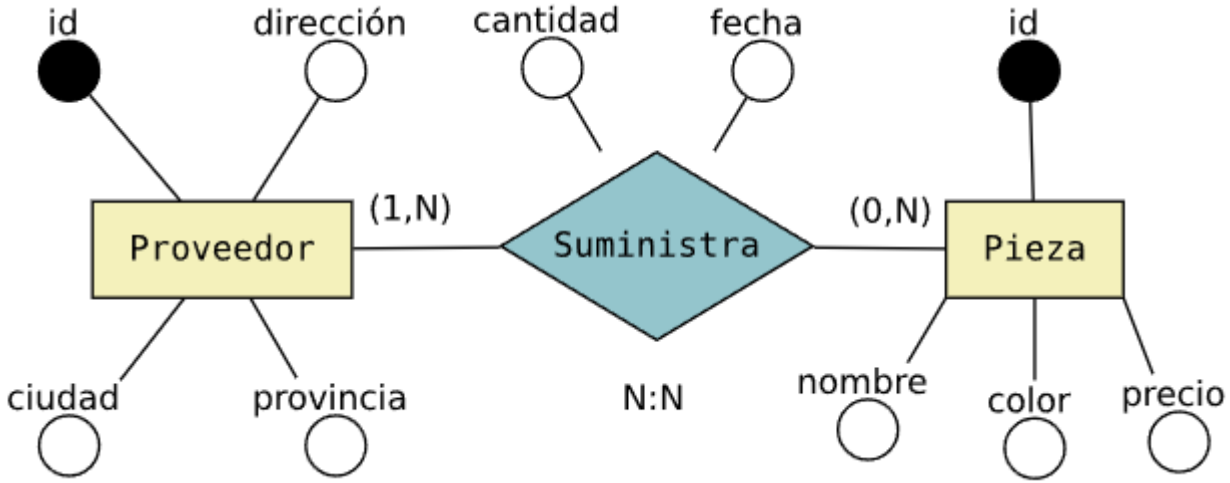


- En este caso se crea una nueva tabla donde se almacenan las claves primarias de las dos entidades que participan en la relación. Las claves primarias de las entidades también serán claves primarias de la nueva tabla. Si la relación contiene algún atributo, se deberán añadir a la nueva tabla.

  - ALUMNO(<u>id</u>, nombre, apellido1, apellido2, nif, grupo)
  - EXAMEN_TEÓRICO(<u>id</u>, título, número_preguntas, fecha)
  - ALUMNO_HACE_EXAMEN_TEÓRICO(<u>*id_alumno, id_examen*</u>, nota)
    - id_alumno: FOREIGN KEY de ALUMNO(id)
    - id_examen: FOREIGN KEY de EXAMEN(id)

5.2 Relaciones

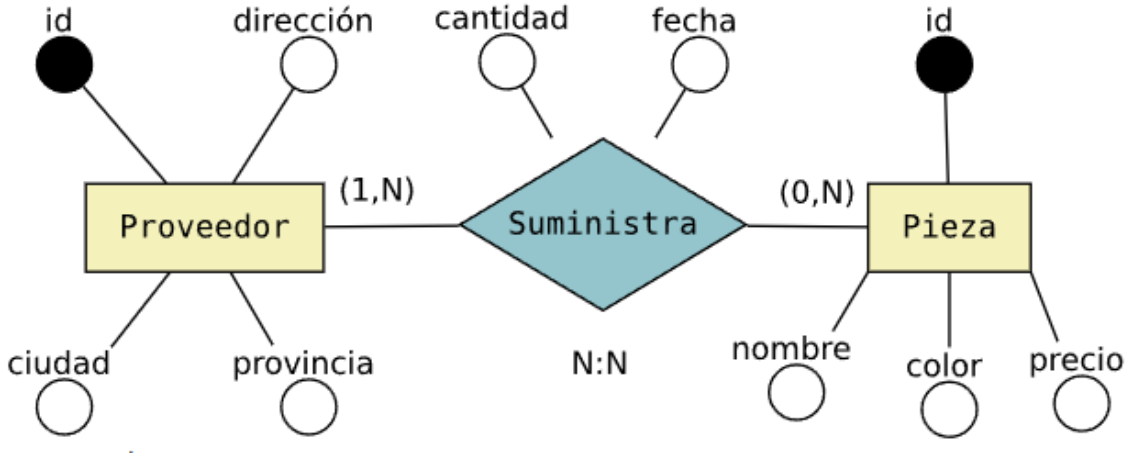
- **Cardinalidad N:M:** habrá casos donde los atributos de la relación también formarán parte de la clave primaria de la nueva tabla. Estos casos aparecerán cuando en la relación existan atributos de tipo fecha y sea necesario almacenar un histórico de las relaciones entre las dos entidades en función de las fechas. Estos casos también pueden resolverse añadiendo un nuevo identificador de tipo entero con autoincremento en lugar de utilizar una clave primaria compuesta por varias columnas.



- PROVEEDOR(id, dirección, ciudad, provincia)
- PIEZA(id, nombre, color, precio)
- PROVEEDOR_SUMINISTRA_PIEZA(id_proveedor, id_pieza, fecha, cantidad)
  - id_proveedor: FOREIGN KEY de PROVEEDOR(id)
  - id_pieza: FOREIGN KEY de PIEZA(id)

5.2 Relaciones

- **Cardinalidad N:M:**

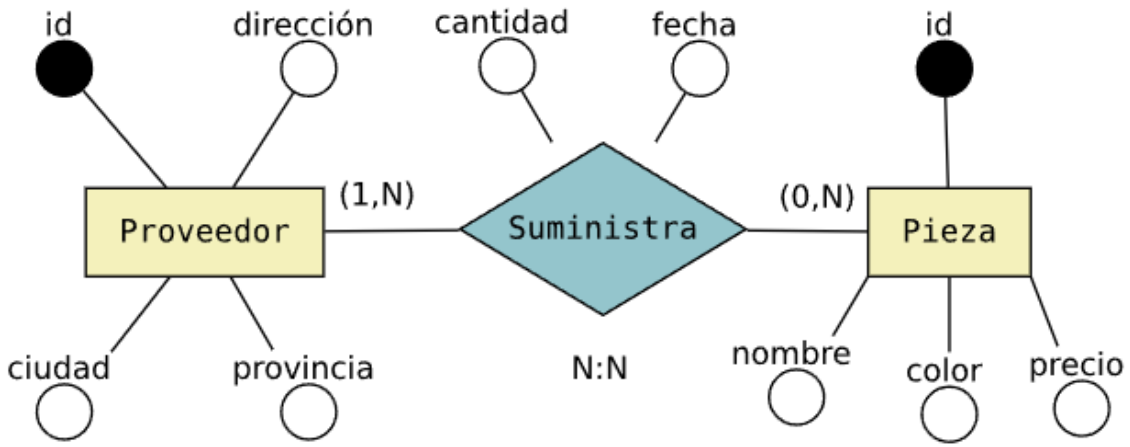


- Con esta solución podemos tener un problema en el caso de que un proveedor nos suministre piezas con el mismo id en fechas diferentes. En este caso no podríamos almacenar esta información en la tabla porque se produciría un error de claves primarias duplicadas.

#id_proveedor	#id_pieza	fecha	cantidad
1	1	01/01/2018	100
1	1	20/01/2018	100

5.2 Relaciones

- **Cardinalidad N:M:**

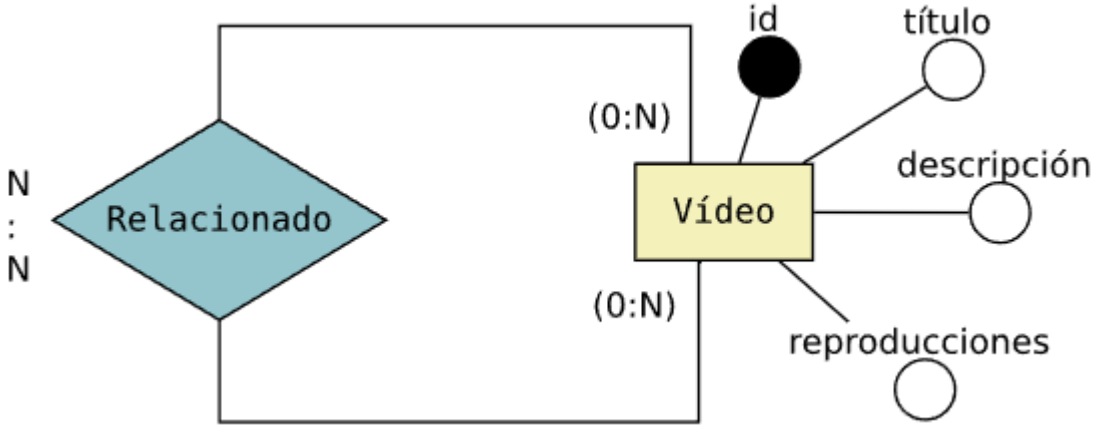


- Para solucionarlo podemos incluir el atributo fecha como parte de la clave primaria de la tabla, de modo que la clave primaria estaría compuesta por id_proveedor, id_pieza y fecha. La solución sería la siguiente:
  - PROVEEDOR_SUMINISTRA_PIEZA(*id_proveedor, id_pieza*, fecha, cantidad)
    - id_proveedor: FOREIGN KEY de PROVEEDOR(id)
    - id_pieza: FOREIGN KEY de PIEZA(id)
- 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

5.2 Relaciones

- **Cardinalidad N:M → *Relaciones reflexivas con cardinalidad N:M***

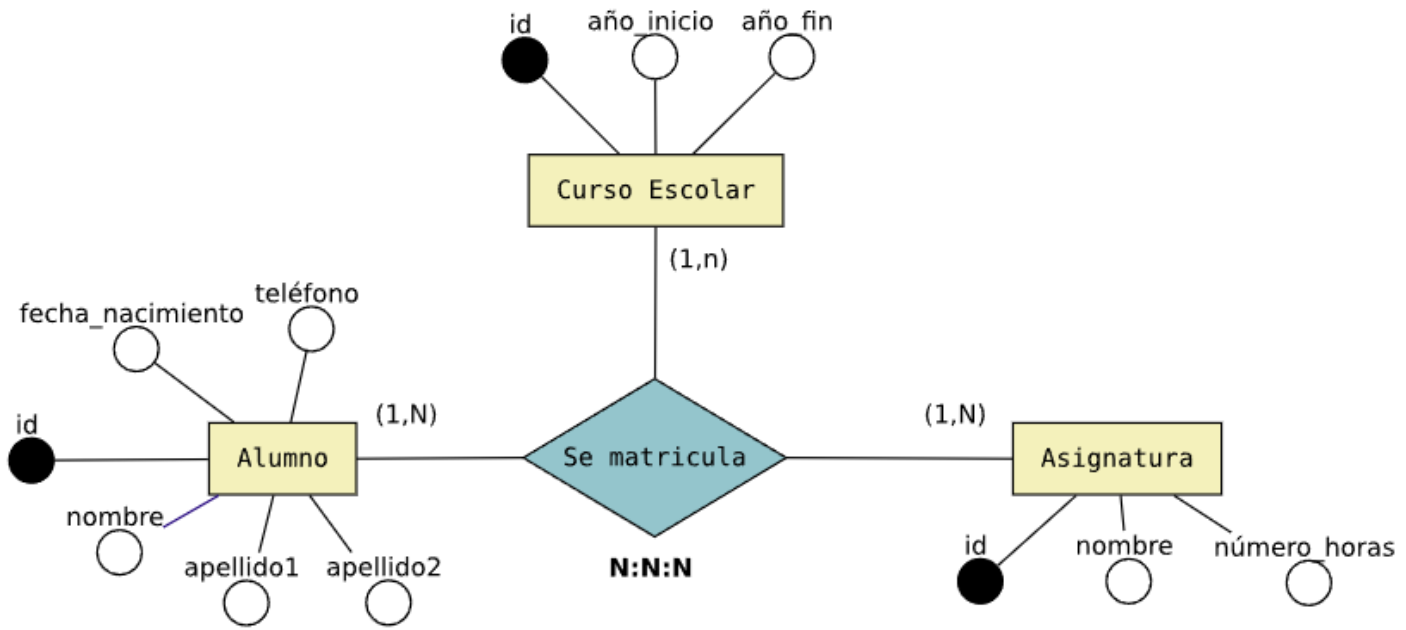


- En este caso tendremos dos tablas en el modelo relacional:

  - VÍDEO(<u>id</u>, título, descripción, reproducciones)
  - VÍDEOS_RELACIONADOS(<u>*id_video, id_video_relacionado*</u>)
    - id_video: FOREIGN KEY de VÍDEO(id)
    - id_video_relacionado: FOREIGN KEY de VÍDEO(id)

5.2 Relaciones ternarias

- **Cardinalidad N:N:M:**



- En este caso creamos una tabla. La clave primaria de la nueva tabla estará formada por las tres claves de las entidades que participan en la relación:
  - MATRICULA(*id_alumno, id_curso, id_asignatura*)
    - id_alumno: FOREIGN KEY de ALUMNO(id)
    - id_curso: FOREIGN KEY de CURSO_ESCOLAR(id)
    - id_asignatura: FOREIGN KEY de ASIGNATURA(id)

5.2 Relaciones ternarias

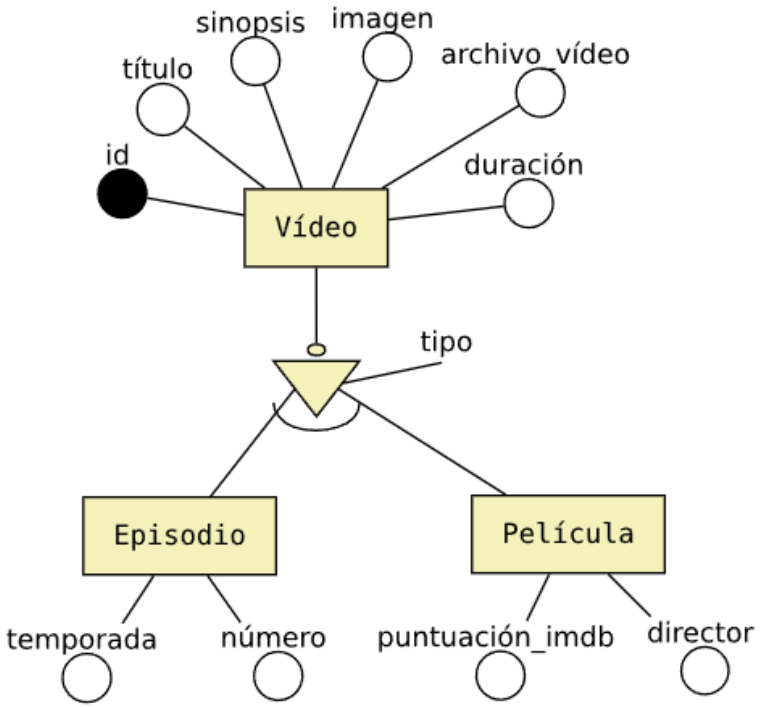
- **Cardinalidad 1:N:M**
  - En este caso creamos una tabla. La clave primaria de la nueva tabla estará formada por las dos claves de las entidades que participan como N en la relación.
- **Cardinalidad 1:1:N**
  - En este caso no es necesario crear una tabla. La entidad que participa como N recibe las claves de las dos entidades que participan como 1.

5.3 Generalización y especialización

- Existen varias soluciones para realizar el paso a tablas de una especialización. La solución que se elija en cada caso dependerá del tipo de especialización que estemos resolviendo: total, parcial, inclusiva o exclusiva.
- Las 3 soluciones posibles que podemos aplicar son las siguientes:
  1. Crear una única tabla para la superclase. En este caso todos los atributos de las subclases se guardarían en la superclase.
  2. Crear una tabla sólo para las subclases. En este caso los atributos de la superclase habría que guardarlos en cada una de las subclases.
  3. Crear una tabla para cada una de las entidades, tanto para la superclase como las subclases. En este caso las subclases tendrían que guardar la clave de la primaria de la superclase.

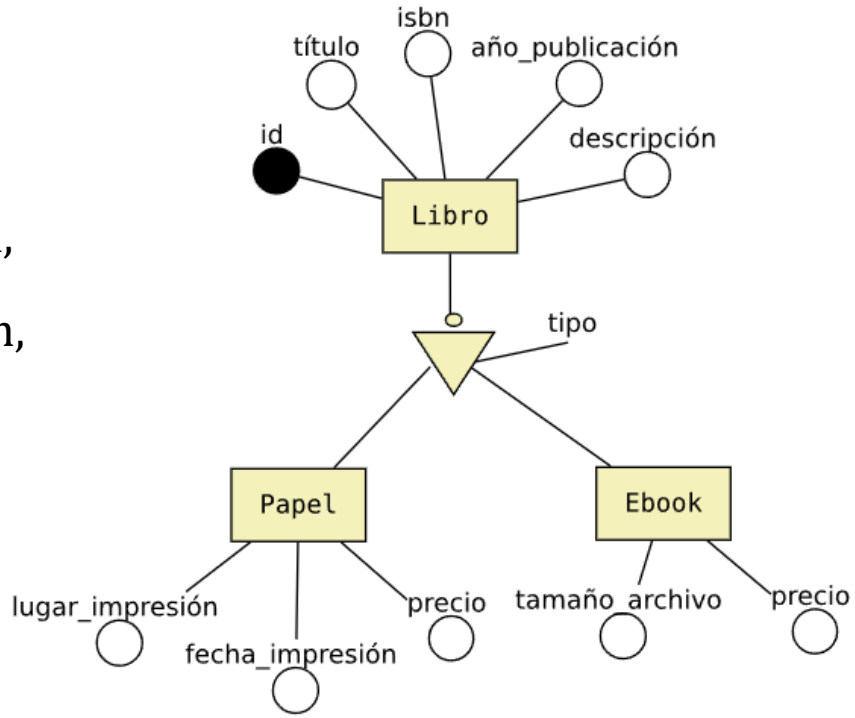
5.3 Generalización y especialización

- **Ejemplo de especialización exclusiva/total:**
- *Solución 2: Crear una tabla sólo para las subclases*.
  - EPISODIO(id, título, sinopsis, imagen, archivo_vídeo, duración temporada, número)
  - PELÍCULA(id, título, sinopsis, imagen, archivo_vídeo, duración puntuación_imdb, director)
- *Solución 3: Crear una tabla para cada una de las entidades*.
  - VÍDEO(id, título, sinopsis, imagen, archivo_vídeo, duración, **tipo**)
  - EPISODIO(*id*, temporada, número)
    - id: FK de VÍDEO(id)
  - PELÍCULA(*id*, puntuación_imdb, director)
    - id: FK de VÍDEO(id)



5.3 Generalización y especialización

- **Ejemplo de especialización solapada/total:**
- *Solución 1. Crear una única tabla para la superclase*.
  - LIBRO(id, título, isbn, año_publicación, descripción, **tipo**, lugar_impresión, fecha_impresión, precio_papel, tamaño_archivo, precio_ebook)
- *Solución 2: Crear una tabla sólo para las subclases*.
  - LIBRO_PAPEL(id, título, isbn, año_publicación, descripción, lugar_impresión, fecha_impresión, precio)
  - LIBRO_EBOOK(id, título, isbn, año_publicación, descripción, tamaño_archivo, precio)
- *Solución 3: Crear una tabla para cada una de las entidades*.
  - LIBRO(id, título, isbn, año_publicación, descripción, tipo)
  - LIBRO_PAPEL(*id*, fecha_impresión, precio)
    - id: FK de LIBRO(id)
  - LIBRO_EBOOK(*id*, tamaño_archivo, precio)
    - id: FK de LIBRO(id)

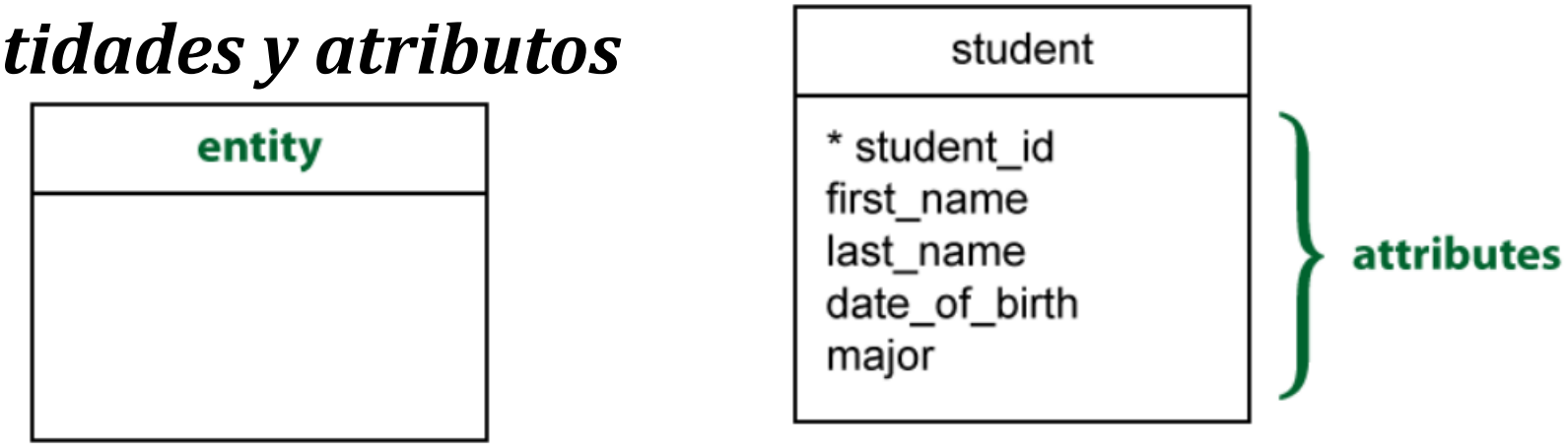


5.4 De modelo E/R a CROW'S FOOT

2. **<u>El grafo relacional crow's foot (patas de gallo)</u>**: es similar al grafo relacional clásico, pero en este caso se representa cada relación como una ***tabla***, y las claves foráneas serán ***enlaces*** entre tablas (cada clave foránea se enlazará con la clave primaria de la que depende). En los extremos de los enlaces se indica la ***cardinalidad*** de cada relación.

- Representación de cada componente/característica:

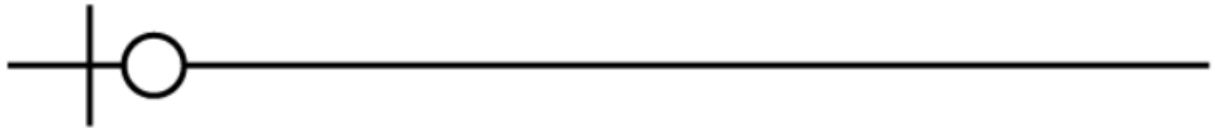
1. Entidades y atributos



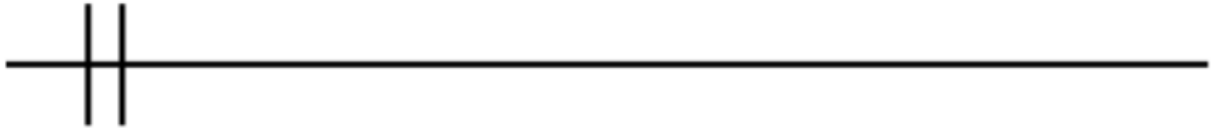
5.4 De modelo E/R a CROW'S FOOT

2. **<u>El grafo relacional crow's foot (patas de gallo)</u>**: representación de cada componente/característica:
   2. ***Cardinalidades a nivel de entidad:***

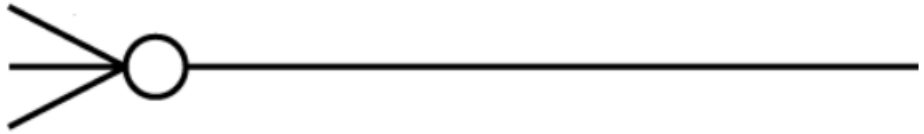
- (0,1)



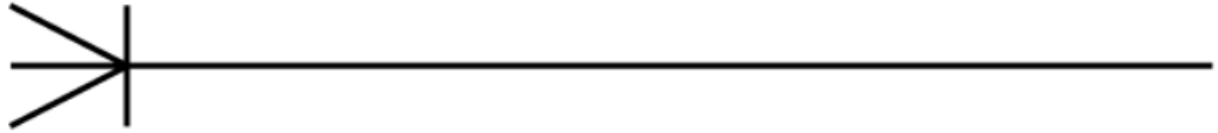
- (1,1)



- (0,N)



- (1,N)

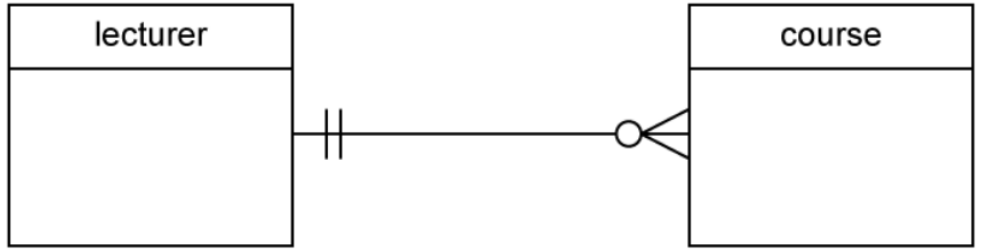


5.4 De modelo E/R a CROW'S FOOT

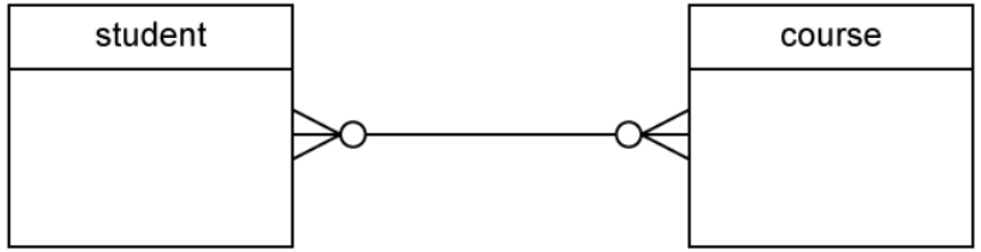
2. **<u>El grafo relacional crow's foot (patas de gallo)</u>**: representación de cada componente/característica:
   3. ***Cardinalidades de la relación:***

- 1:1

- 1:N

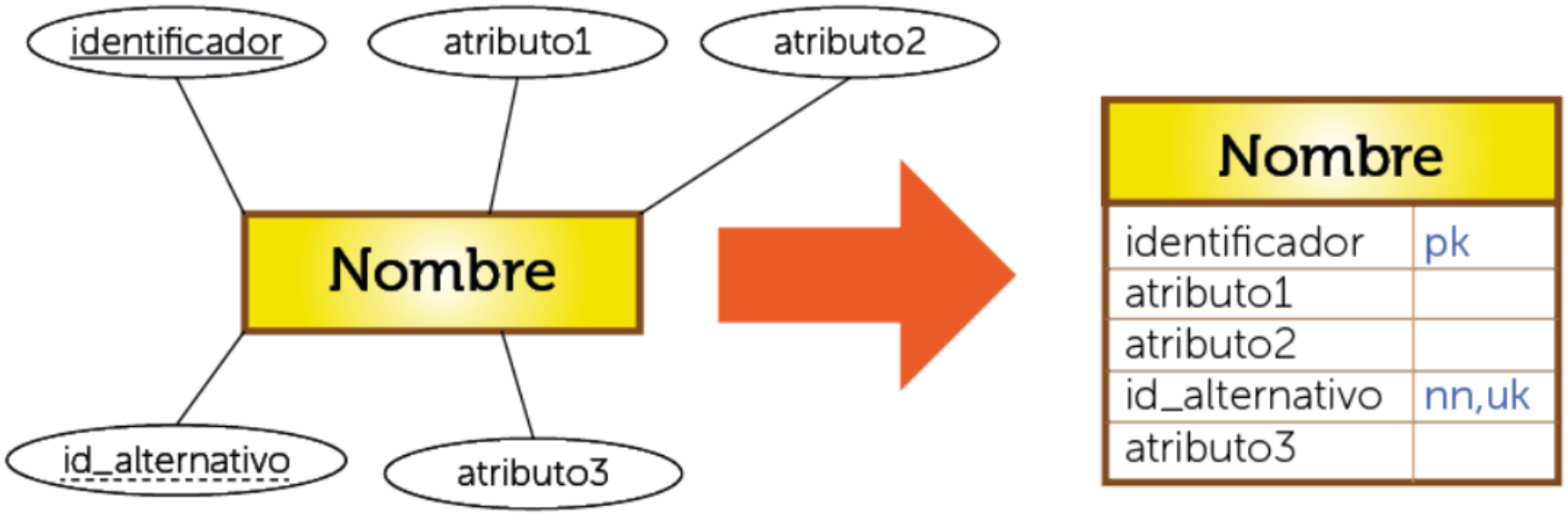


- N:M



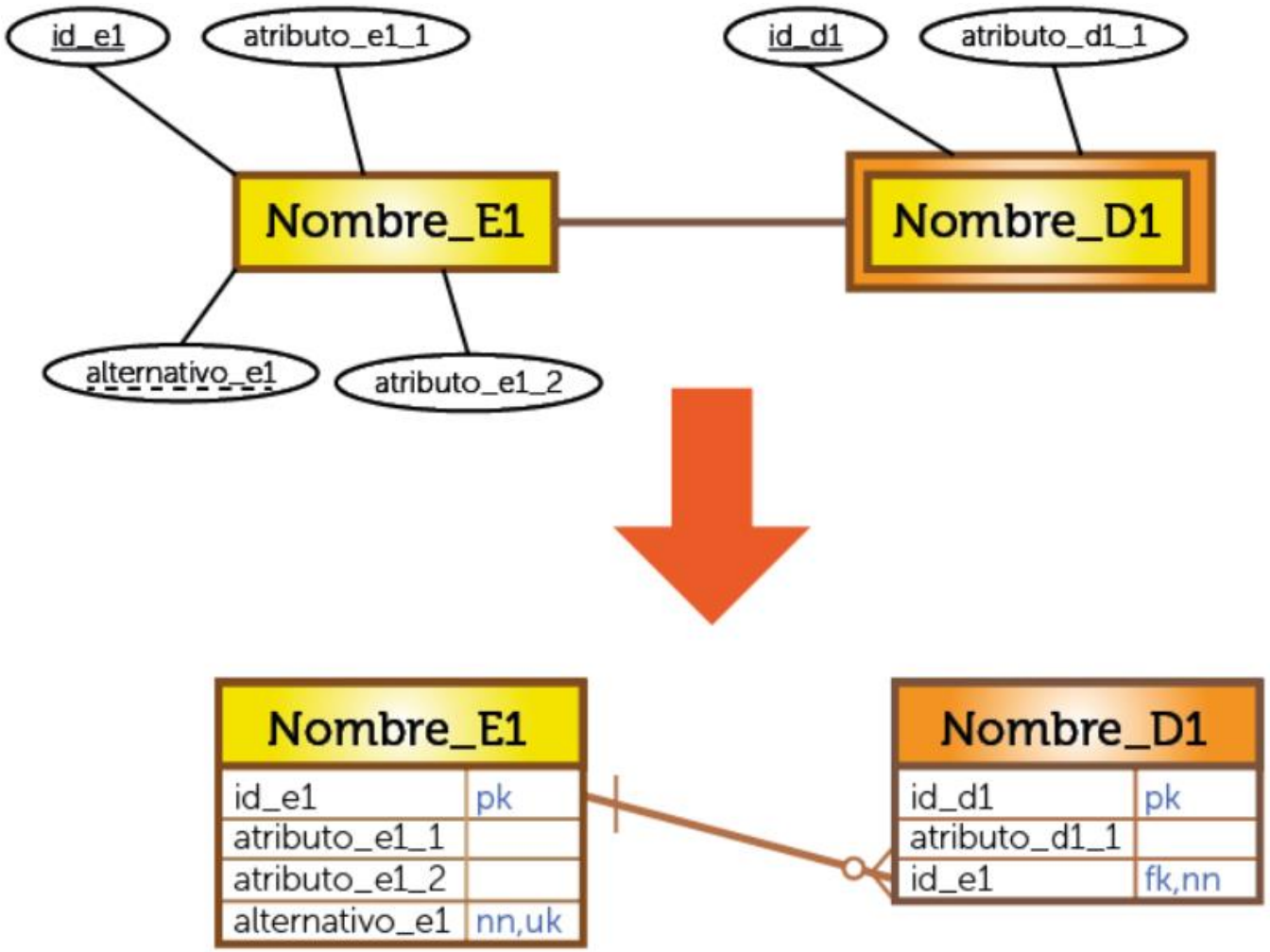
5.4 De modelo E/R a CROW'S FOOT

- ***Transformación entidades fuertes***



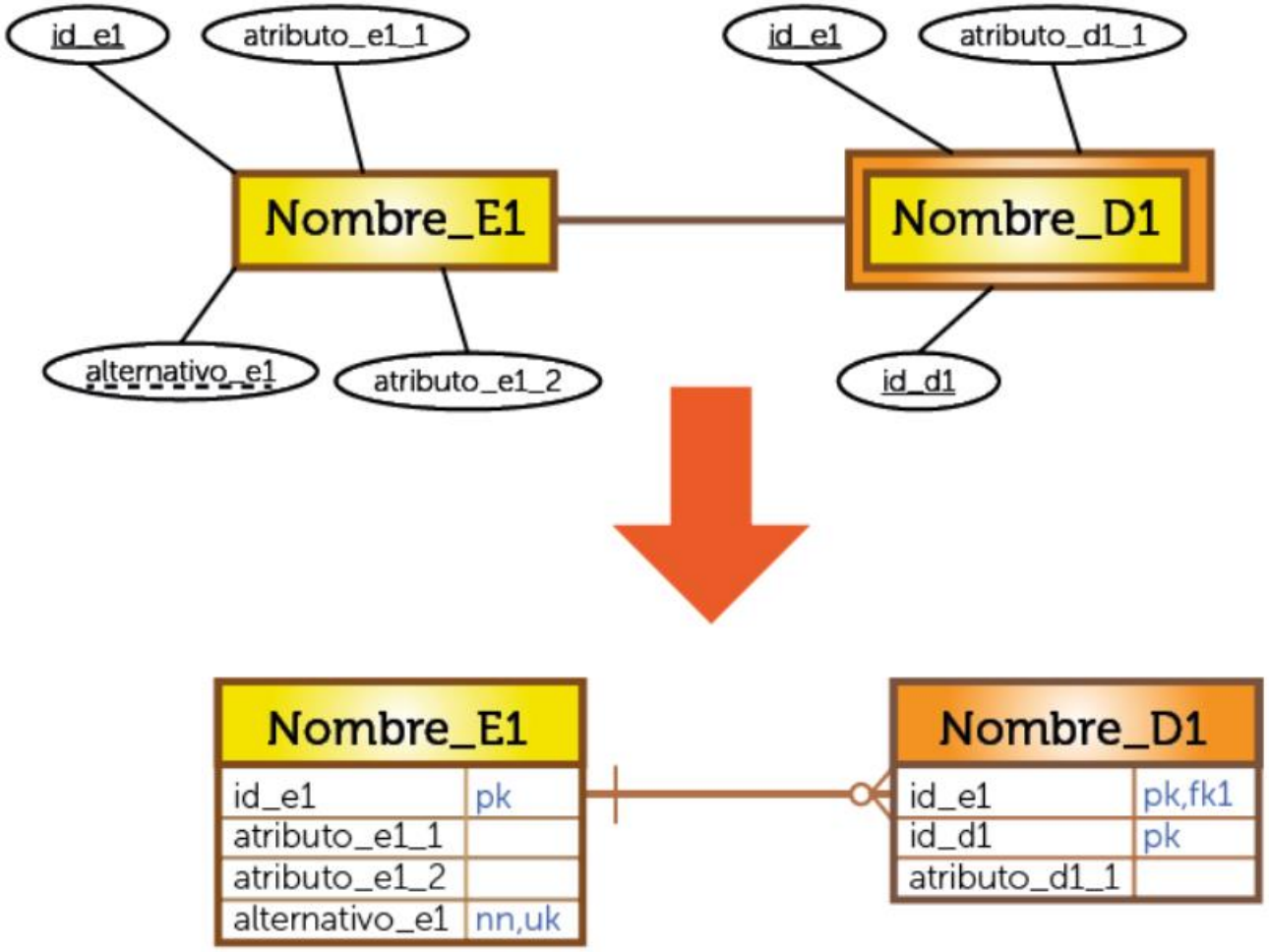
5.4 De modelo E/R a CROW'S FOOT

- ***Transformación entidades débiles: dependencia en existencia:*** bastará con añadir como atributo y clave foránea en la entidad débil, el identificador de la entidad fuerte.



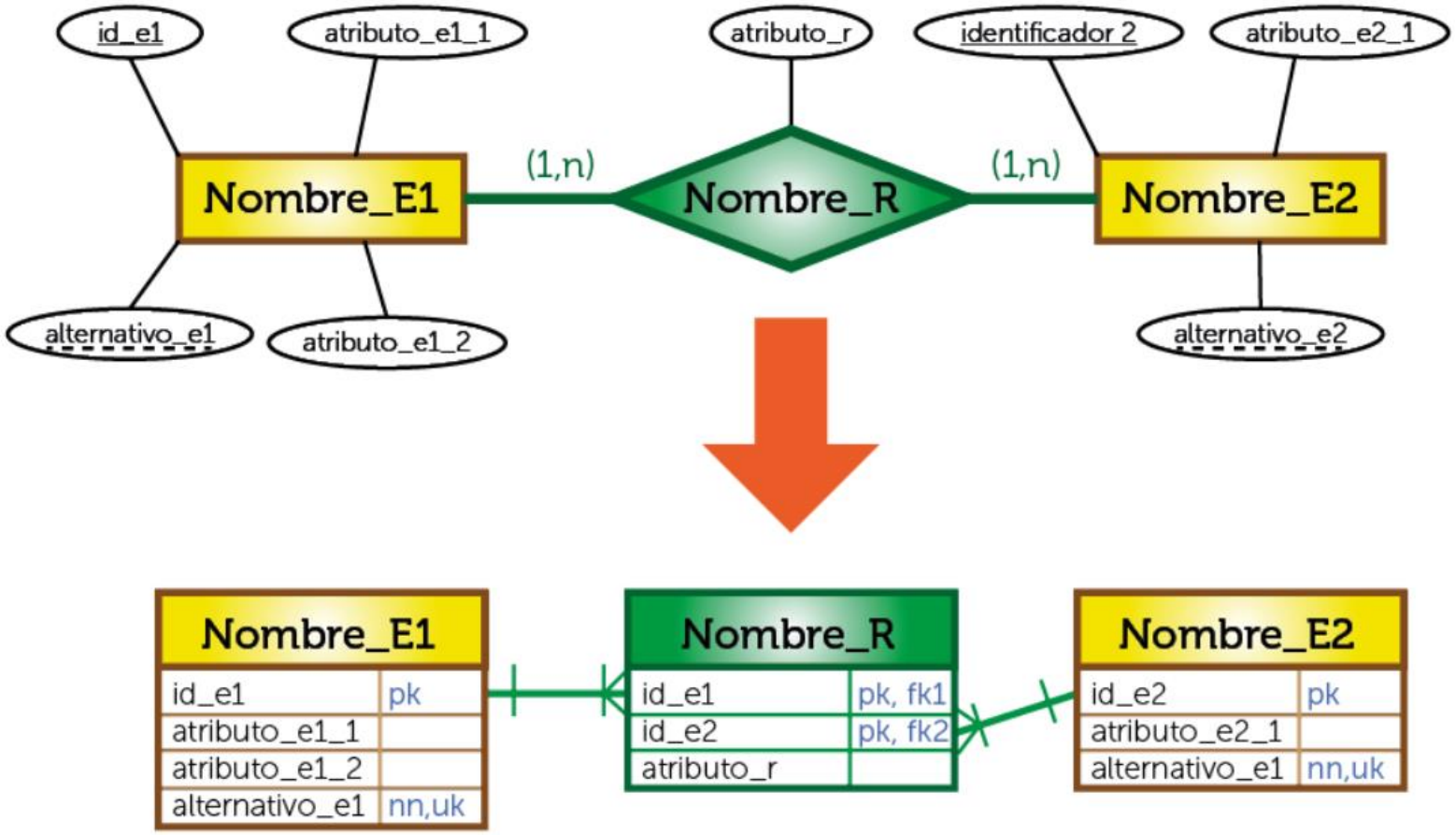
5.4 De modelo E/R a CROW'S FOOT

- ***Transformación entidades débiles: dependencia en <u>identificación:</u>*** en estos casos no hace falta añadir de nuevo como clave externa el identificador de la entidad fuerte.



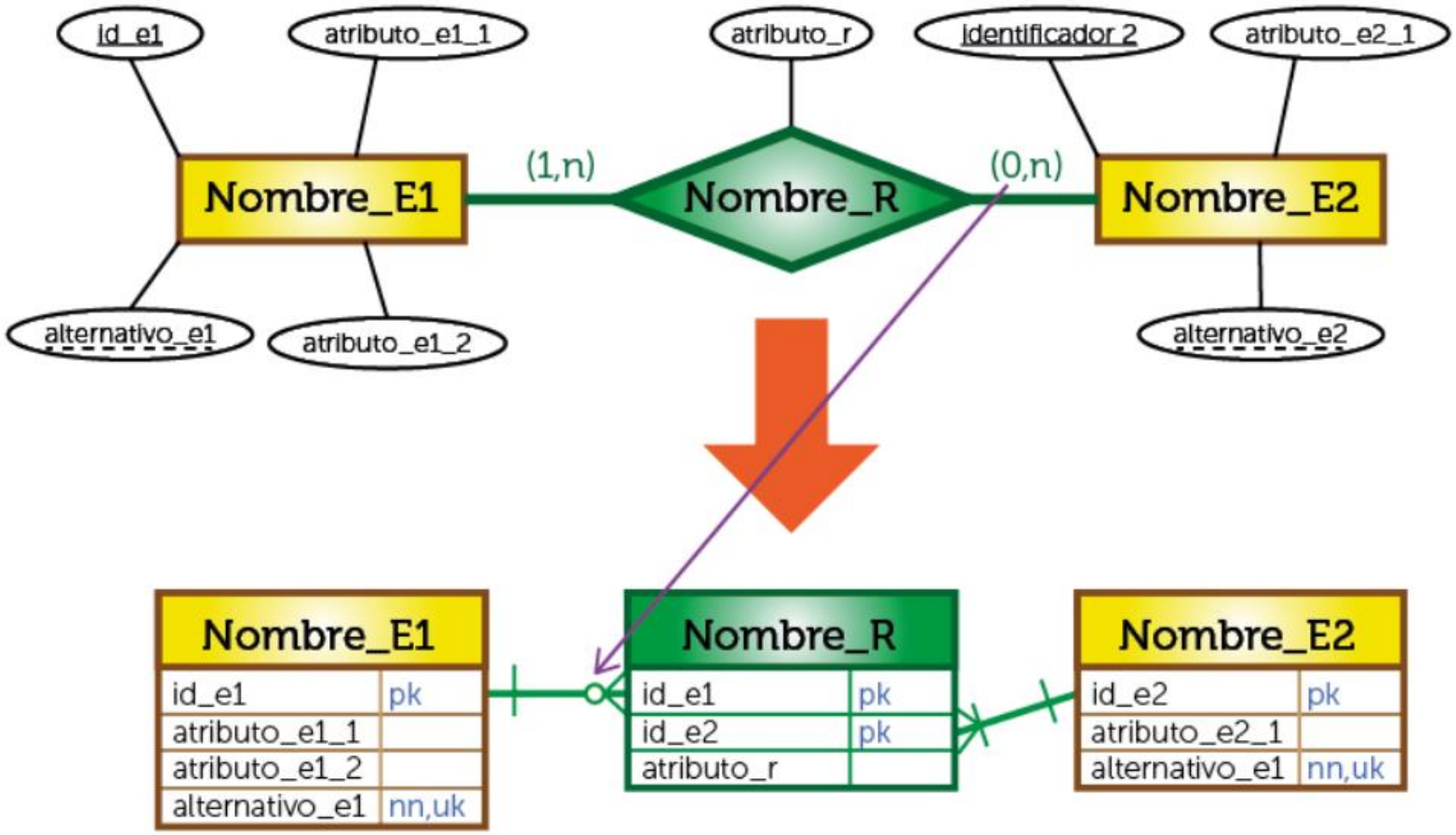
5.4 De modelo E/R a CROW'S FOOT

- ***Relación N:M*** → (relación (1,n) a (1,n), se convierte en dos relaciones (1,1) a (1,n))



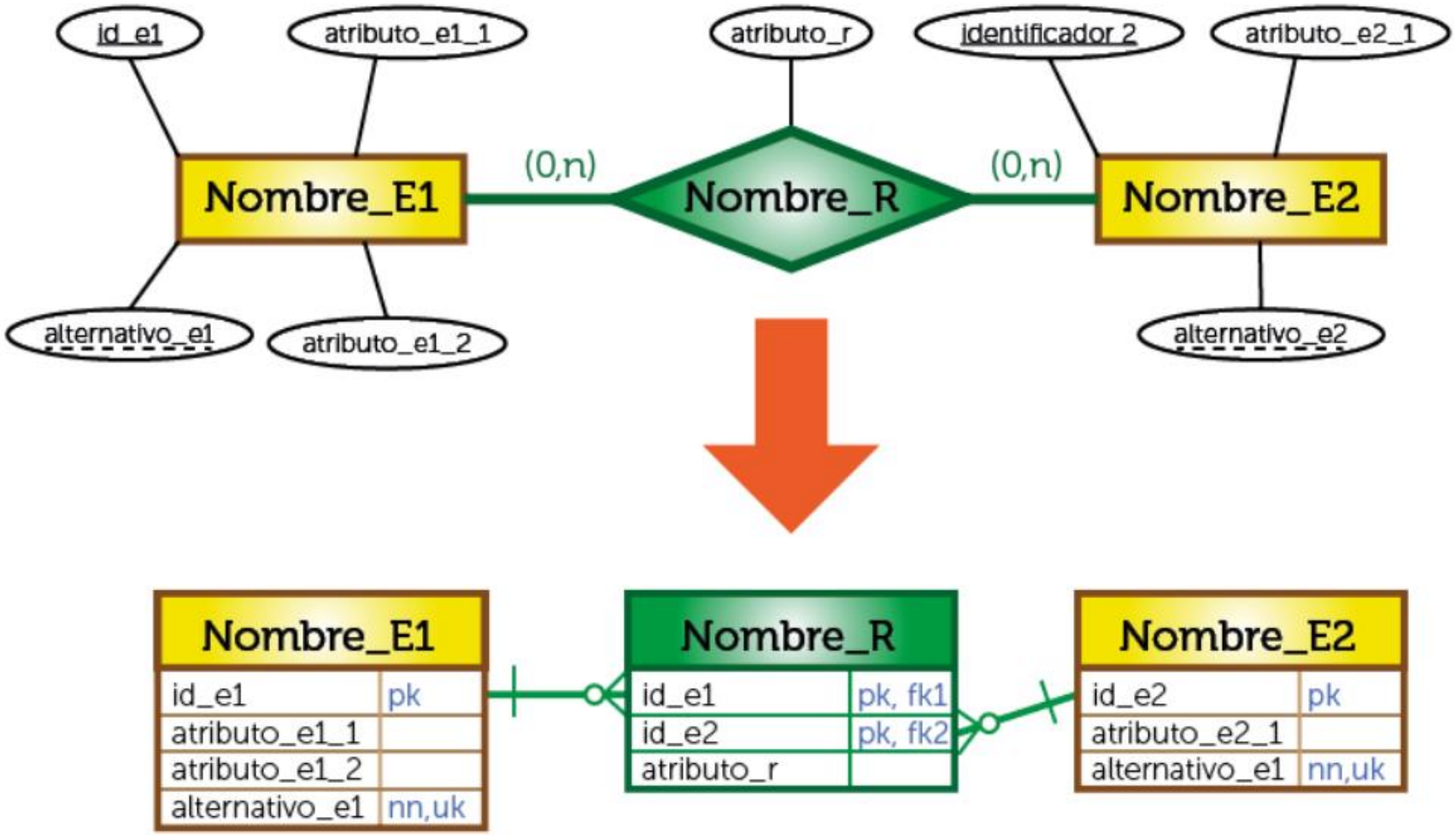
5.4 De modelo E/R a CROW'S FOOT

- ***Relación N:M*** → (relación (1,n) a (0,n), se transforma en una relación (1,1) a (1,n) y en otra (1,1) a (0,n))



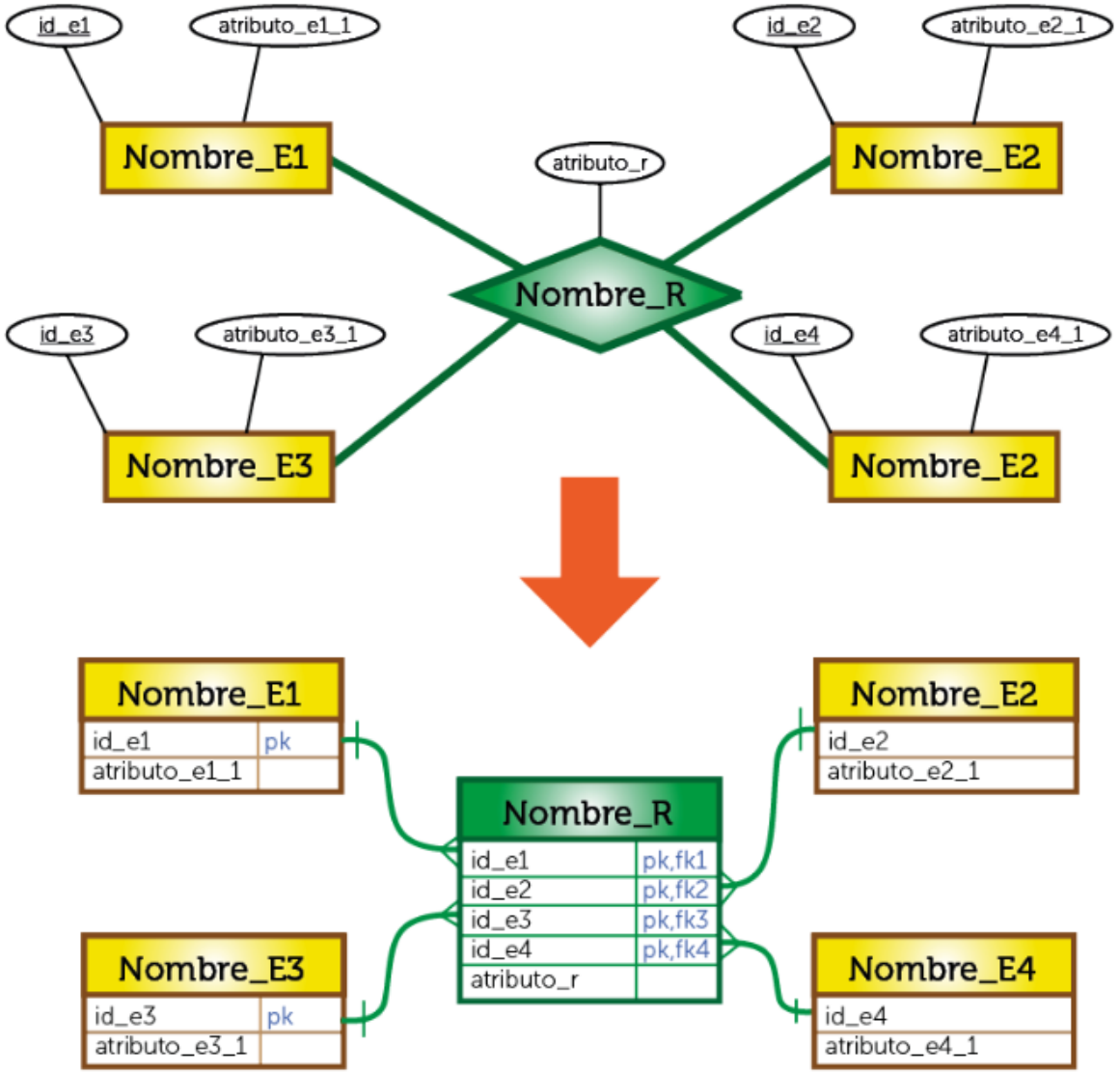
5.4 De modelo E/R a CROW'S FOOT

- ***Relación N:M*** → (relación (0,n) a (0,n), se convierte en dos relaciones (1,1) a (0,n))



5.4 De modelo E/R a CROW'S FOOT

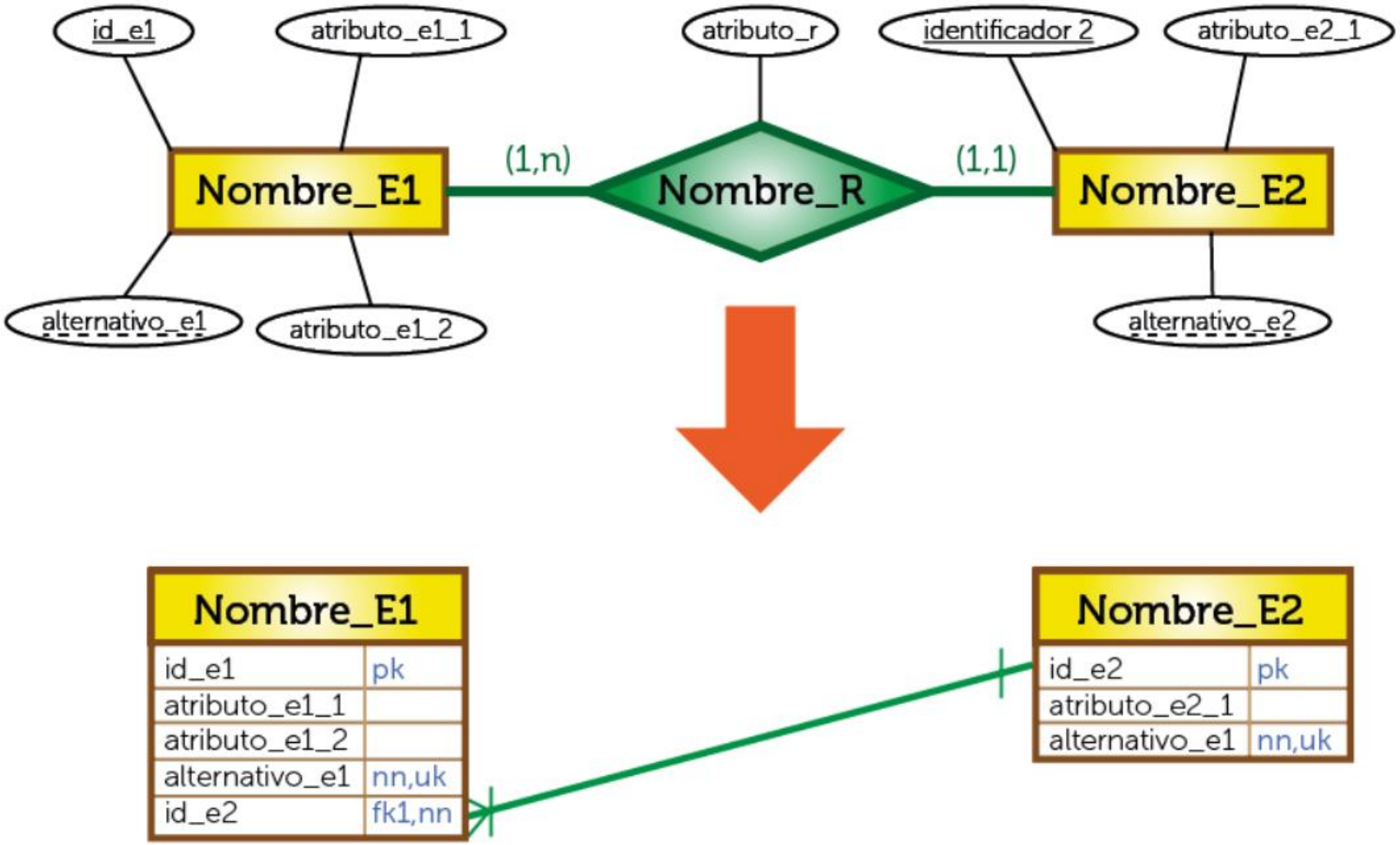
- ***Relaciones N-arias***



5.4 De modelo E/R a CROW'S FOOT

- ***Relación 1:N***

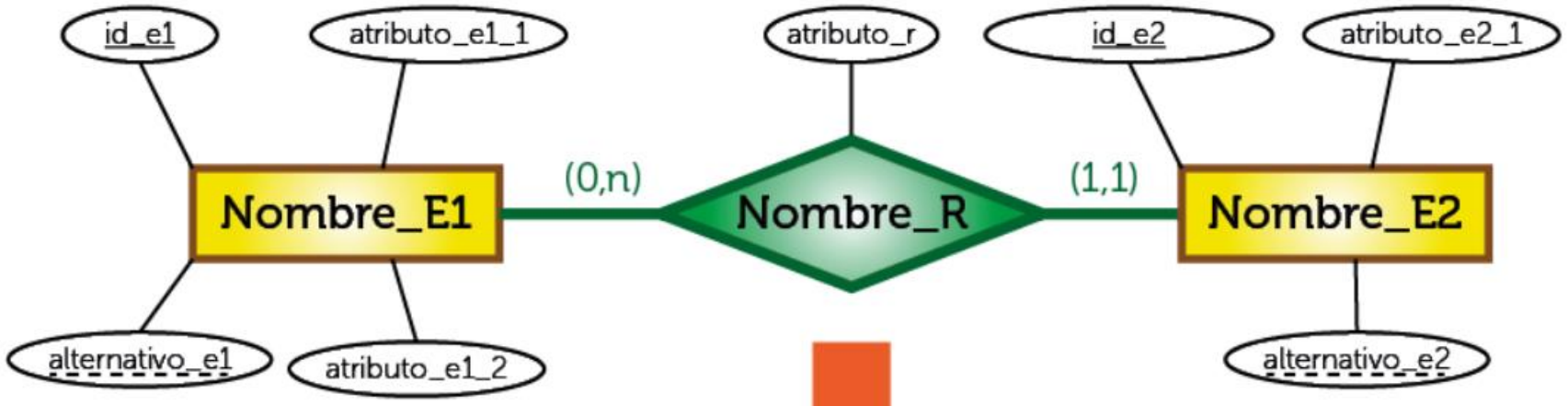
1. ***(1,N)..(1,1)***



5.4 De modelo E/R a CROW'S FOOT

- ***Relación 1:N***

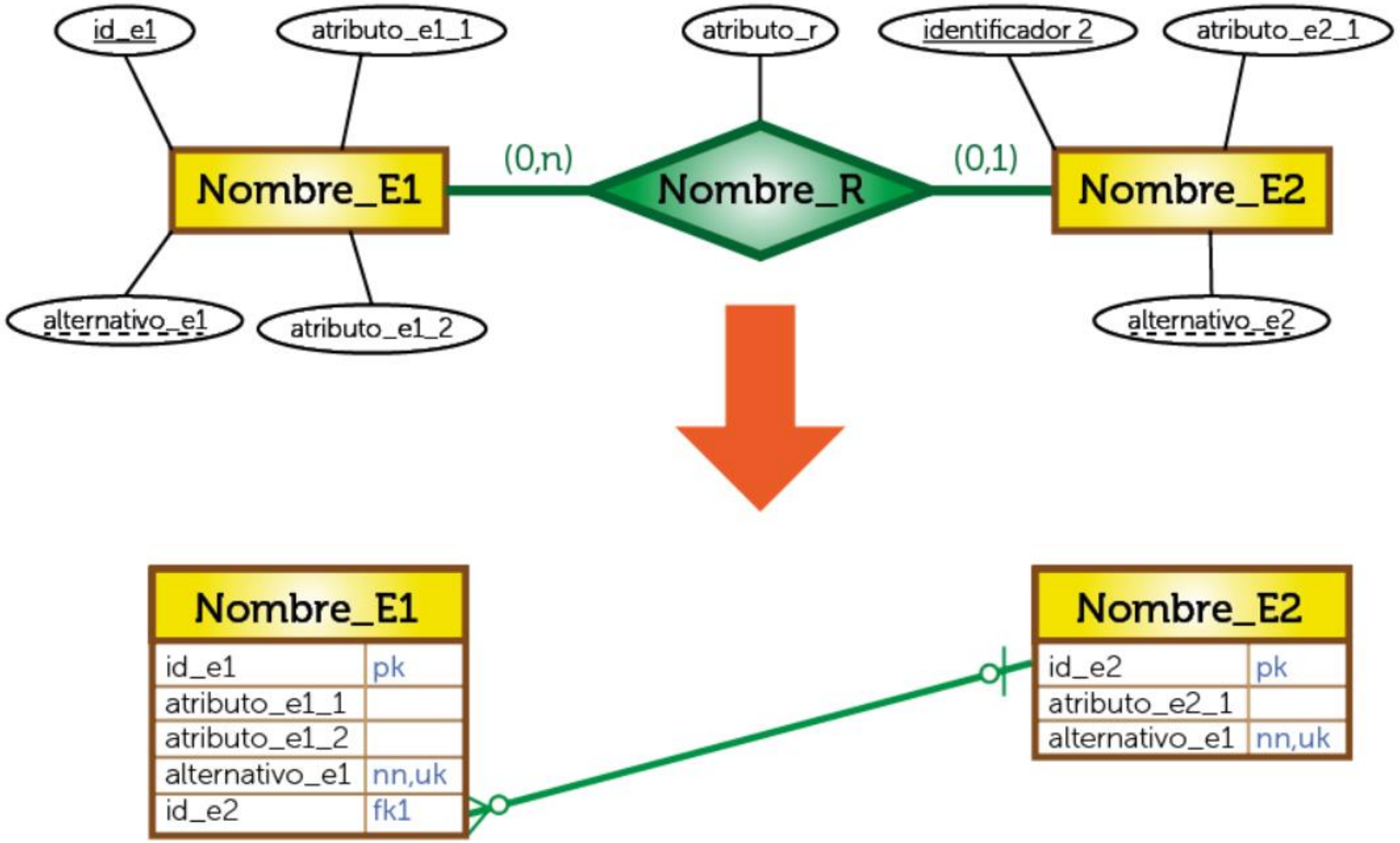
2. (0,N)..(1,1)



5.4 De modelo E/R a CROW'S FOOT

- ***Relación 1:N***

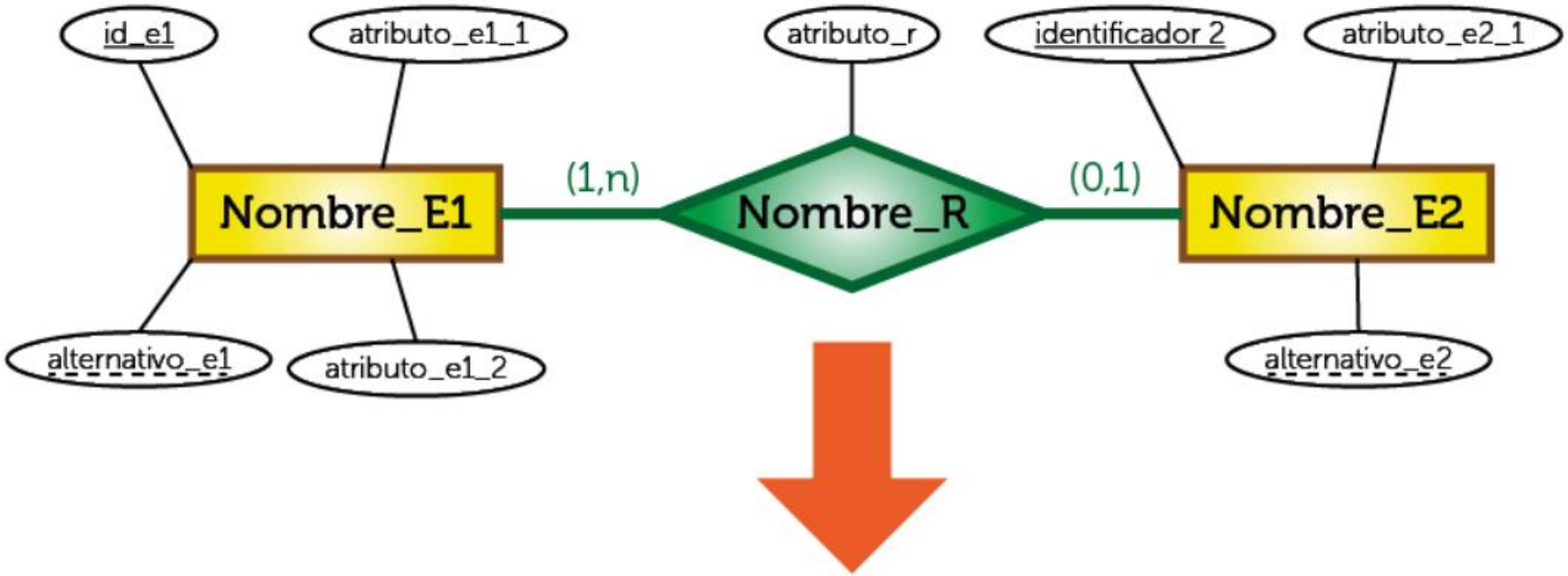
3. (0,N)..(0,1)



5.4 De modelo E/R a CROW'S FOOT

- ***Relación 1:N***

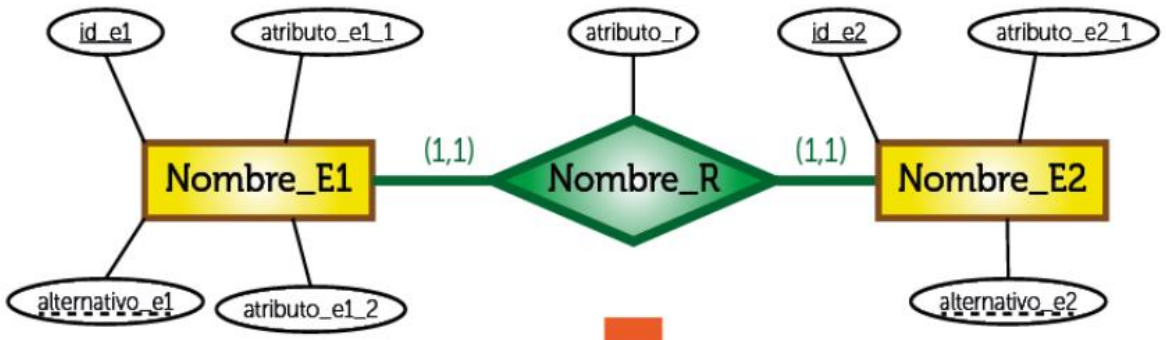
4. (1,N)..(0,1)



5.4 De modelo E/R a CROW'S FOOT

- ***Relación 1:1***

1. ***(1,1)..(1,1)***



Solución 1

Nombre_E1	
id_e1	pk
atributo_e1_1	
atributo_e1_2	
alternativo_e1	nn,uk1
atributo_r	
id_e2	uk2,nn
atributo_e2_1	
alternativo_e2	uk3,nn

Solución 2

Nombre_E1	
id_e1	pk
atributo_e1_1	
atributo_e1_2	
alternativo_e1	nn,uk1
id_e2	fk,uk2,nn
atributo_r	

Nombre_E2	
id_e2	pk
atributo_e2_1	
alternativo_e1	nn,uk

Solución 3

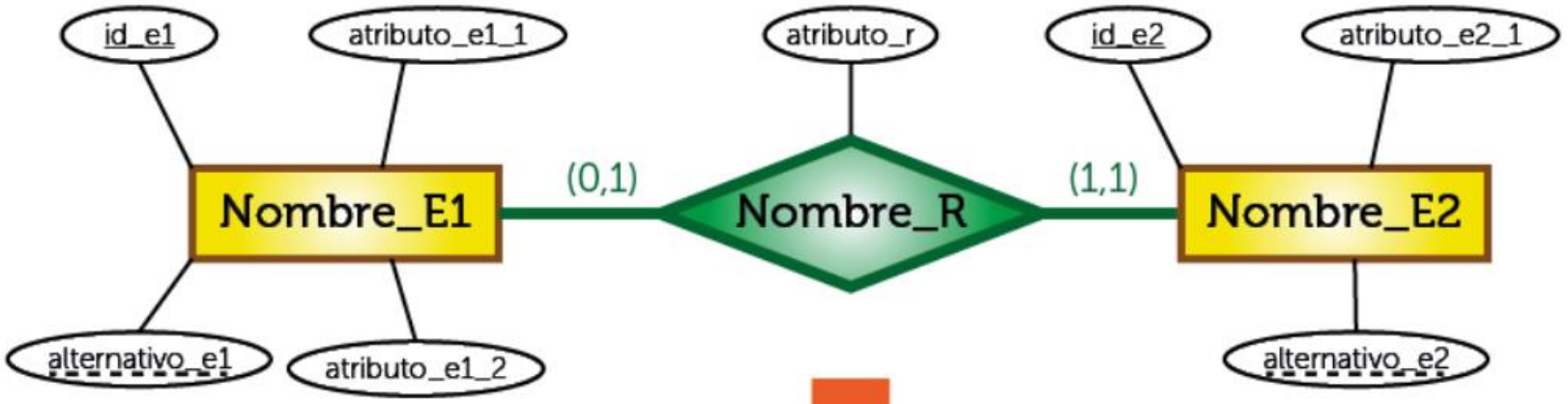
Nombre_E1	
id_e1	pk
atributo_e1_1	
atributo_e1_2	
alternativo_e1	nn,uk1

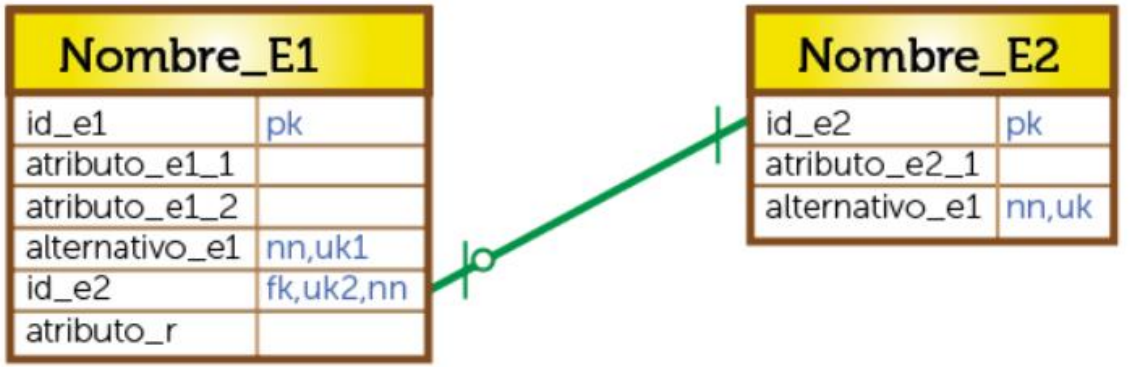
Nombre_E2	
id_e2	pk
atributo_e2_1	
alternativo_e1	nn,uk1
atributo_r	
id_e1	nn,fk,uk2

5.4 De modelo E/R a CROW'S FOOT

- ***Relación 1:1***

2. ***(0,1)..(1,1)***

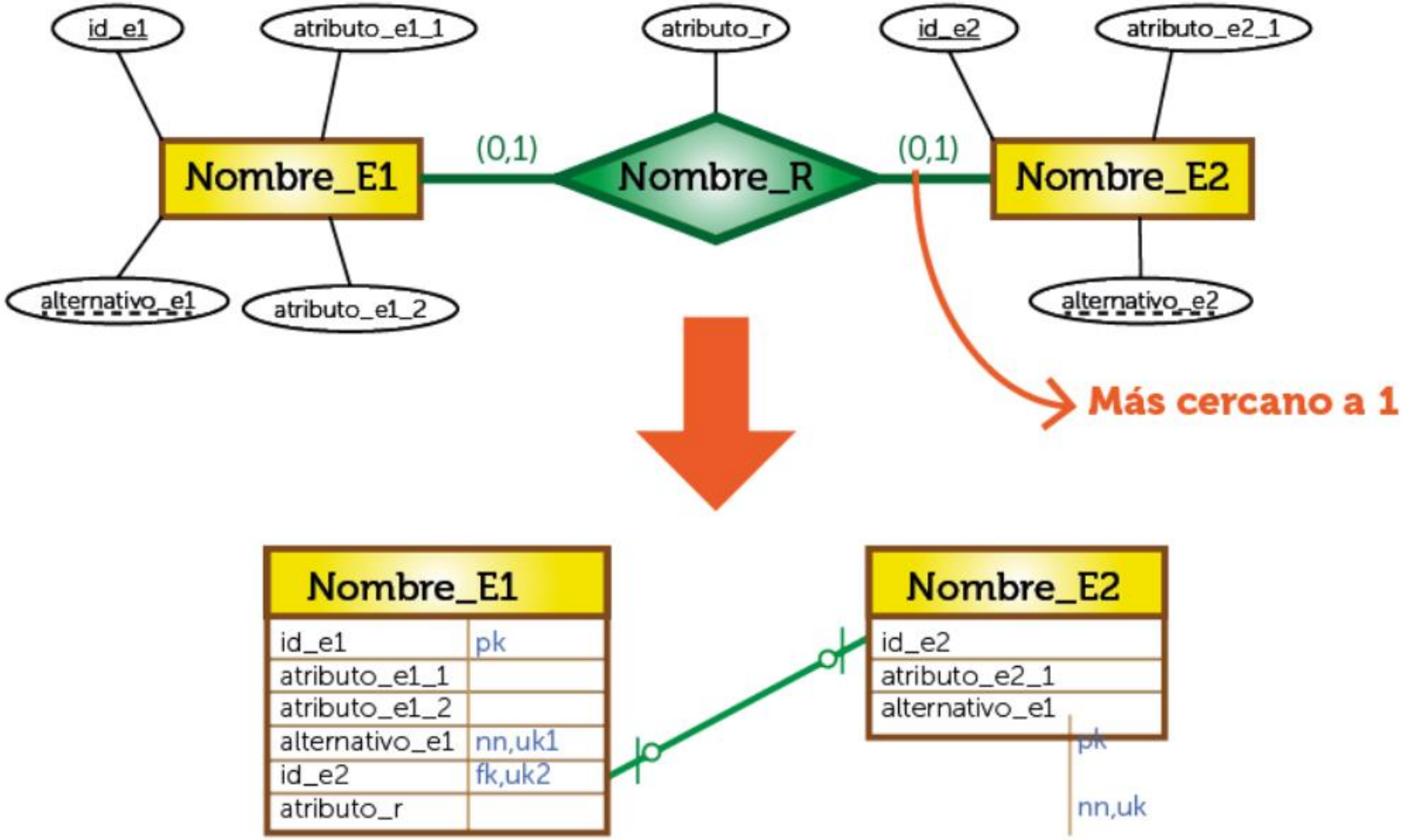




5.4 De modelo E/R a CROW'S FOOT

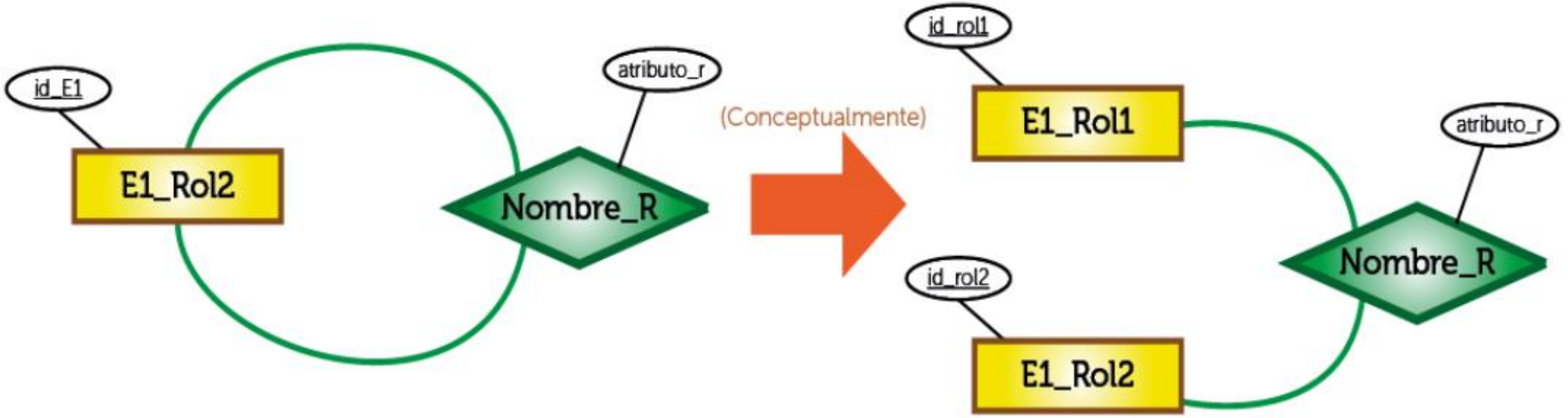
- ***Relación 1:1***

3. (0,1)..(0,1)



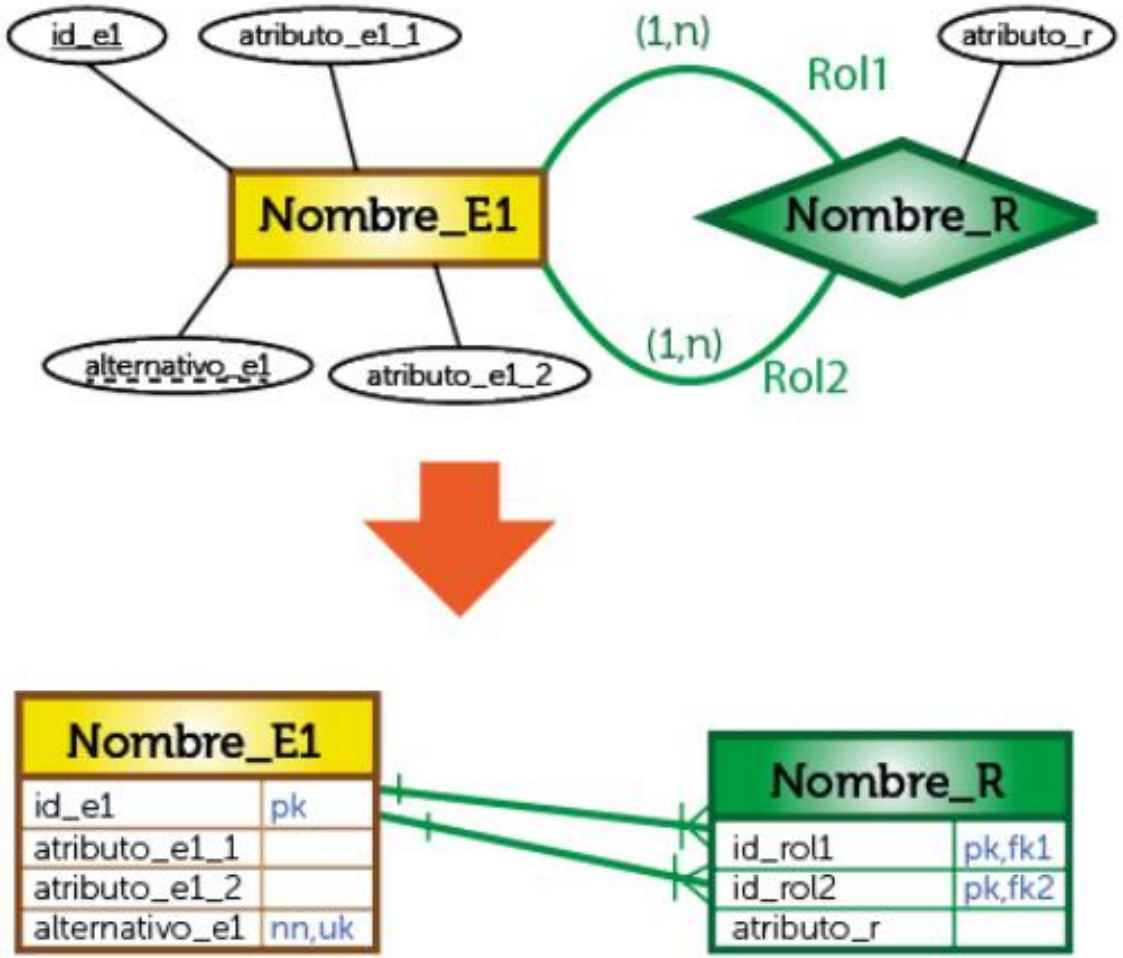
5.4 De modelo E/R a CROW'S FOOT

- ***Relaciones unarias/reflexivas:*** en realidad se convierten igual que las relaciones normales. Solo hay que tener en cuenta que la tabla relacionada es la misma.
- Conviene imaginarse como si la entidad relacionada fueran dos:



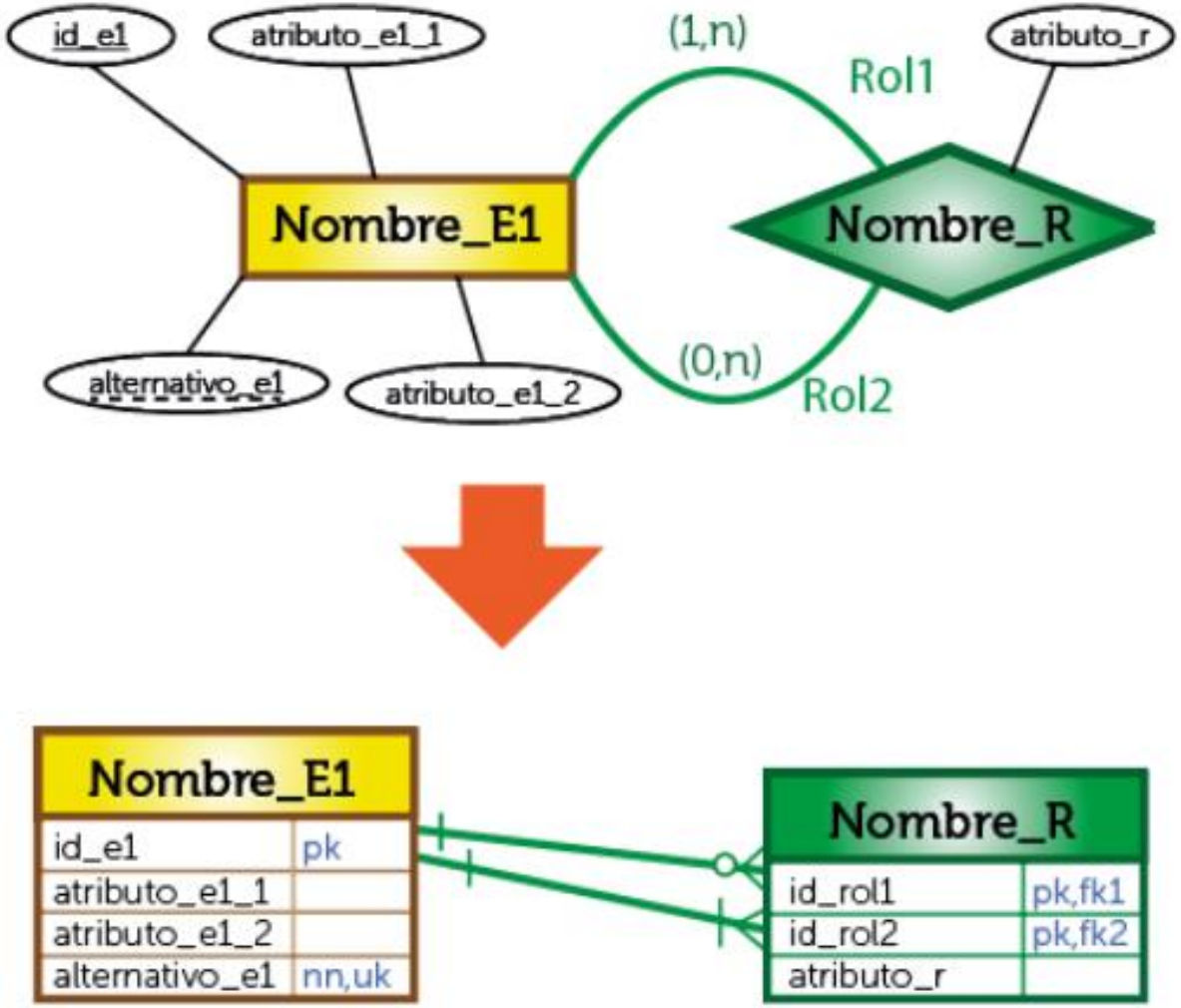
5.4 De modelo E/R a CROW'S FOOT

- ***Relaciones unarias/reflexivas:*** (1,N)..(1,N)



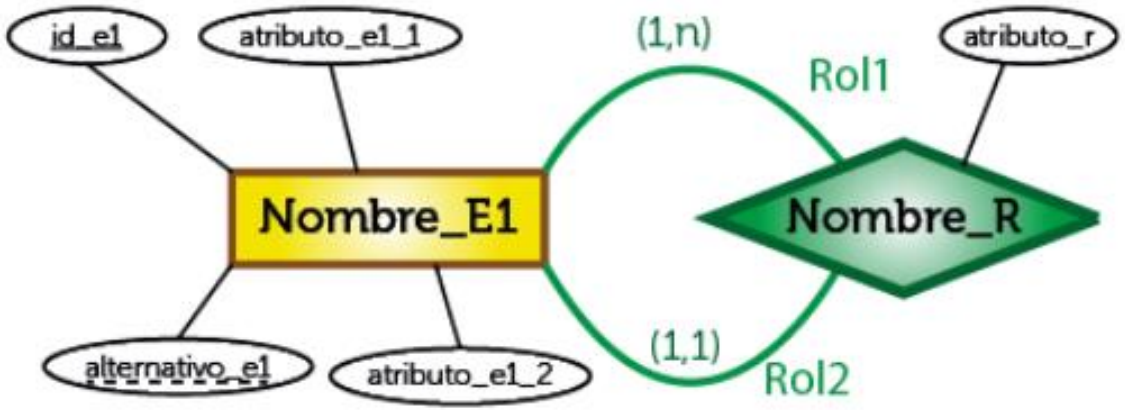
5.4 De modelo E/R a CROW'S FOOT

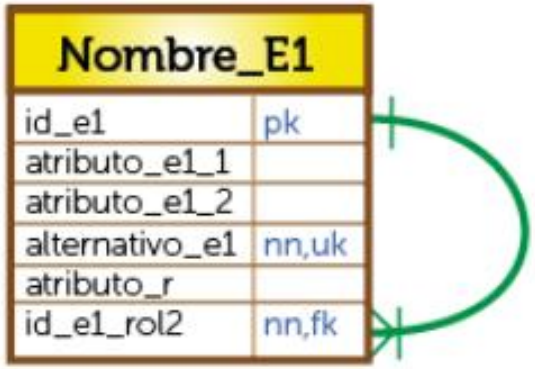
- ***Relaciones unarias/reflexivas:*** (1,N)..(0,N)



5.4 De modelo E/R a CROW'S FOOT

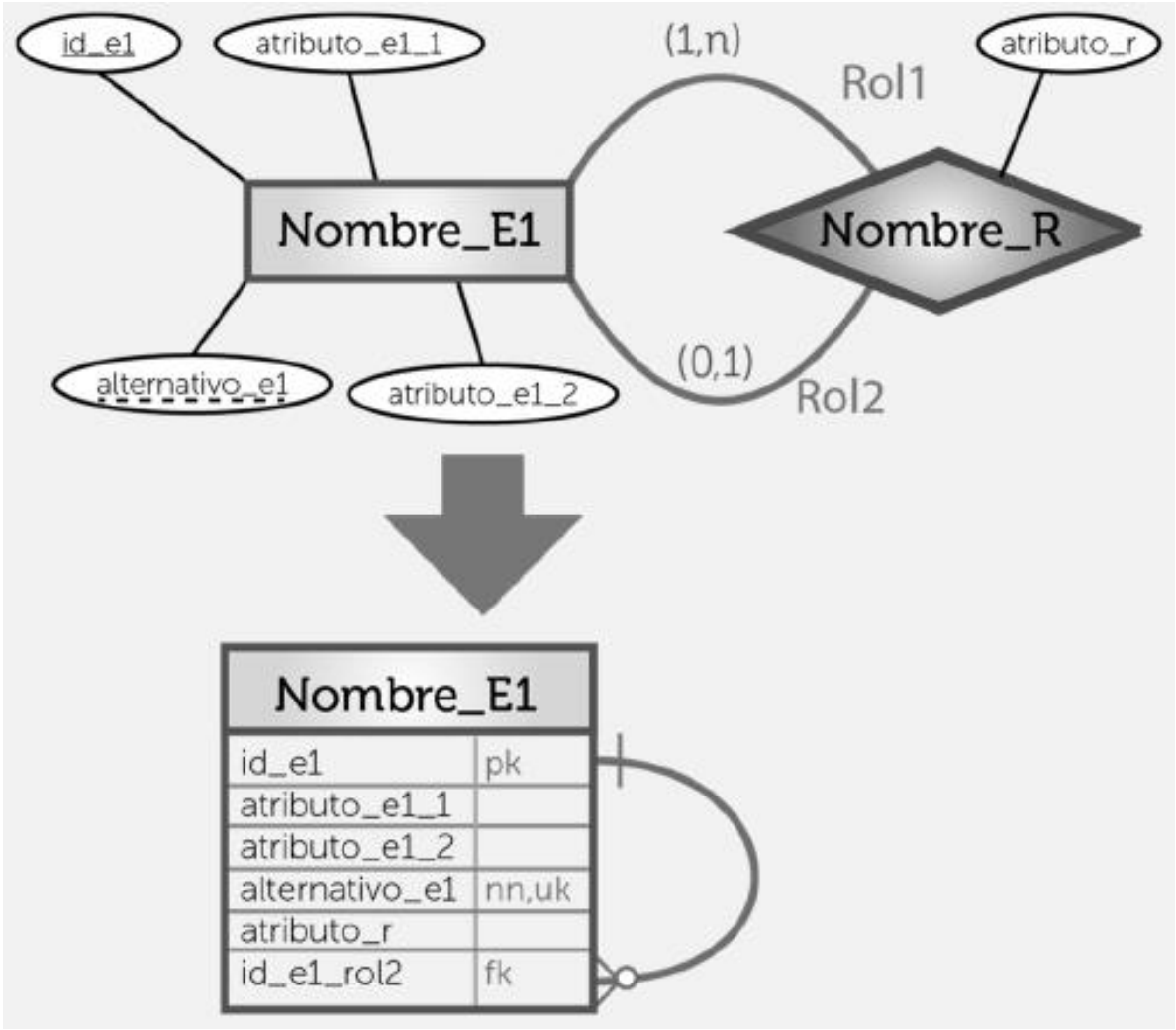
- ***Relaciones unarias/reflexivas:*** (1,N)..(1,1)





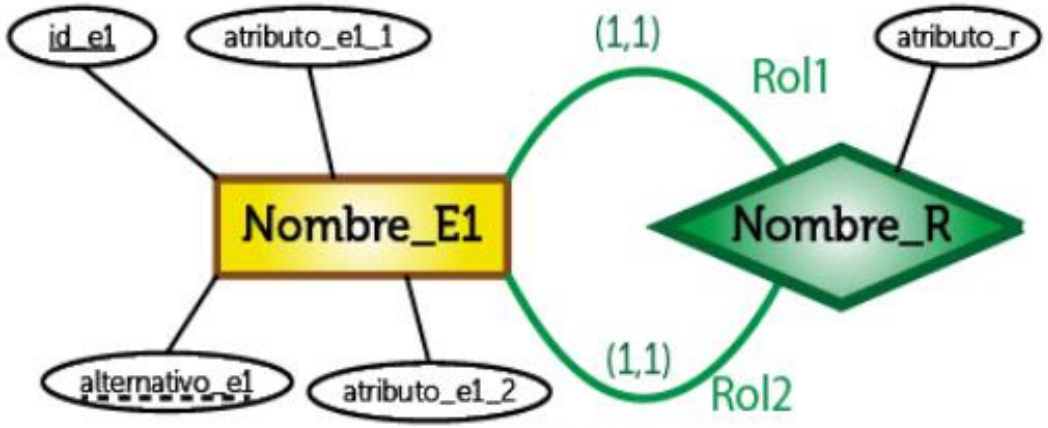
5.4 De modelo E/R a CROW'S FOOT

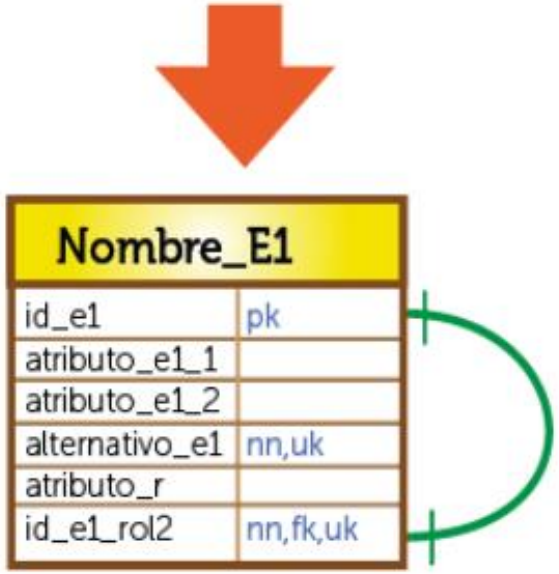
- ***Relaciones unarias/reflexivas:*** (1,N)..(0,1)



5.4 De modelo E/R a CROW'S FOOT

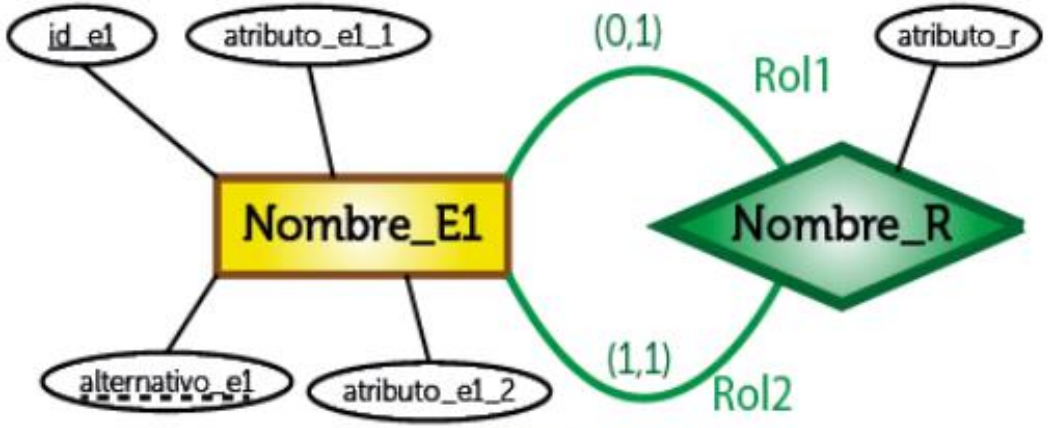
- ***Relaciones unarias/reflexivas:*** (1,1)..(1,1)

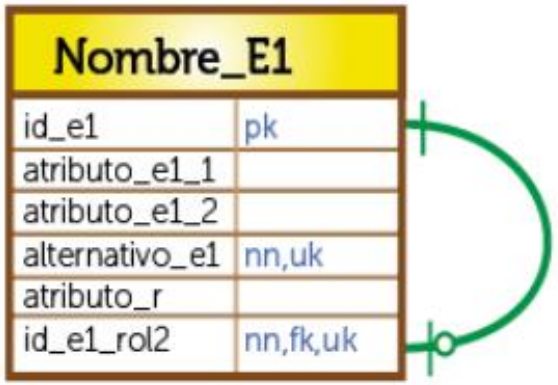




5.4 De modelo E/R a CROW'S FOOT

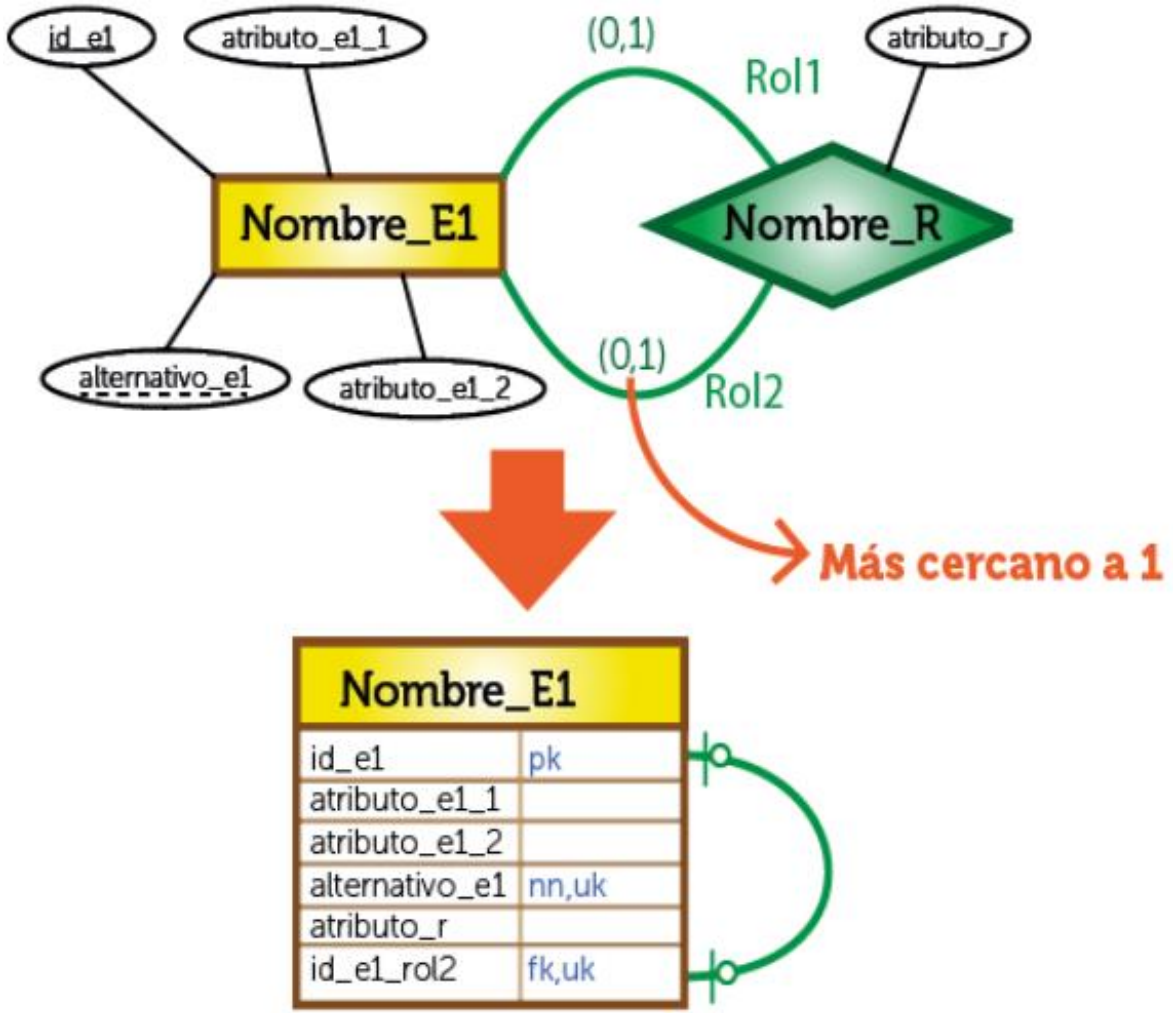
- ***Relaciones unarias/reflexivas:*** (1,1)..(0,1)





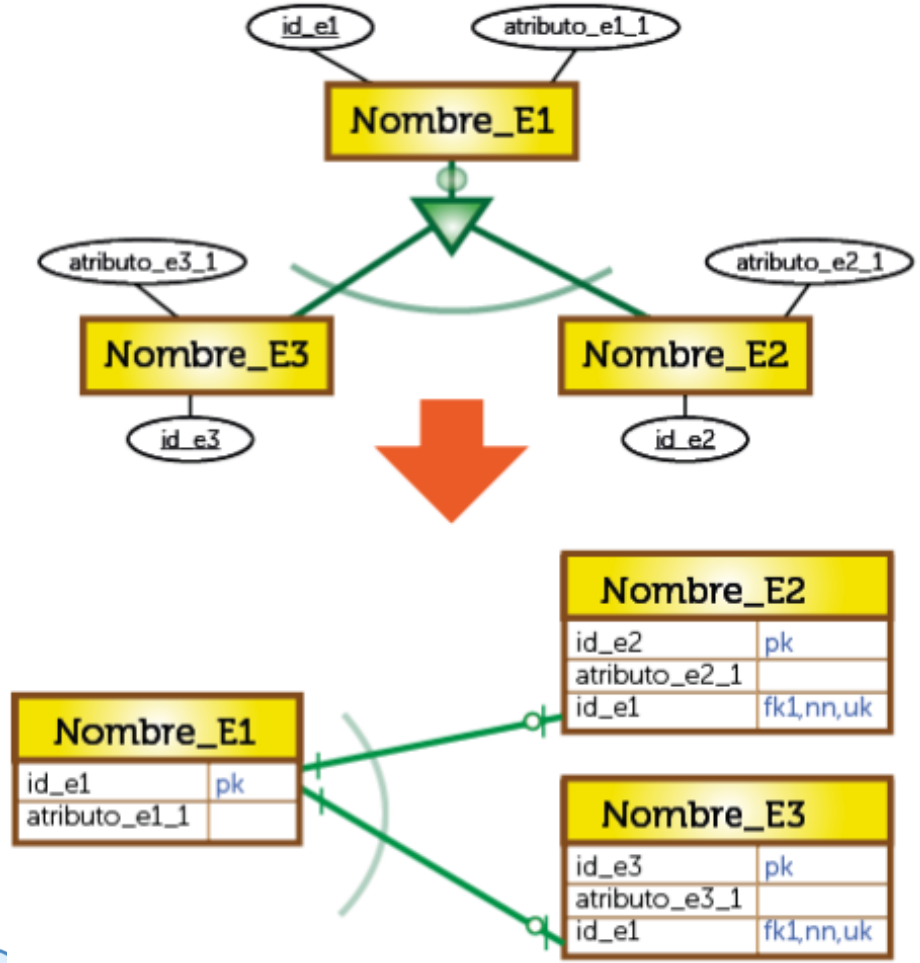
5.4 De modelo E/R a CROW'S FOOT

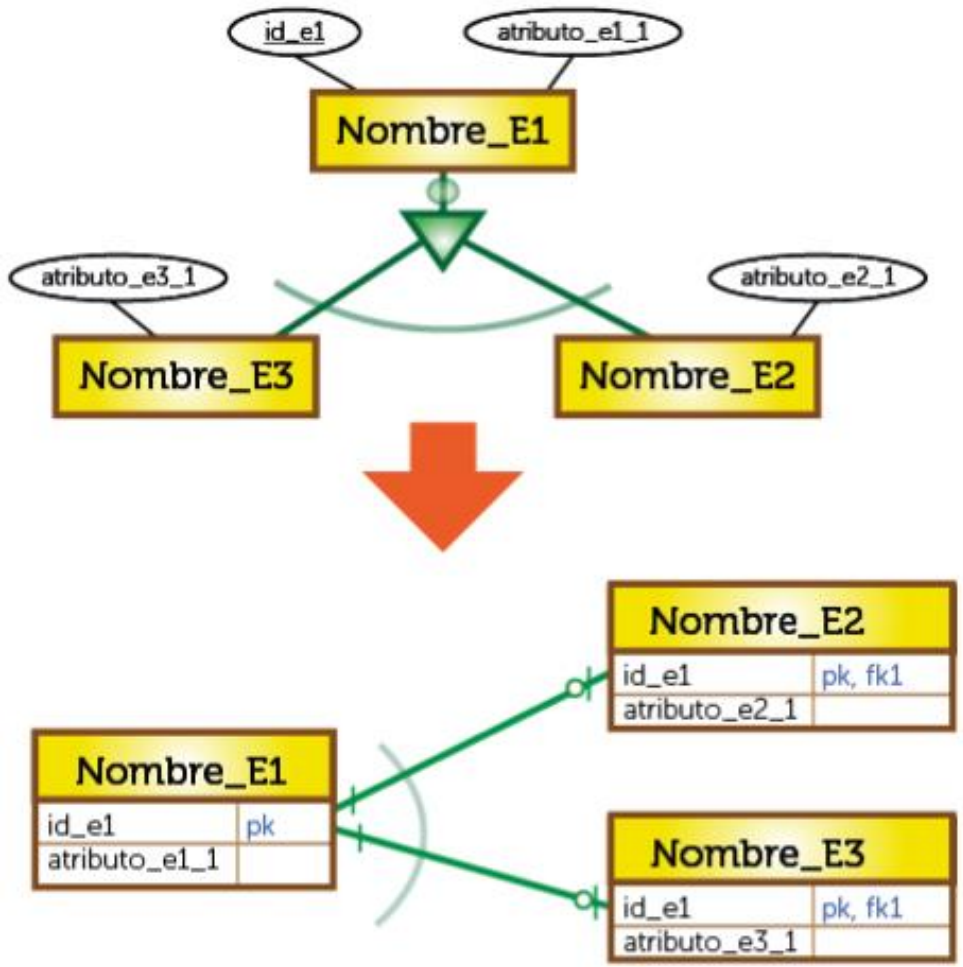
- ***Relaciones unarias/reflexivas:*** (0,1)..(0,1)



5.4 De modelo E/R a CROW'S FOOT

- ***Generalización/especialización***





KAHOOT!