

Repaso: Modelado de Bases de Datos

1. INTRODUCCIÓN	2
1.1. Etapas en el diseño de una BD	2
1.2. ¿Qué es el modelo ER?	3
1.3. Versiones	3
2. MODELADO CON CROW'S FOOT MODEL	4
2.1. Entidades	4
2.2. Relaciones	5
2.3. Ejemplos introductorios	7
3. MODELADO EXTENDIDO CON CROW'S FOOT MODEL	9
3.1. RELACIONES DE DEPENDENCIA	9
3.1.1. Entidad fuerte vs Entidad débil	9
3.1.2. Ejemplo de entidades fuertes relacionadas	11
3.1.3. Relaciones de dependencia de identificación	12
3.1.4. Mezcla de relaciones identificativas y no-identificativas	14
3.2. RELACIONES REFLEXIVAS	16
3.3. HERENCIA o JERARQUIA	17
4. MODELO RELACIONAL	18
4.1. Clave primaria y Clave ajena	18
4.2. Transformación de relaciones 1:N	19
4.3. Transformación de relaciones N:N	19
4.4. Transformación de relaciones reflexivas	19
4.5. Transformación de entidades débiles	19
4.6. Transformación de jerarquías	20
4. RESUMEN	21

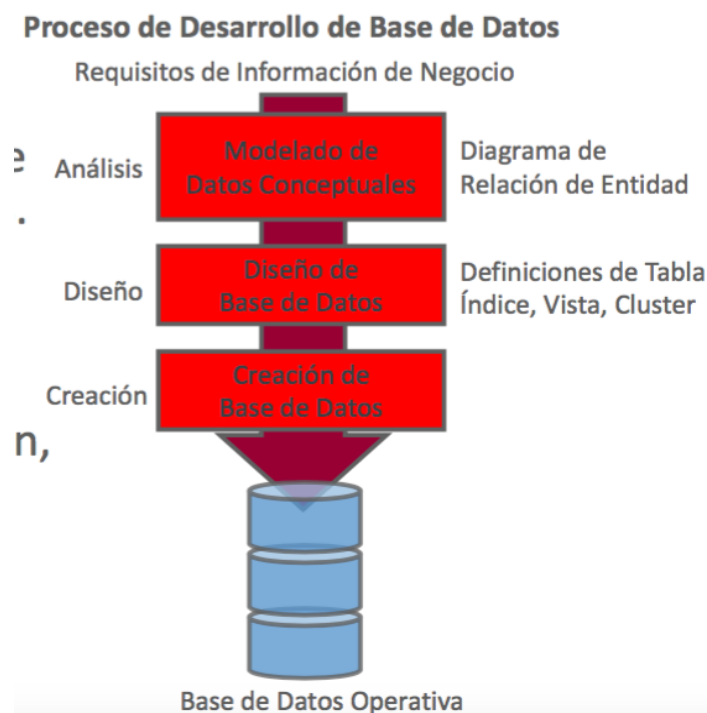
1. INTRODUCCIÓN

1.1. Etapas en el diseño de una BD

El primer paso que se realiza es la **recopilación de información y análisis de requerimientos**, durante el cual los diseñadores entrevistan a los futuros usuarios del Sistema de Base de datos para comprender y redactar los requerimientos de información. El resultado de este paso será un conjunto de requerimientos redactados de forma concisa.

Paralelamente a la especificación de requerimientos de datos, conviene especificar los **requerimientos funcionales** de la aplicación. Se trata de las operaciones de usuarios (transacciones) que se aplicarán en la base de datos e incluye la obtención y actualización de datos.

Una vez realizado el análisis de requerimientos, el siguiente paso es **crear el esquema conceptual** para la base de datos mediante un modelo de datos conceptual de alto nivel. A ese paso se le denomina **diseño conceptual** de la base de datos. El esquema conceptual es una descripción concisa de los requerimientos de información de los usuarios y contiene información sobre las relaciones y restricciones. Este esquema conceptual puede servir como punto de referencia para asegurarse de que se satisfacen todos los requerimientos de los usuarios. Representamos al mundo real en esquemas conceptuales E/R. El esquema E/R viene a ser para una BD como los planos de una casa para un arquitecto.



Una vez que haya terminado el proceso de diseño es necesario empezar a implementar el SBD con un SGBD comercial. La mayoría de los SGBD actuales utilizan un modelo de datos de implementación, por lo que es necesario traducir el modelo de datos de alto nivel al modelo de datos de implementación, lo que se conoce como el proceso de diseño lógico de la base de datos o transformación de modelos de datos. El resultado es un esquema de la base de datos especificado en el modelo de datos de implementación de SGBD. Nosotros transformaremos los esquemas conceptuales E/R en esquemas relacionales.

El último paso es la fase de **diseño físico** de la base de datos, en la que se especifican las estructuras de almacenamiento internas y la organización de los archivos de la base de datos. Además, deben diseñarse e implementarse los programas de aplicación para las transacciones que se recopilaban en el análisis de requerimientos funcionales.

1.2. ¿Qué es el modelo ER?

El modelo ER es un **modelo de datos conceptual de alto nivel** y que suele usarse bastante en el diseño de bases de datos, con independencia de la máquina en la que se implementen. Fue **propuesto por Peter Chen en 1976**. Desde entonces muchos autores se han interesado por él, estudiándolo y ampliándolo, consiguiendo así diversas variantes del modelo (diferentes formas de representación de los objetos), pero todas ellas parten del mismo concepto: el conocimiento del mundo real que se desea representar a través de un análisis de los requisitos o especificaciones del problema.

El modelo ER consiste en un conjunto de conceptos, reglas y notaciones que permiten formalizar la semántica del mundo real que se pretende moldear (también denominada Universo del Discurso) en una representación gráfica o diagrama que denominamos esquema de la Base de Datos.

1.3. Versiones

Al menos 3 versiones distintas del modelo ER son utilizadas actualmente.

Uno de ellos, el modelo Information Engineering (IE), fue desarrollado por James Martin en 1990. Este modelo utiliza “pies de cuervo” para indicar la multiplicidad de una relación, y por tanto, se llama **IE Crow's Foot model**.

En 1993, el National Institute of Standards and Technology propuso otra versión del modelo ER modelo como estándar. Esta versión se llamó **Integrated Definition 1, Extended (IDEF1X)**. Este estándar incorpora las ideas básicas del modelo clásico ER, pero utiliza diferentes símbolos gráficos. Aunque este modelo es un estándar nacional y utilizado en el gobierno de EEUU, es difícil de entender y utilizar.

Al mismo tiempo, para añadir una mayor complejidad, la metodología orientada a objetos **Unified Modeling Language (UML)** adoptó el modelo ER, pero introdujo sus propios símbolos para dar una nueva perspectiva a la programación orientada a objetos.

La existencia de diferencias debido a distintas versiones del modelo ER ha causado confusión y enredos. El problema es que los productos de software de modelado de datos utilizan distintas técnicas. Por ejemplo, Erwin utiliza unas y Microsoft Visio otra. Cuando se crea un modelo de datos, será necesario conocer la versión concreta del modelo ER que se desea utilizar.

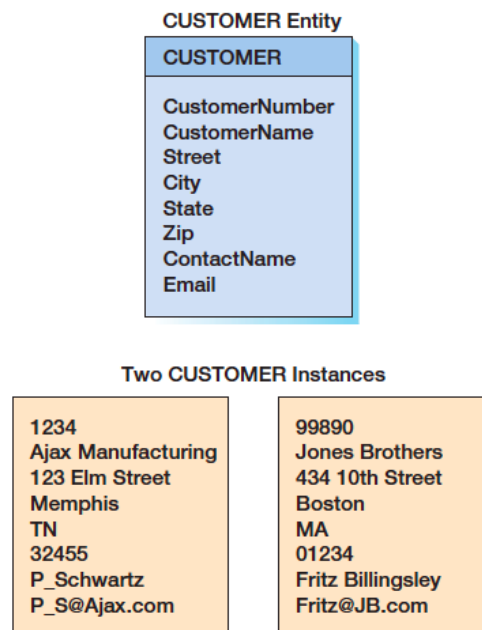
En este documento contrastaremos el modelado con la primera de las técnicas, el IE Crow's Foot.

2. MODELADO CON CROW'S FOOT MODEL

2.1. Entidades

Las **entidades**, también llamadas **tipos de entidad**, representan conjuntos de elementos con existencia propia y que se caracterizan por las mismas propiedades. Generalmente son personas, cosas, lugares,..., es decir, conceptos sobre los que necesitamos guardar información y distinguibles de los demás objetos. Su representación gráfica se realiza por medio de un cuadrado dentro del cual se escribe el nombre de la entidad en mayúsculas (sustantivo) y su lista de características.

Ejemplo: entidad CLIENTE. Cada cliente concreto que almacenen dentro de la BD diremos que es una instancia de la entidad CLIENTE.



Las entidades poseen **atributos** que describen sus características. El modelo ER presupone que todas estas instancias de una clase de entidad concreta tienen sus mismos atributos. En CFM, los atributos se escribirán dentro del mismo rectángulo que la entidad. Las instancias de entidad tienen **identificadores**, que son atributos que llaman o identifican a las instancias de entidad. El identificador de una instancia puede estar compuesto por uno o más atributos de la entidad. Los identificadores formados por dos o más atributos se llaman identificadores compuestos.

Los atributos carecen de existencia propia, es decir, sólo tienen sentido en el esquema de la BD mientras aparecen formando parte de una entidad.

Es importante destacar que un mismo concepto no tiene por qué representarse siempre de la misma forma (por ejemplo, como una entidad o como un atributo). Así, si estuviéramos moldeando una Base de datos para una tienda de ropa, probablemente tendríamos una entidad denominada PIEZA y uno de sus atributos podría ser Color (roja, negra, etc.). Sin embargo, si estuviéramos hablando de una Base de datos para gestionar la información de un taller de vehículos dedicado a trabajos de chapa y pintura, el concepto de color puede tener tal

importancia que pase a ser una entidad COLOR, pues tiene existencia propia y un conjunto de propiedades (código de color, textura, tipos de mezcla, etc.)

Para poder distinguir un ejemplar de otro, dentro de un mismo tipo de entidad, el modelo E/R obliga a que cada vez que definimos un tipo de entidad se defina un atributo que identifique cada ejemplar, es decir, uno **identificador principal (en inglés, UID unique identifier)**. Por tanto en todos los tipos de entidad debe aparecer de forma obligatoria una característica que identifique de forma única cada uno de los ejemplares. Ésta es la representación que nos proporciona el modelo E/R para distinguir este tipo de atributo del resto de atributos que componen el tipo de entidad. En un tipo de entidad sólo puede aparecer un identificador principal, pero pueden existir distintos atributos que también identifiquen los ejemplares de ésta; este tipo de atributos se denominan **identificadores alternativos (IA)**.

2.2. Relaciones

Las entidades pueden estar asociadas con una o más **relaciones**.

En Crow's Foot Model representaremos las relaciones mediante una línea discontinua que una a las entidades implicadas. Opcionalmente se le pueden dar nombres a las relaciones para que las describan. Los nombres para las relaciones serán necesarios cuando existan dos relaciones diferentes entre las propias entidades.

Grado de la relación

Una relación puede implicar asociar a dos o más entidades. El número de entidades implicadas en la relación es el **grado de la relación**. Las relaciones de grado 2 se llaman binarias, mientras que las de grado 3 se llaman ternarias.

Cuando se transforma el modelo de datos en un diseño de base de datos relacional, las relaciones de cualquier grado son tratadas como relaciones binarias y todos los productos de software de modelado de datos requieren la expresión de las relaciones como de tipo binario.

Cardinalidad

Se define como el número máximo de instancias de una entidad que pueden estar asociadas a otra instancia de otra entidad.

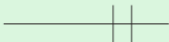
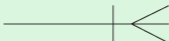
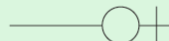

Adicionalmente en el Crow's Foot Model también debemos especificar los valores mínimos existentes entre las relaciones, que tomarán los valores 0 o 1.

Los posibles valores máximos que podrá valer la cardinalidad de la relación entre dos entidades A y B determinarán que la relación tenga:

- **Cardinalidad 1:1** especifica que una entidad A pueda estar vinculada mediante una relación a una y sólo una ocurrencia de otra entidad B. A su vez una ocurrencia de la entidad B sólo puede estar vinculada a una ocurrencia de la entidad A. Por ejemplo, se puede limitar el número de directores de departamento mediante una relación 1:1. Así, un empleado sólo puede ser jefe de un departamento y un departamento sólo puede tener un jefe.
- **Cardinalidad 1:N** especifica que una entidad A puede estar vinculada mediante una relación a varias ocurrencias de otra entidad B. Sin embargo, una de las ocurrencias de la entidad B sólo podrá estar vinculada a una ocurrencia de la entidad A. Por ejemplo, un manager gestiona las carreras de varios actores y un actor sólo podrá tener un representante.
- **Cardinalidad N:N** especifica que una entidad A puede estar vinculada mediante una relación a varias ocurrencias de la entidad B, ya su vez, una ocurrencia de la entidad B puede estar vinculada a varias de la entidad A. Por ejemplo, un empleado puede trabajar

para varios proyectos y al mismo tiempo, en un mismo proyecto pueden trabajar varios empleados.

El CFM utiliza la siguiente notación para indicar la cardinalidad de las relaciones:

Symbol	Meaning
	Mandatory—One
	Mandatory—Many
	Optional—One
	Optional—Many

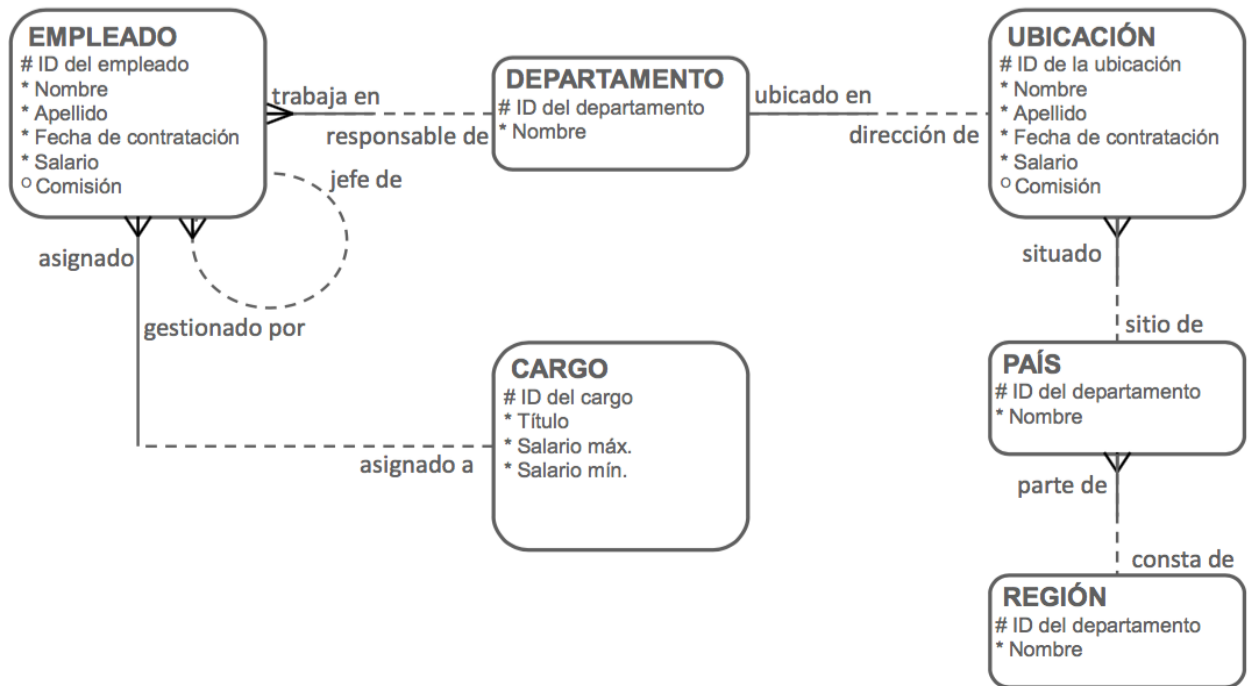
El símbol més pròxim a l'entitat és el màxim de la cardinalitat, mentre que el següent és el mínim. La **marca vertical indica 1** (obligatorietat), el **cercle indica zero** (opcionalitat) i la **pota de corb indica múltiples**.

2.3. Ejemplos introductorios

Ejemplo 1: Departamento RRHH

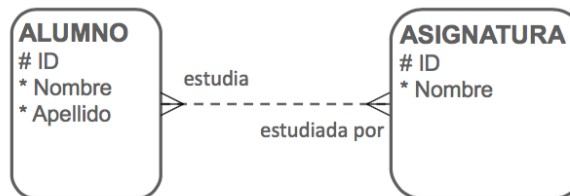
El departamento de recursos humanos de una gran compañía necesita almacenar datos sobre cada uno de sus empleados. Se necesita realizar un seguimiento de su nombre, email y opcionalmente su teléfono. A cada empleado se le asigna un número de empleado único. La compañía está dividida en departamentos. Cada empleado está asignado a un departamento, por ejemplo contabilidad, ventas o desarrollo. Cada departamento tiene un número único.

Si evaluamos un modelo más complejo, donde se tengan en cuenta los cargos, jefes y ubicaciones de los departamentos, el modelo de datos se ampliará siendo la siguiente figura una posible solución:



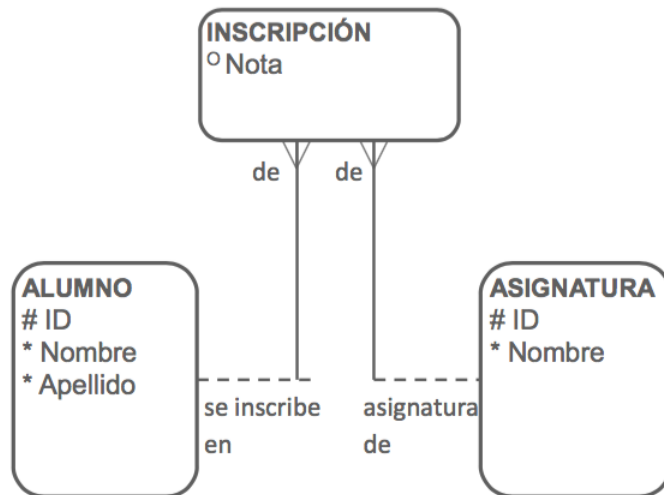
Ejemplo 2: Centro educativo

En un centro educativo un ALUMNO puede estudiar una o varias asignaturas. Cada ASIGNATURA puede ser estudiada por uno o varios alumnos.



Pero, ¿qué pasaría si deseamos poder registrar la nota que consiga un alumno para cada asignatura? ¿A qué entidad pertenecería el atributo "Nota"? Si ponemos "Nota" en la entidad ALUMNO, ¿cómo sabremos para qué ASIGNATURA es? Si ponemos "Nota" en la entidad ASIGNATURA, ¿cómo sabremos qué ALUMNO obtuvo esta nota?

Solución: Agregar una entidad de intersección (INSCRIPCIÓN), que incluye el atributo "Nota". Por tanto, la relación N:N original se ha convertido en dos relaciones 1:N.



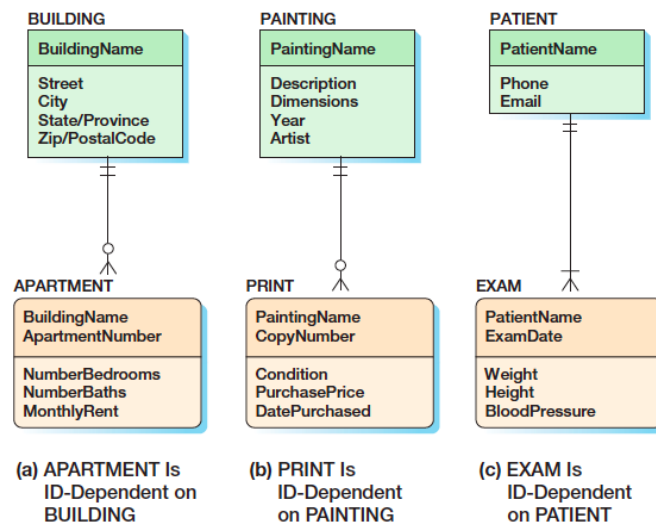
3. MODELADO EXTENDIDO CON CROW'S FOOT MODEL

3.1. RELACIONES DE DEPENDENCIA

3.1.1. Entidad fuerte vs Entidad débil

Una entidad fuerte es aquella que representa algo que puede existir de forma independiente.

El modelo ER incluye un tipo especial de entidad débil llamada **ID-dependent entity** que depende de la existencia de una entidad fuerte. Una ID-dependent entity es una entidad cuyos identificadores incluyen el identificador de la entidad de la que depende.



En cada uno de los ejemplos, la entidad débil no puede existir sin que su padre (cuya entidad fuerte depende) exista. Por tanto, la cardinalidad mínima de la entidad débil hacia su padre siempre será 1.

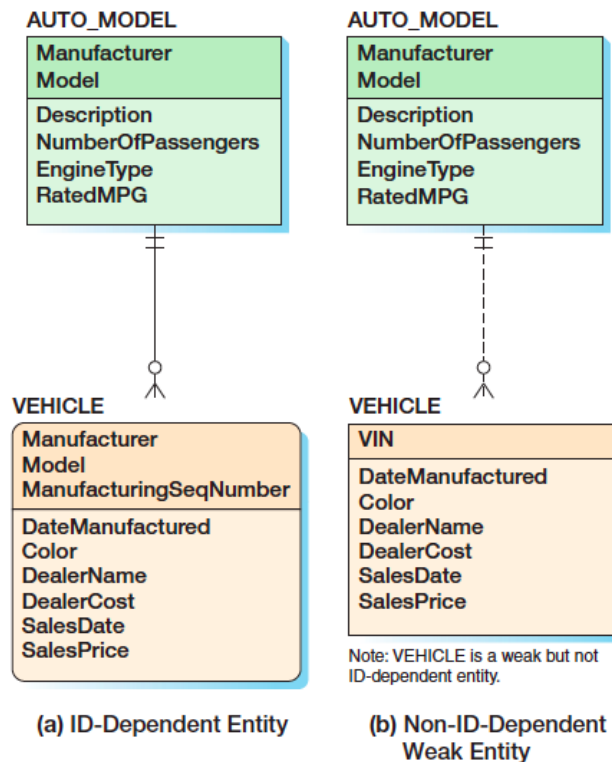
Sin embargo, el padre también podría requerir tener una entidad débil siempre asociada según los requerimientos de la aplicación. Por ejemplo, APARTMENT y PRINT son opcionales, pero EXAM es requerido. Las restricciones vendrán dadas por la naturaleza de la aplicación y no requerimiento lógico alguno.

En el modelo ER, utilizaremos una entidad con las **esquinas redondeadas** para representar a las entidades débiles y una línea sólida para representar la relación entre la entidad débil y la fuerte. Este tipo de relación se llama **relación de identificación**. Cuando la relación se pinta con una línea discontinua entre dos entidades, esto significa que las entidades son no-dependientes en la relación.

Las entidades débiles establecen restricciones en el proceso de creación de la base de datos que se construirá en base al modelo. Las filas que representen a la entidad padre tendrán que ser creadas antes de que cualquier fila hija pueda ser creada. Además, cuando una fila padre sea borrada, todas las filas hijas también tendrán que serlo.

En el modelo ER, también podríamos tener entidades débiles que no dependen de la identificación de la otra entidad fuerte y diremos que no son ID-dependiente y tienen una **relación de existencia**.

En el siguiente ejemplo se puede ver la diferencia:



Si se utiliza un VIN (Vehicle Identification Number), entonces VEHÍCULO tiene un identificador único y no precisa ser identificado por su relación con AUTO_MODEL. Por tanto, VEHÍCULO es no ID-dependiente. Eso sí, debemos recordar que VEHÍCULO es un AUTO_MODEL, y si un AUTO_MODEL particular no existe, entonces, su VEHÍCULO asociado tampoco podrá existir. Por tanto, VEHÍCULO es una entidad débil con una relación de dependencia existencial, pero no de identificación (non-ID-dependent entity)

Entonces, ¿cuándo escogemos entre una entidad débil con dependencia de identificación o sólo existencial?

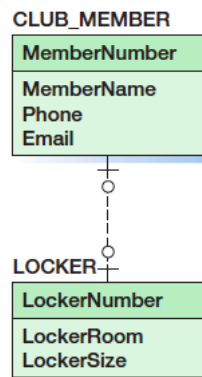
Una entidad débil es una entidad cuya existencia depende de otra entidad:

- Una ID-dependent entity es una entidad débil, cuyo identificador incluye el identificador de la entidad fuerte.
- Las relaciones de identificación (con líneas continuas) son utilizadas para mostrar ID-dependent entities.
- Algunas entidades son débiles, pero no ID-dependient. Utilizando las herramientas del modelado de datos, podremos mostrar estos dos tipos diferentes de relaciones, identificación y existencial.

3.1.2. Ejemplos de entidades fuertes relacionadas

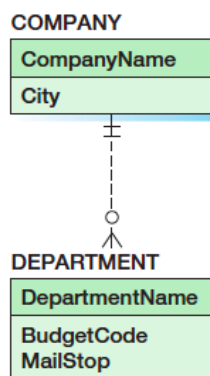
Cardinalidad 1:1

Para modelar este tipo de relación, utilizaremos una nonidentifying relationship (fuerte y not ID-dependent) mediante una línea discontinua entre las dos entidades. Las cardinalidades máximas serán de 1:1.

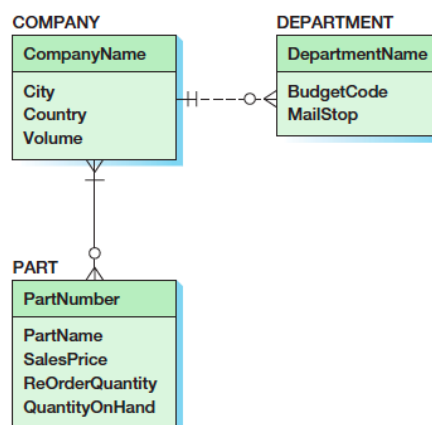


Es poco probable que todas las taquillas estén ocupadas, y habrá algunas que estén sin utilizar.

Cardinalidad 1:N



Cardinalidad N:N



Una COMPAÑÍA (vista como un proveedor) suministra muchos ítems (PARTS), y un ítem es suministrado por muchos proveedores. Por tanto, la relación es N:N. Debido a que no todas las empresas son los proveedores, la relación de la compañía con la PARTE debe ser opcional. Sin embargo, cada parte debe ser suministrada desde un lugar, por lo que la relación con COMPANY es obligatoria.

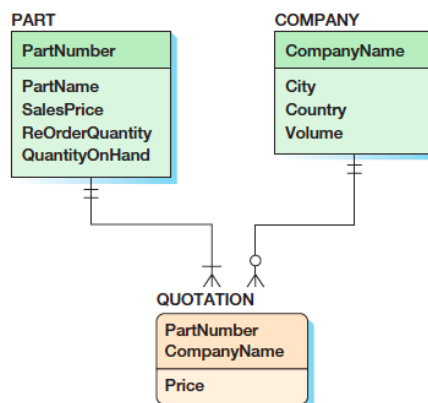
3.1.3. Relaciones de dependencia de identificación

Dos patrones destacan en el uso de entidades ID-dependent: 'Asociación' y 'Atributos multivaluados'.

Patrón de Asociación

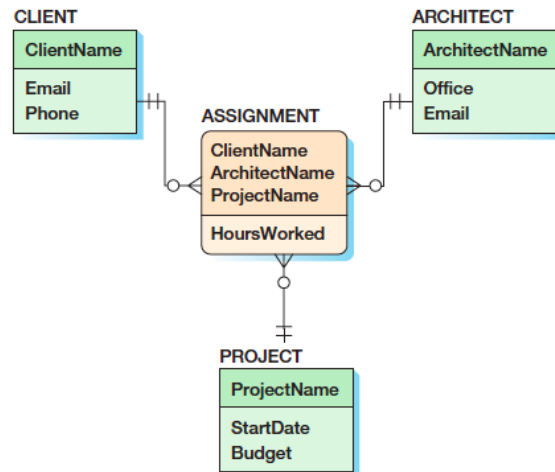
El patrón de Asociación es sutil y confusamente similar a una relación fuerte N:N.

En el siguiente ejemplo vemos que el atributo precio no es ni atributo de la COMPANY ni del ITEM, sino de su combinación:



Al igual que en las relaciones de identificación, las entidades fuertes o padres son obligatorias. Por tanto, la cardinalidad mínima por QUOTATION (cotización) hacia PART (ítem) y del PART hacia COMPANY es 1. La cardinalidad mínima en el sentido contrario viene determinado por los requerimientos de negocio. En este caso, un ÍTEM deberá tener una QUOTATION, pero una COMPANY no tiene porqué tener una QUOTATION.

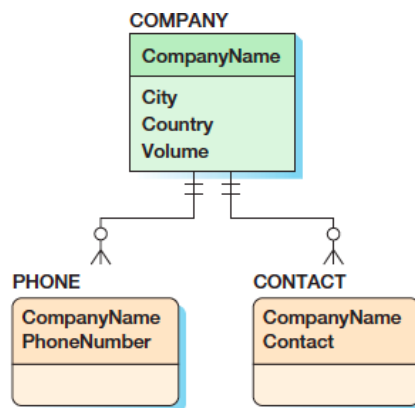
Las asociaciones pueden producirse entre más de dos tipos de entidad. Por ejemplo, en la siguiente captura se muestra un modelo de datos para la asignación de un cliente en particular a un arquitecto en particular para un proyecto en particular. El atributo de la asignación es hoursWorked.



Patrón de atributos multivaluados

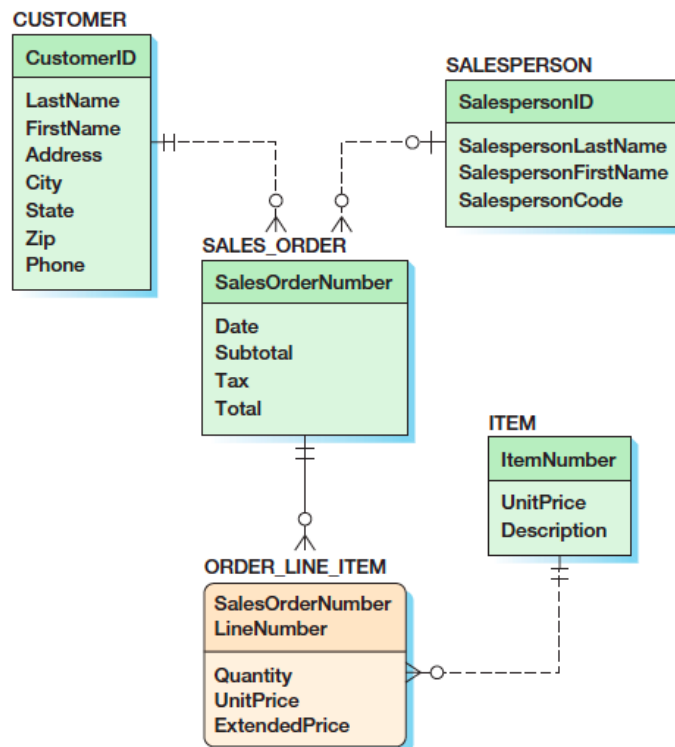
En el modelo ER, los atributos siempre tendrán que tener un único valor.

A continuación se muestra como representar los atributos multivaluados teléfono y contacto de una compañía:



3.1.4. Mezcla de relaciones identificativas i no –identificativas

Ejemplo 1

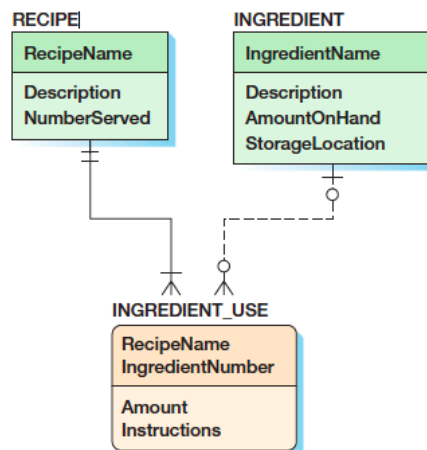


Estas formas suelen tener datos sobre el propio pedido, como el número de pedido y su fecha, los datos sobre el cliente, los datos sobre el vendedor, y datos sobre los asuntos relacionados con el pedido.

La parte importante recae sobre las líneas de los pedidos. Algunos de los valores de los datos pertenecen al mismo pedido, pero otros valores de datos pertenecen a los elementos en general. En particular, la *Quantity* y *ExtendedPrice* pertenecen al *SALES_ORDER*, pero *ItemNumber*, *Description* y *UnitPrice* pertenecen al ítem.

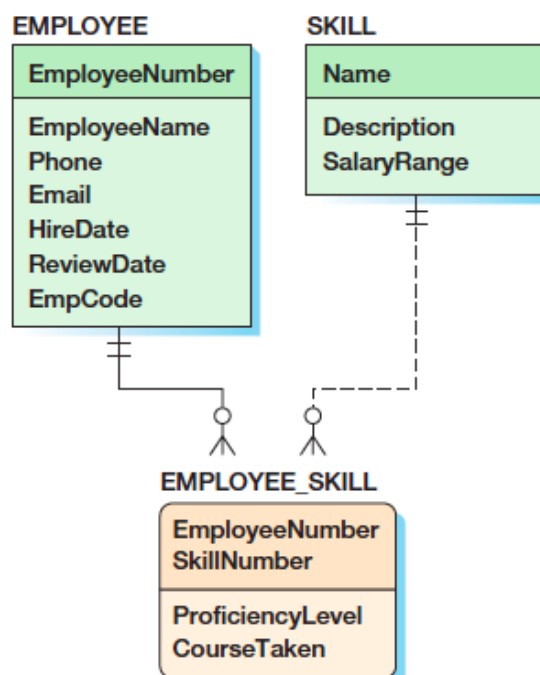
Ejemplo 2

Cada receta requiere de una cierta cantidad de un ingrediente específico, como la harina, el azúcar o la mantequilla. La lista de ingredientes es un grupo compuesto de varios valores, pero uno de los elementos de ese grupo, el nombre del ingrediente, es el identificador de una entidad fuerte. La receta y los ingredientes son entidades fuertes, pero la cantidad e instrucciones para utilizar cada ingrediente son dependent-ID de la receta.



Ejemplo 3

Consideramos las habilidades de los empleados. El nombre de la habilidad y el nivel de competencia que tiene el empleado son un grupo de varios valores, pero la propia habilidad es una entidad fuerte.



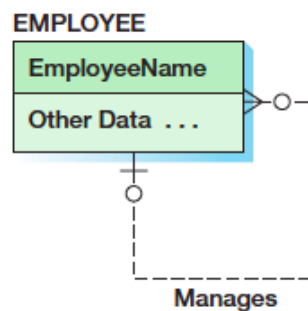
3.2. RELACIONES REFLEXIVAS

Una relació reflexiva és aquella que associa dues entitats del mateix tipus.

Per exemple, quan un treballador pot gestionar a un altre grup de treballadors (zero o molts) i un treballador té un únic cap (o potser no té cap), llavors tindrem una relació reflexiva 1:N com la següent:

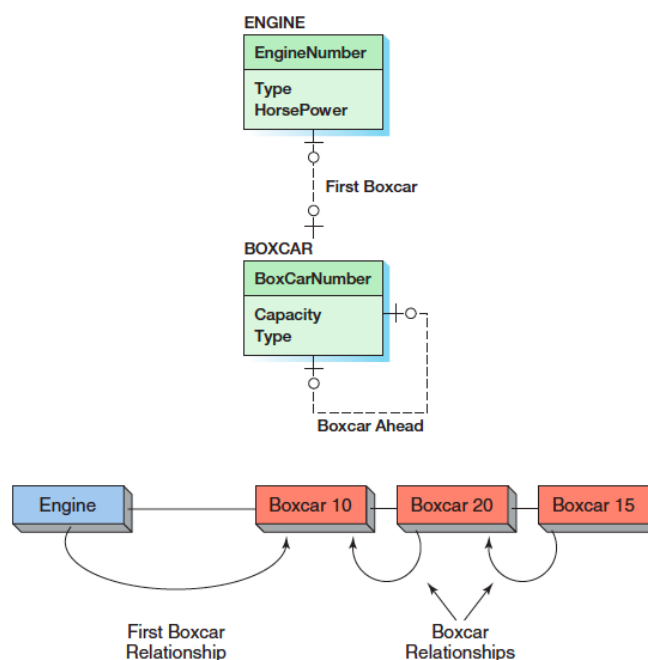
Una relación reflexiva es aquella que asocia a dos entidades del mismo tipo.

Por ejemplo, cuando un trabajador puede gestionar a otro grupo de trabajadores (cero o muchos) y un trabajador tiene una única cabeza (o quizás no tiene ninguna), entonces tendremos una relación reflexiva 1:N como la siguiente:



En la siguiente figura, se muestra un modelo de datos sobre un tren de carga, en el que cada vagón (BOXCAR) que sigue a la locomotora (ENGINE) tiene una relación reflexiva 1:1 con su siguiente vagón. Es decir, todo vagón podrá o no tener un vagón previo y otro posterior.

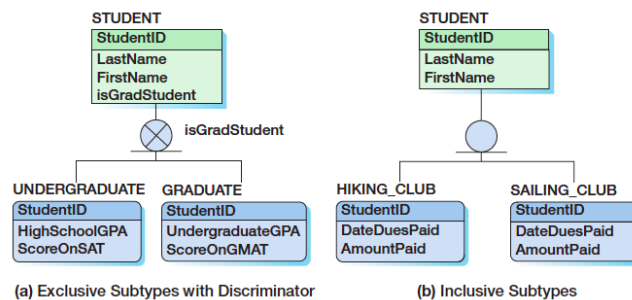
Este modelo asume que un tren tiene una sola locomotora en un extremo. Para moldear los trenes con varias locomotoras, crearíamos una segunda relación recursiva entre los motores.



3.3. HERENCIA o JERARQUIA

El modelo ER extendido introduce el concepto de subtipo. Una entidad subtipo es un caso especial de otra entidad llamada supertipo.

Los estudiantes, por ejemplo, pueden ser clasificados como estudiantes de grado o posgrado. En este caso, el estudiante es el supertipo, y de grado y posgrado son los subtipos.



En nuestros modelos ER utilizamos un círculo para indicar una relación supertipo-subtipo. En caso de que marquemos una línea recta debajo del círculo representaremos que la relación es 1:1 (total) de forma que no podremos tener supertipos que no sean de ningún subtipo.

Adicionalmente, los subtipos pueden ser exclusivos o inclusivos. Con subtipos exclusivos, una instancia del supertipo está relacionada como máximo con un subtipo y lo representaremos con una cruz sobre el círculo de la jerarquía. Con subtipo inclusivo, una instancia supertipo puede relacionarse con uno o más subtipos.

Ejemplo (a): la X en el círculo significa que los subtipos de grado y posgrado son exclusivos. Por tanto, un estudiante puede ser o bien un estudiante de grado o uno de posgrado, pero no ambos a la vez.

Ejemplo (b): muestra que un estudiante puede unirse a cualquier club, al **HIKING_CLUB** o **SAILING_CLUB**, o a ambos. Estos subtipos son inclusivos (no hay una X en el círculo). Como un supertipo puede referirse a más de un subtipo, los subtipos inclusivos no tienen un discriminador.

Lo más importante (algunos dirían que la única) razón para crear subtipos en un modelo de datos es evitar valores nulos excesivos.

3.3.1. Tipus de jerarquies

- **Especialización Exclusiva:** en este caso, cada una de las ocurrencias de la superclase sólo puede materializarse en una de las especializaciones. Por ejemplo, si tenemos una jerarquía **EMPLEADO** puede ser **DIRECTIVO**, **TÉCNICO** o **COMERCIAL**, entonces los empleados sólo podrán ser de un tipo subclase concreto. En Crow's Foot, la especialización exclusiva se representará con un aspa sobre el símbolo de la jerarquía.
- **Especialización Inclusiva:** se produce cuando las ocurrencias de la superclase pueden materializarse a la vez en varias ocurrencias de las subclases. En ese caso, el empleado directivo podría también ser técnico y/o comercial. En Crow's Foot, la especialización inclusiva se representará sin aspa sobre el símbolo de la jerarquía.

- **Especialización Total:** se produce cuando la entidad superclase debe materializarse obligatoriamente en una de las especializaciones. No podremos tener empleados que no sean directores o técnicos o comerciales.
- **Especialización Parcial:** la entidad superclase no tiene porque materializarse en una de las especializaciones (es opcional).

4. MODELO RELACIONAL

El modelo de datos conceptual se transformará en un diseño de base de datos relacional.

Esto significa que nuestras entidades, atributos, relaciones e identificadores únicos se convertirán en objetos en una base de datos relacional.

Una base de datos relacional es una base de datos que el usuario ve como una recopilación de tablas bidimensionales, donde cada una contiene filas y columnas.

La siguiente tabla contiene datos de empleados:

EMPLOYEES (nombre de tabla)

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
200	Jennifer	Whalen	10
205	Shelley	Higgins	110

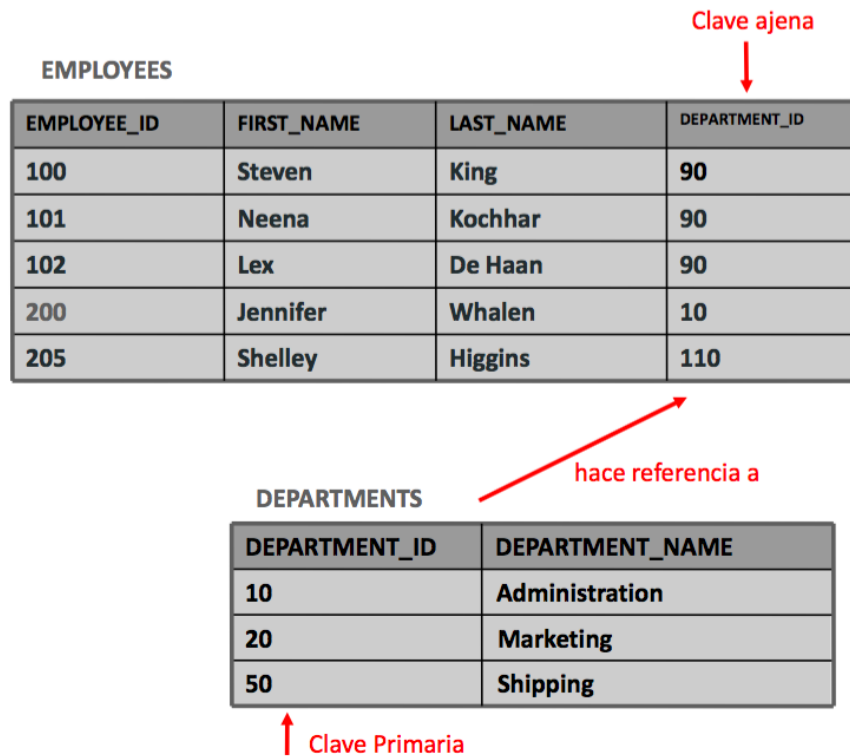
Fila

Columna

4.1. Clave Primaria (PK, Primary Key) y Clave Ajena (Foreign Key, FK)

La **clave primaria (PK, primary key)** es una columna o juego de columnas que identifica de forma única cada fila de una tabla. Toda tabla debe tener una clave primaria y debe ser única. Una clave primaria permite acceder a un registro concreto de una base de datos (sin generar confusiones ni conflictos puesto que es único y no nulo).

Una **clave ajena (FK, foreign key)** es una columna o combinación de columnas de una tabla que contiene valores que coinciden con el valor de clave primaria de otra tabla.



4.2. Transformación de relaciones 1:N

A partir de ahora, en todos nuestros modelos deberemos añadir a la tabla que se corresponde con el lado N, una FK que haga referencia a la PK de la tabla del lado 1 con la que está asociada.

4.3. Transformación de relaciones N:N

Se crea una nueva tabla que tendrá como clave primaria la concatenación de los dos identificadores principales de las entidades que asocia. El orden de los identificadores se elegirá según se considere más adecuado.

Además, cada uno de los atributos que forman la clave primaria de esta tabla también son claves ajenas que referencian a las tablas en que se han convertido las entidades relacionadas.

4.4. Transformación de relaciones reflexivas

Las relaciones reflexivas se tratan de la misma forma que las demás, sólo que un mismo atributo puede figurar dos veces en una tabla como resultado de la transformación.

4.5. Transformación de entidades débiles

Toda entidad débil incorpora una relación implícita con su entidad fuerte. Esta relación no necesita incorporarse como tabla al modelo relacional (al tratarse de una relación N:1), bastará con añadir como atributo y clave ajena el identificador de la entidad fuerte en la entidad débil. En este supuesto, se obliga a una modificación y un borrado en cascada.

4.6. Transformación de jerarquías

Para transformar las jerarquías se puede optar por 4 opciones. Cada opción se adaptará mejor o peor a los distintos tipos de especializaciones:

Solución 1. Se puede crear una tabla para la superclase y otras tantas para cada subclase, incorporando el campo clave de la superclase en las tablas de las subclases:

Empleados(idEmpleado, nombre, dni)

Directivos(idEmpleado, departamento)

Tecnicos(idEmpleado, maquinas)

Comerciales(idEmpleado, comision)

Solución ideal para una especialización **PARCIAL** (independientemente de que sea exclusiva o inclusiva)

Solución 2. Se puede crear una tabla para cada subclase incorporando todos los atributos de la clase padre, y no crear una tabla para la superclase.

Directivos (idEmpleado, nombre, dni, departamento)

Tecnicos (idEmpleado, nombre, dni, maquinas)

Comerciales (idEmpleado, nombre, dni, comision)

Solución ideal para una especialización **TOTAL Y EXCLUSIVA**

Solución 3. Se puede crear una sola tabla para la superclase, incorporando los atributos de todas las subclases y añadir, para distinguir el tipo de la superclase, un campo llamado “tipo”, que contendrá el tipo de la subclase al que representa cada una tupla:

Empleados (idEmpleado, nombre, dni, departamento, maquinas, comision, tipo)

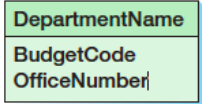
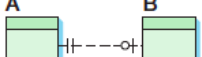
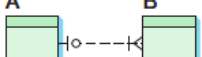
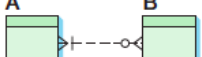
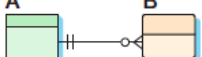
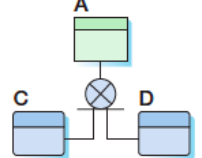
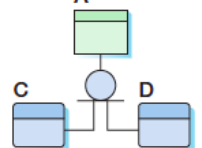
Solución ideal para una especialización **EXCLUSIVA** (independientemente de que sea TOTAL o PARCIAL).

Solución 4. Se puede crear una sola tabla para la superclase como en la opción anterior, pero en lugar de añadir un solo campo “tipo”, se añaden varios campos que indican si se cumple o no un perfil, de este modo, se soportan las especializaciones inclusivas.

Empleados (idEmpleado, nombre, dni, departamento, maquinas, comision, esdirectivo, estecnico, escomercial)

Solución ideal para una especialización **INCLUSIVA** (independientemente de que sea TOTAL o PARCIAL)

5. RESUMEN

	<p>DEPARTMENT entity; DepartmentName is identifier; BudgetCode and OfficeNumber are attributes.</p>
	<p>1:1, nonidentifying relationship. A relates to zero or one B; B relates to exactly one A. Identifier and attributes not shown.</p>
	<p>1:N, nonidentifying relationship. A relates to one or many Bs; B relates to zero or one A. Identifier and attributes not shown.</p>
	<p>Many-to-many, nonidentifying relationship. A relates to zero or more Bs; B relates to one or more As.</p>
	<p>1:N identifying relationship. A relates to zero, one, or many Bs. B relates to exactly one A. Identifier and attributes not shown. For identifying relationships, the child must always relate to exactly one parent. The parent may relate to zero, one, many, or a combination of these minimum cardinalities.</p>
	<p>A is supertype, C and D are exclusive subtypes. Discriminator not shown. Identifier and attributes not shown.</p>
	<p>A is supertype, C and D are inclusive subtypes. Identifier and attributes not shown.</p>