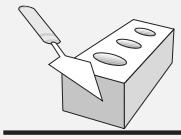

2

INTRODUCCIÓN AL DISEÑO DE BASES DE DATOS

- ¿Qué pasos se siguen en el diseño de bases de datos?
- ¿Por qué se utiliza el modelo ER para crear el diseño inicial?
- ¿Cuáles son los conceptos principales del modelo ER?
- ¿Cuáles son las directrices para un empleo eficaz del modelo ER?
- ¿Cómo encaja el diseño de bases de datos en el marco del diseño global de software complejo en las grandes empresas?
- ¿Qué es UML y qué relación tiene con el modelo ER?
- **Conceptos fundamentales:** diseño de bases de datos, diseño conceptual, lógico y físico; modelo entidad-relación (ER), conjunto de entidades, conjunto de relaciones, atributo, ejemplar, clave; restricciones de integridad, relaciones de una a varias y de varias a varias, restricciones de participación; entidades débiles, jerarquías de clases, agregación; UML, diagramas de clases, diagramas de bases de datos, diagramas de componentes.

> Los grandes hombres de éxito del mundo han empleado su imaginación. Se adelantan en su pensamiento, crean su propia imagen mental y luego trabajan para materializar esa imagen en todos sus detalles, rellenando aquí, añadiendo un poco allá, alterando esa pizca y aquélla, pero trabajando sin pausa, construyendo sin pausa.
>
> — Robert Collier

El *modelo de datos entidad-relación (ER)* permite describir los datos implicados en empresas reales en términos de objetos y de sus relaciones, y se emplea mucho para desarrollar el diseño preliminar de las bases de datos. Aporta conceptos útiles que permiten pasar de una descripción informal de lo que los usuarios desean de su base de datos a otra más detallada y precisa que se pueda implementar en un SGBD. En este capítulo se presenta el modelo ER

y se estudia el modo en que sus características permiten modelar fielmente una amplia gama de datos.

Se comenzará con una introducción al diseño de bases de datos en el Apartado 2.1 con objeto de motivar el estudio del modelo ER. Dentro del contexto más general del proceso global de diseño, el modelo ER se emplea en la fase denominada *diseño conceptual de bases de datos*. Luego se introducirá el modelo ER en los apartados 2.2, 2.3 y 2.4. En el Apartado 2.5 se estudian aspectos del diseño de bases de datos que afectan al modelo ER. El diseño conceptual de bases de datos para grandes empresas se trata brevemente en el Apartado 2.6. En el Apartado 2.7 se presenta una visión general de UML, un enfoque de diseño y modelado de ámbito más general que el modelo ER.

En el Apartado 2.8 se presenta el estudio de un caso que se emplea como ejemplo a lo largo de todo el libro. El caso que se estudia es el diseño de principio a fin de una base de datos para una tienda en Internet. Se ilustran los dos primeros pasos del diseño de bases de datos (análisis de requisitos y diseño conceptual) en el Apartado 2.8. En capítulos posteriores se amplía este caso de estudio para tratar el resto de pasos del proceso de diseño.

Es de destacar que se han utilizado muchas variaciones de los diagramas ER y que no se ha impuesto ninguna norma generalmente aceptada. La presentación de este capítulo es representativa de la familia de modelos ER e incluye una selección de las características más populares.

2.1 DISEÑO DE BASES DE DATOS Y DIAGRAMAS ER

Se comienza el estudio del diseño de bases de datos mediante la observación de que no suele tratarse más que de una parte, aunque fundamental para las aplicaciones que hacen amplio uso de los datos, del diseño de sistemas de software de mayor alcance. No obstante, el principal centro de atención de este libro es el diseño de la base de datos, y sólo se tratarán por encima otros aspectos del diseño de software. Este punto se vuelve a tratar en el Apartado 2.7.

El proceso de diseño de bases de datos puede dividirse en seis etapas. El modelo ER es muy relevante para los tres primeros pasos.

1. **Análisis de requisitos.** El primer paso del diseño de aplicaciones de bases de datos es comprender los datos que se deben guardar en la base de datos, las aplicaciones que se deben construir sobre ellos y las operaciones que son más frecuentes e imponen requisitos de rendimiento. En otras palabras, hay que averiguar lo que los usuarios esperan de la base de datos. Se trata normalmente de un proceso informal que supone discusiones con grupos de usuarios, un estudio del entorno operativo en vigor y del modo en que se espera que éste cambie, el análisis de toda la documentación disponible sobre las aplicaciones existentes que se espera que la base de datos sustituya o complemente, etc. Se han propuesto varias metodologías para la organización y presentación de la información recogida en esta etapa, y se han desarrollado algunas herramientas automatizadas para soportar el proceso.

2. **Diseño conceptual de bases de datos.** La información reunida en el análisis de requisitos se emplea para desarrollar una descripción de alto nivel de los datos que se van a guardar en la base de datos, junto con las restricciones que se sabe que se impondrán sobre esos datos. Este paso se suele llevar a cabo empleando el modelo ER y se discute en el resto

Herramientas para el diseño de bases de datos. Los fabricantes de SGBDR y otros fabricantes ofrecen herramientas de diseño. Por ejemplo, véase el siguiente enlace para conocer más detalles sobre las herramientas de diseño y análisis de Sybase:
`http://www.sybase.com/products/application_tools`
El siguiente ofrece detalles sobre las herramientas de Oracle:
`http://www.oracle.com/tools`

de este capítulo. El modelo ER es uno de los modelos de datos de alto nivel, o **semánticos**, empleados en el diseño de bases de datos. El objetivo es crear una descripción sencilla de los datos que se acerque mucho a lo que los usuarios y los desarrolladores piensan de los datos (y de la gente y de los procesos que se van a representar con esos datos). Esto facilita la discusión entre todas las personas implicadas en el proceso de diseño, aunque no tengan formación técnica. Al mismo tiempo, el diseño inicial debe ser lo bastante preciso como para permitir una traducción directa a un modelo de datos soportado por algún sistema comercial de bases de datos (lo que, en la práctica, significa el modelo relacional).

3. **Diseño lógico de bases de datos.** Hay que escoger un SGBD que implemente nuestro diseño de la base de datos y transformar el diseño conceptual de la base de datos en un esquema de base de datos del modelo de datos del SGBD elegido. Sólo se considerarán SGBD relacionales y, por tanto, la tarea en la etapa de diseño lógico es convertir el esquema ER en un esquema de base de datos relacional. Este paso se examinará con detalle en el Capítulo 3; el resultado es un esquema conceptual, a veces denominado **esquema lógico**, del modelo de datos relacional.

2.1.1 Más allá del diseño ER

El diagrama ER no es más que una descripción aproximada de los datos creada mediante la evaluación subjetiva de la información reunida durante el análisis de requisitos. A menudo, un análisis más detenido puede refinar el esquema lógico obtenido al final del Paso 3. Una vez se dispone de un buen esquema lógico hay que tomar en consideración los criterios de rendimiento y diseñar el esquema físico. Finalmente, hay que abordar los aspectos de seguridad y garantizar que los usuarios puedan tener acceso a los datos que necesiten, pero no a los que se les desee ocultar. Las tres etapas restantes del diseño de bases de datos se describen brevemente a continuación:

4. **Refinamiento de los esquemas.** El cuarto paso del diseño de bases de datos es el análisis del conjunto de relaciones del esquema relacional de la base de datos para identificar posibles problemas y refinarlo. A diferencia de los pasos de análisis de requisitos y de diseño conceptual, que son esencialmente subjetivos, el refinamiento del esquema se puede guiar por una teoría elegante y potente. Esa teoría de *normalización* de relaciones —reestructurarlas para garantizar propiedades deseables— se estudia en el Capítulo 12.

5. **Diseño físico de bases de datos.** En este paso se toman en consideración las cargas de trabajo típicas esperadas que deberá soportar la base de datos y se refinará aún más el diseño de la base de datos para garantizar que cumple los criterios de rendimiento deseados. Puede que este paso no implique más que la creación de índices para algunas tablas y el agrupamiento de otras, o puede que suponga un rediseño sustancial de partes del esquema de la base de datos obtenido de los pasos de diseño anteriores. El diseño físico y el ajuste de las bases de datos se estudia en el Capítulo 13.

6. **Diseño de aplicaciones y de la seguridad.** Cualquier proyecto de software que implique a una base de datos debe tomar en consideración aspectos de su aplicación que van más allá de la propia base de datos. Las metodologías de diseño como UML (Apartado 2.7) intentan abordar todo el ciclo de diseño y desarrollo del software. En resumen, hay que identificar las entidades (por ejemplo, los usuarios, los grupos de usuarios, los departamentos) y los procesos relacionados con la aplicación. Hay que describir el papel de cada entidad en cada proceso que se refleje en una tarea de la aplicación, como parte del flujo de trabajo completo de esa tarea. Para cada papel hay que identificar las partes de la base de datos que debe tener accesibles y las que *no* debe tener accesibles, y adoptar las medidas necesarias para que esas reglas de acceso se cumplan. Los SGBD ofrecen varios mecanismos para ayudar en este paso, y se tratan en el Capítulo 14.

En la fase de implementación hay que codificar cada tarea en un lenguaje de programación (por ejemplo, Java) y emplear el SGBD para tener acceso a los datos. El desarrollo de aplicaciones se estudia en los Capítulos 6 y 7.

En general, la división del proceso de diseños en varios pasos debería verse como una clasificación de los *tipos* de pasos relacionados con el diseño. De manera realista, aunque se podría comenzar con el proceso en seis etapas que se ha esbozado aquí, el diseño completo de una base de datos probablemente necesite una fase posterior de **ajuste** en la que los seis tipos de etapas del diseño se entrelacen y se repitan hasta que el diseño sea satisfactorio.

2.2 ENTIDADES, ATRIBUTOS Y CONJUNTOS DE ENTIDADES

Una **entidad** es un objeto del mundo real que puede distinguirse de otros objetos. Ejemplos de entidades son: el juguete Dragón verde, el departamento de juguetes, el responsable del departamento de juguetes o la dirección postal del responsable del departamento de juguetes. Suele resultar útil identificar conjuntos de entidades similares. Un conjunto así se denomina **conjunto de entidades**. Obsérvese que no hace falta que los conjuntos de entidades sean disjuntos; tanto el conjunto de empleados del departamento de juguetes como el conjunto de empleados del departamento de electrodomésticos pueden contener al empleado Juanjo Díaz (que resulta que trabaja en los dos departamentos). También se podría definir un conjunto de entidades denominado Empleados que contuviera tanto al conjunto de empleados del departamento de juguetes como al de los empleados del departamento de electrodomésticos.

Cada entidad se describe empleando un conjunto de **atributos**. Todas las entidades de un conjunto de entidades dado tienen los mismos atributos; eso es lo que quiere decir *similar*. (Esta afirmación es una simplificación excesiva, como se verá cuando se discutan las jerarquías

de herencia en el Apartado 2.4.4, pero resulta suficiente por ahora y hace destacar la idea principal.) La selección de los atributos refleja el nivel de detalle con el que se desea representar la información relativa a las entidades. Por ejemplo, el conjunto de entidades Empleados podría utilizar el nombre, el número del documento nacional de identidad (DNI) y el número de la plaza de aparcamiento de cada empleado. Sin embargo, no se guardará, por ejemplo, la dirección del empleado (ni su sexo ni su edad).

Para cada atributo asociado con un conjunto de entidades hay que identificar el **dominio** de valores posibles. Por ejemplo, el dominio asociado con el atributo *nombre* de Empleados podría ser el conjunto de cadenas de caracteres de longitud veinte[1]. Como ejemplo adicional, si la empresa califica a sus empleados según una escala del uno al diez y guarda las calificaciones en un campo denominado *calificaciones*, el dominio asociado consistirá en los enteros del uno al diez. Además, para cada conjunto de entidades, se escoge una *clave*. Una **clave** es un conjunto mínimo de atributos cuyos valores identifican de manera unívoca a cada entidad del conjunto. Puede haber más de una clave **candidata**; en ese caso, se escogerá una de ellas como clave **principal**. Por ahora se supondrá que cada conjunto de entidades contiene, como mínimo, un conjunto de atributos que identifica de manera unívoca a cada entidad de ese conjunto; es decir, que el conjunto de entidades contiene una clave. Este punto se volverá a tratar en el Apartado 2.4.3.

El conjunto de entidades Empleados con los atributos *dni, nombre* y *plaza* se muestra en la Figura 2.1. Cada conjunto de entidades se representa por un rectángulo, y cada atributo por un óvalo. Los atributos de la clave principal están subrayados. Se podría indicar la información sobre el dominio junto con el nombre de cada atributo, pero se ha omitido para que las figuras sean más concisas. La clave es *dni*.

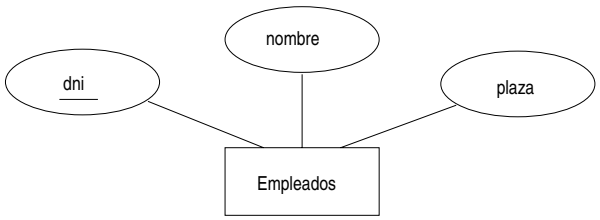

Figura 2.1 El conjunto de entidades Empleados

2.3 LAS RELACIONES Y LOS CONJUNTOS DE RELACIONES

Una **relación** es una asociación entre dos o más entidades. Por ejemplo, puede que se tenga una relación en que Avelino trabaja en el departamento farmaceútico. Al igual que con las entidades, puede que se desee reunir un conjunto de relaciones similares en un **conjunto de**

[1]Para evitar confusiones se da por supuesto que los nombres de los atributos no se repiten en los conjuntos de entidades. Esto no supone una verdadera limitación, ya que siempre se puede utilizar el nombre del conjunto de entidades para resolver las ambigüedades si el mismo nombre de atributo se utiliza en más de un conjunto de entidades.

entidades. Se puede considerar a los conjuntos de relaciones como conjuntos de n-tuplas:

$$\{(e_1, \ldots, e_n) \mid e_1 \in E_1, \ldots, e_n \in E_n\}$$

Cada n-tupla denota una relación que implica a n entidades, e_1 a e_n, donde la entidad e_i se halla en el conjunto de entidades E_i. En la Figura 2.2 puede verse el conjunto de relaciones Trabaja_en, en la que cada relación indica un departamento en el que trabaja ese empleado. Obsérvese que puede que varios conjuntos de relaciones impliquen a los mismos conjuntos de entidades. Por ejemplo, también se podría tener un conjunto de relaciones Dirige que implique a Empleados y Departamentos.

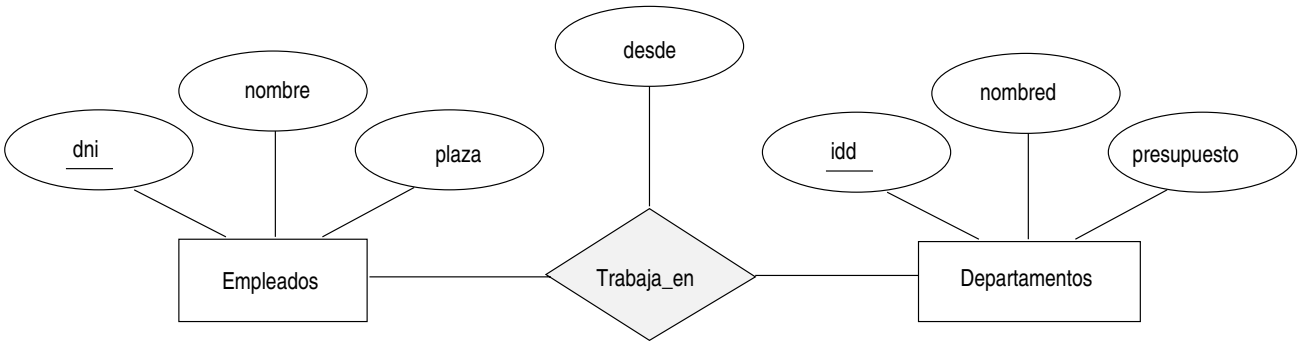

Figura 2.2 El conjunto de relaciones Trabaja_en

Las relaciones también pueden tener **atributos descriptivos**. Los atributos descriptivos se emplean para registrar la información sobre la relación, más que sobre las entidades participantes; por ejemplo, puede que se desee registrar que Avelino trabaja en el departamento farmaceútico desde enero de 1991. Esta información se captura en la Figura 2.2 añadiendo un atributo, *desde*, a Trabaja_en. Cada relación debe identificarse de manera unívoca por sus entidades participantes, sin necesidad de referencia alguna a los atributos descriptivos. En el conjunto de relaciones Trabaja_en, por ejemplo, cada relación Trabaja_en debe quedar identificada de manera unívoca por la combinación de empleado *dni* y departamento *idd*. Por tanto, para una pareja empleado-departamento dada, no se puede tener más de un valor *desde* asociado.

Cada **ejemplar** de un conjunto de relaciones es un conjunto de relaciones. De manera intuitiva se puede considerar cada ejemplar como una "instantánea" del conjunto de relaciones en un momento temporal dado. Puede verse un ejemplar del conjunto de relaciones Trabaja_en en la Figura 2.3. Cada entidad Empleados se denota por su *dni*, y cada entidad Departamentos se denota por su *idd*, en aras de la simplicidad. El valor *desde* se muestra junto a cada relación. (Los comentarios "varias a varias" y "participación total" de la figura se explican más adelante, cuando se estudian las restricciones de integridad.)

Como ejemplo adicional de diagrama ER, supóngase que cada departamento tiene oficinas en varias ubicaciones y que se desea registrar las ubicaciones en las que trabaja cada empleado. Esta relación es **ternaria**, ya que hay que registrar la asociación entre cada empleado, el departamento y la ubicación. El diagrama ER de esta variante de Trabaja_en, que se denominará Trabaja_en2, puede verse en la Figura 2.4.

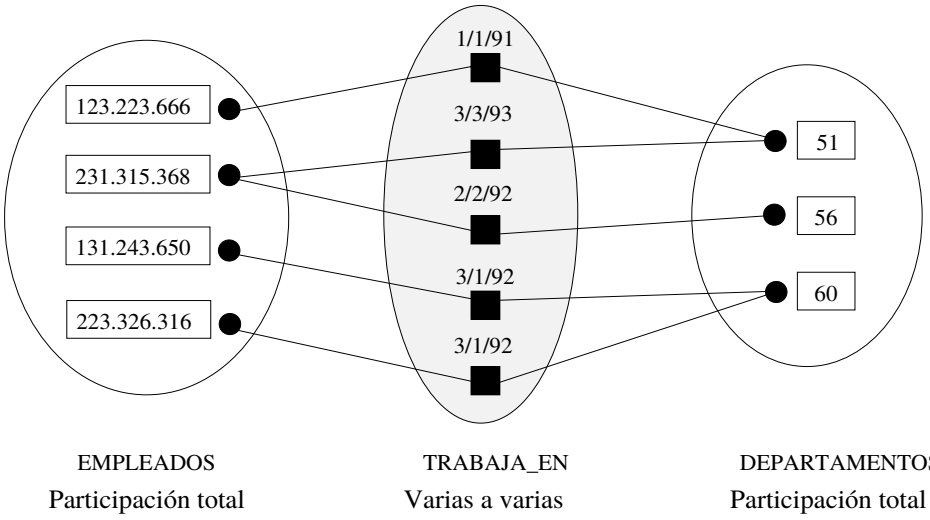

Figura 2.3 Un ejemplar del conjunto de relaciones Trabaja_en

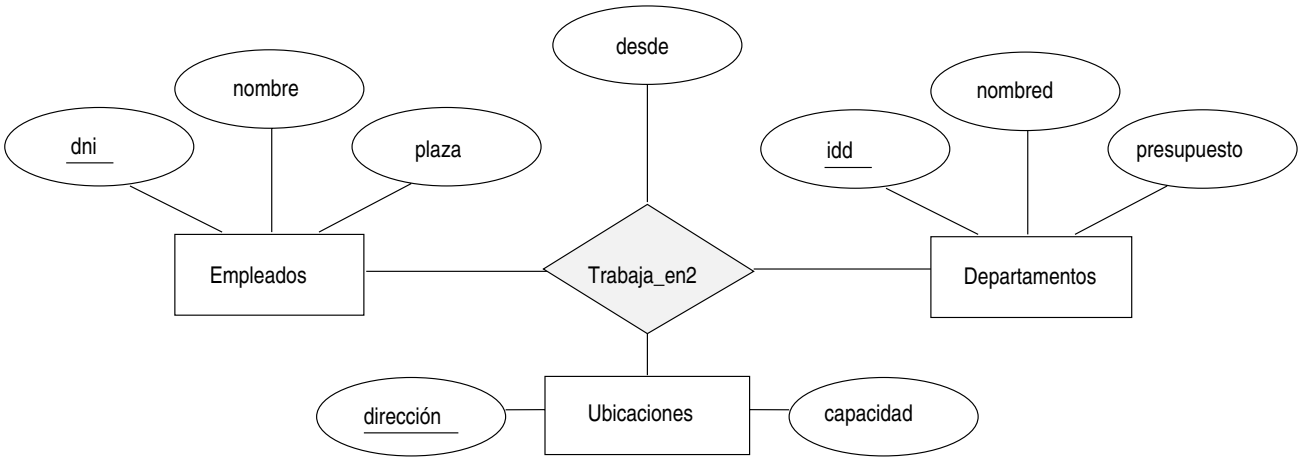

Figura 2.4 Un conjunto de relaciones ternarias

No hace falta que los conjuntos de entidades que participan en un conjunto de relaciones sean distintos; puede que a veces una relación implique a dos entidades del mismo conjunto de entidades. Por ejemplo, considérese el conjunto de relaciones Informa_a que puede verse en la Figura 2.5. Como los empleados rinden cuentas a otros empleados, las relaciones de Informa_a son de la forma (emp_1, emp_2), donde tanto emp_1 como emp_2 son entidades de Empleados. Sin embargo, interpretan **papeles** diferentes: emp_1 rinde cuentas al empleado encargado emp_2, lo que se refleja en los **indicadores de papeles** *supervisor* y *subordinado* en la Figura 2.5. Si un conjunto de entidades desempeña más de un papel, el indicador de papeles concatenado con un nombre de atributo del conjunto de entidades da un nombre único para cada atributo del conjunto de relaciones. Por ejemplo, el conjunto de relaciones Informa_a tiene los atributos correspondientes al *dni* del supervisor y al *dni* del subordinado, y el nombre de esos atributos es *dni_supervisor* y *dni_subordinado*.

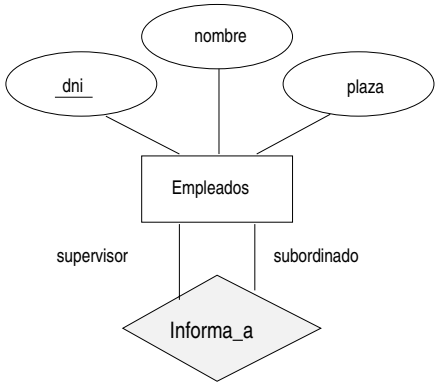

Figura 2.5 El conjunto de relaciones Informa_a

2.4 OTRAS CARACTERÍSTICAS DEL MODELO ER

A continuación se examinarán algunas de las estructuras del modelo ER que permiten describir algunas propiedades sutiles de los datos. La expresividad del modelo ER es una razón importante de su amplia utilización.

2.4.1 Restricciones de clave en relaciones

Considérese la relación Trabaja_en de la Figura 2.2. Cada empleado puede trabajar en varios departamentos, y cada departamento puede tener varios empleados, como puede verse en la Figura 2.3. El empleado 231.315.368 ha trabajado en el Departamento 51 desde 3/3/93 y en el Departamento 56 desde 2/2/92. El Departamento 51 tiene dos empleados.

Considérese ahora otro conjunto de relaciones denominado Dirige entre los conjuntos de entidades Empleados y Departamentos tal que cada departamento tenga como máximo un encargado, aunque se permite que cada empleado dirija más de un departamento. La restricción de que cada departamento tenga como máximo un encargado es un ejemplo de **restricción de clave**, e implica que cada entidad de departamentos aparezca como máximo en una relación

Dirige en cualquier ejemplar admisible de Dirige. Esta restricción se indica en el diagrama ER de la Figura 2.6 mediante una flecha que va de Departamentos a Dirige. De manera intuitiva, la flecha indica que, dada una entidad Departamentos, se puede determinar de manera unívoca la relación Dirige en la que aparece.

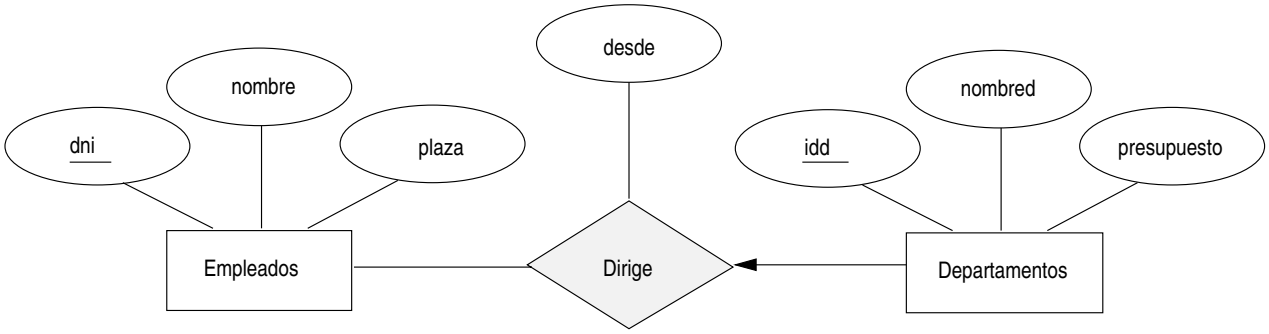

Figura 2.6 Restricción de clave en Dirige

En la Figura 2.7 se muestra un ejemplar del conjunto de relaciones Dirige. Aunque se trate también de un posible ejemplar del conjunto de relaciones Trabaja_en, el ejemplar de Trabaja_en que puede verse en la Figura 2.3 viola la restricción de clave en Dirige.

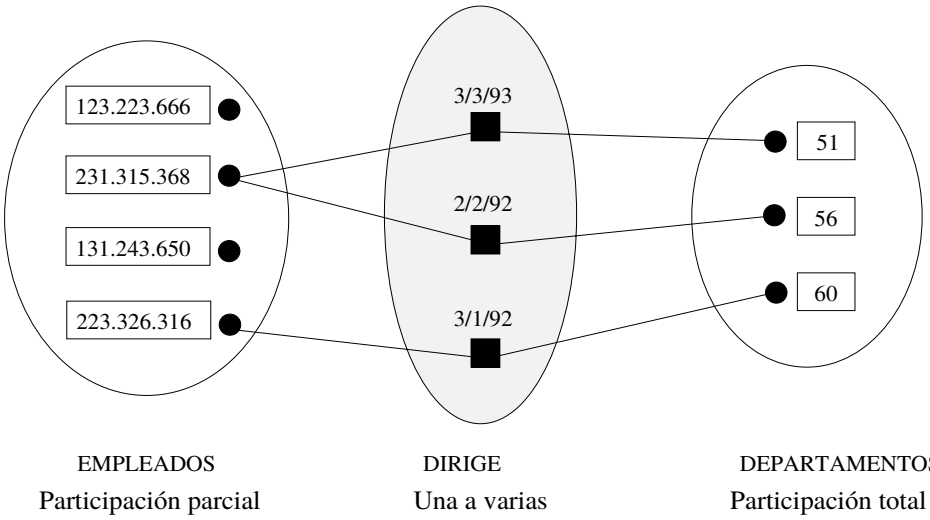

Figura 2.7 Ejemplar del conjunto de relaciones Dirige

Se dice a veces que los conjuntos de relaciones como Dirige son de **una a varias**, para indicar que *cada* empleado se puede asociar con *varios* departamentos (en funciones de encargado), mientras que cada departamento se puede asociar, como máximo, con un empleado como encargado. Por el contrario, se dice que el conjunto de relaciones Trabaja_en, en el que se permite que cada empleado trabaje en varios departamentos y que cada departamento tenga varios empleados, es de **varias a varias**.

Si se añade al conjunto de relaciones Dirige la restricción de que cada empleado pueda dirigir, como máximo, un departamento, lo que se indicaría añadiendo una flecha de Empleados a Dirige en la Figura 2.6, se tendrá un conjunto de relaciones de **una a una**.

Restricciones de clave en relaciones ternarias

Se puede ampliar este convenio —y el concepto subyacente de restricción de clave— a los conjuntos de relaciones que abarcan tres o más conjuntos de entidades: si el conjunto de entidades E tiene una restricción de clave en el conjunto de relaciones R, cada entidad de un ejemplar concreto de E aparecerá, como máximo, en una relación de (el ejemplar correspondiente de) R. Para indicar una restricción de clave sobre el conjunto de entidades E del conjunto de relaciones R, se traza una flecha de E a R.

En la Figura 2.8 se muestra una relación ternaria con una restricción de clave. Cada empleado trabaja, como máximo, en un departamento y en una única ubicación. Puede verse un ejemplar del conjunto de relaciones Trabaja_en3 en la Figura 2.9. Téngase en cuenta que cada departamento puede asociarse con varios empleados y ubicaciones y que cada ubicación puede asociarse con varios departamentos y empleados; sin embargo, cada empleado está asociado con un solo departamento y una única ubicación.

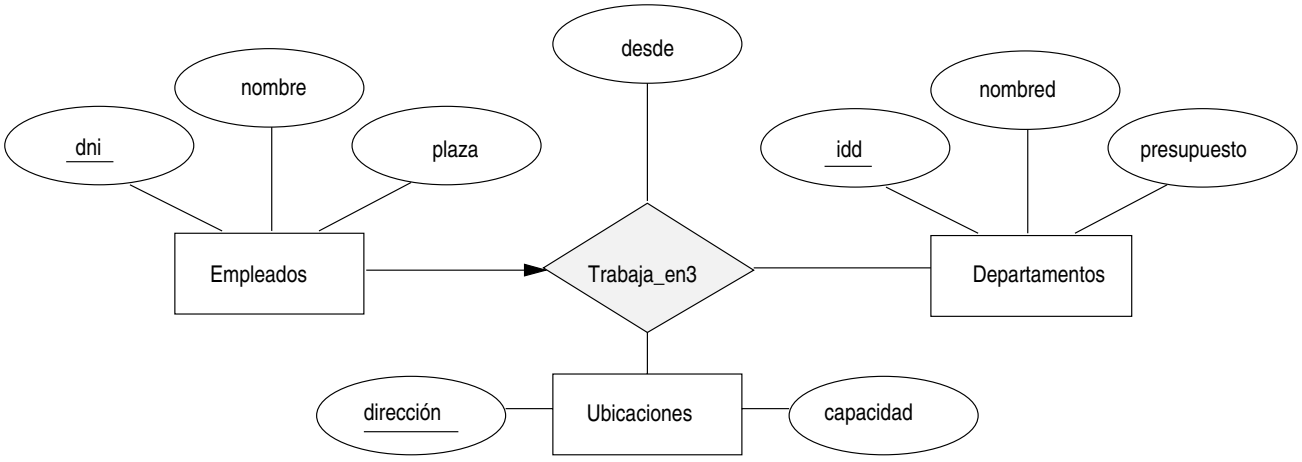

Figura 2.8 Un conjunto de relaciones ternarias con restricción de clave

2.4.2 Restricciones de participación

La restricción de clave sobre Dirige indica que cada departamento tiene, como máximo, un encargado. Una pregunta que resulta lógico formularse es si todos los departamentos tienen encargado. Supongamos que se exige que cada departamento tenga un encargado. Este requisito es un ejemplo de **restricción de participación**; se dice que la participación del conjunto de entidades Departamentos en el conjunto de relaciones Dirige es **total**. Una participación que no es total se dice de que es **parcial**. A modo de ejemplo, la participación del conjunto de entidades Empleados en Dirige es parcial, ya que no todos los empleados consiguen dirigir un departamento.

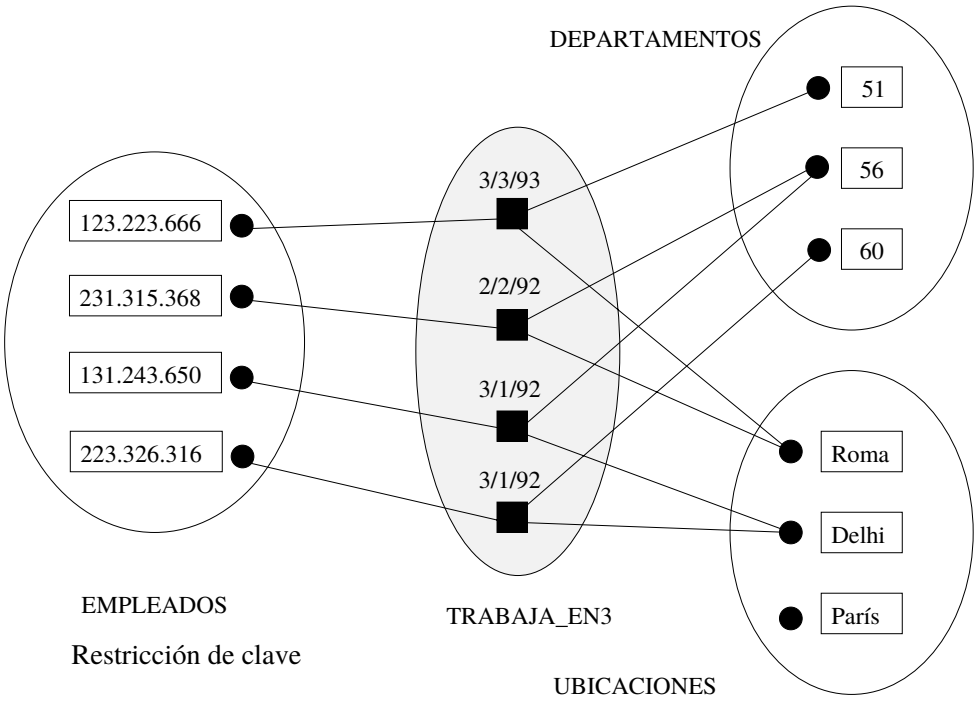

Figura 2.9 Un ejemplar de Trabaja_en3

De vuelta al conjunto de relaciones Trabaja_en, resulta natural esperar que cada empleado trabaje, como mínimo, en un departamento y que cada departamento tenga, como mínimo, un empleado. Esto significa que tanto la participación de Empleados como la de Departamentos en Trabaja_en es total. El diagrama ER de la Figura 2.10 muestra tanto al conjunto de relaciones Dirige como al conjunto de relaciones Trabaja_en y a todas las restricciones dadas. Si la participación de un conjunto de entidades en un conjunto de relaciones es total, ambos se conectan mediante una línea gruesa; de manera independiente, la presencia de una flecha indica una restricción de clave. Los ejemplares de Trabaja_en y Dirige mostrados en las Figuras 2.3 y 2.7 satisfacen todas las restricciones de la Figura 2.10.

2.4.3 Entidades débiles

Hasta ahora se ha supuesto que entre los atributos asociados a un conjunto de entidades se incluye una clave. Esta suposición no siempre se cumple. Por ejemplo, supóngase que los empleados pueden suscribir pólizas de seguros que cubran a las personas que dependen de ellos. Se desea registrar información sobre esas pólizas, incluyendo a las personas cubiertas en cada póliza; pero esa información es, en realidad, lo único que interesa de las personas que dependen de cada empleado. Si un empleado deja de serlo, las pólizas que hubiera suscrito se cancelan, y se desea eliminar de la base de datos toda la información relevante sobre esas pólizas y sobre las personas que dependen de ese antiguo empleado.

Se podría decidir identificar en este caso a cada persona que depende de un empleado únicamente por su nombre, ya que es razonable esperar que todas las personas que dependen de un empleado tengan nombres diferentes. Por tanto, los atributos del conjunto de entidades

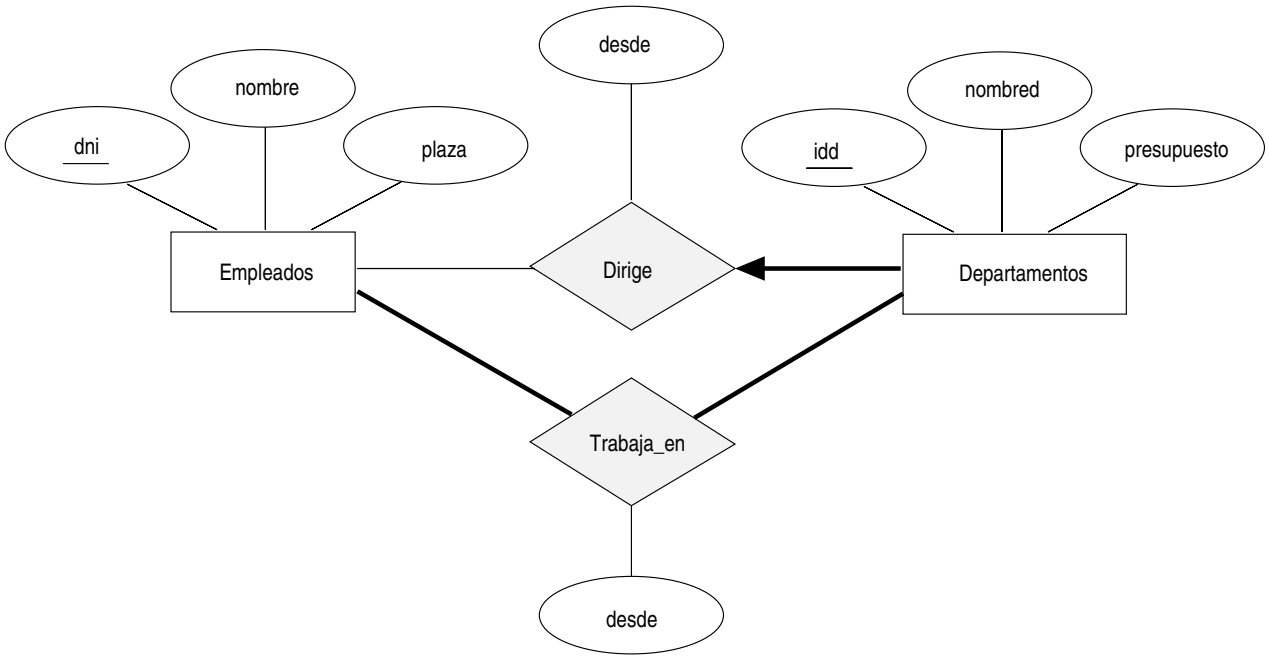

Figura 2.10 Dirige y Trabaja_en

Beneficiarios podrían ser *nombrep* y *edad*. El atributo *nombrep no* identifica de manera unívoca a cada persona que depende de un empleado. Recuérdese que la clave para Empleados es *dni*; por tanto, se podrían tener dos empleados llamados Sánchez con un hijo llamado José.

Beneficiarios es un ejemplo de **conjunto de entidades débiles**. Cada entidad débil sólo se puede identificar de manera unívoca tomando en consideración alguno de sus atributos junto con la clave principal de otra entidad, que se conoce como **propietaria identificadora**.

Se deben cumplir las restricciones siguientes:

- El conjunto de entidades propietario y el conjunto de entidades débiles deben participar en un conjunto de relaciones de una a varias (cada entidad propietaria se asocia con una o varias entidades débiles, pero cada entidad débil sólo tiene una propietaria). Este conjunto de relaciones se denomina **conjunto de relaciones identificadoras** del conjunto de entidades débiles.
- El conjunto de entidades débiles debe tener participación total en el conjunto de relaciones identificadoras.

Por ejemplo, cada entidad de Beneficiarios sólo se puede identificar de manera unívoca si se toma la clave de la entidad Empleados *propietaria* y *nombrep* de la entidad Beneficiarios. El conjunto de atributos de un conjunto de entidades débiles que identifica de manera unívoca a una entidad débil para una entidad propietaria dada se denomina *clave parcial* del conjunto de entidades débiles. En nuestro ejemplo, *nombrep* es una clave parcial de Beneficiarios.

El conjunto de entidades débiles Beneficiarios y su relación con Empleados se muestra en la Figura 2.11. La participación total de Beneficiarios en Póliza se indica enlazándolos mediante una línea gruesa. La flecha que va de Beneficiarios a Póliza indica que cada entidad

de Beneficiarios aparece, como máximo, en una relación de Póliza (en realidad, exactamente en una, debido a la restricción de participación). Para subrayar el hecho de que Beneficiarios es una entidad débil y Póliza es su relación identificadora se dibujan las dos con líneas oscuras. Para indicar que *nombrep* es una clave parcial de Beneficiarios, se subraya con una línea discontinua. Esto significa que puede haber perfectamente dos personas que dependan de empleados y tengan el mismo valor de *nombrep*.

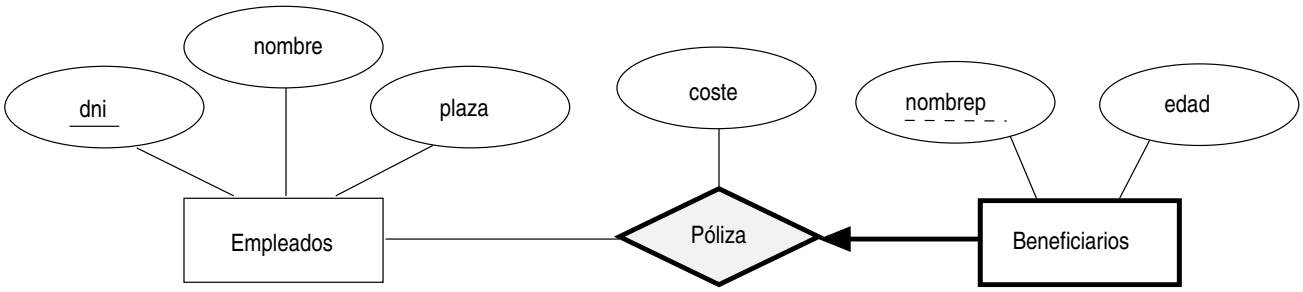

Figura 2.11 Un conjunto de entidades débiles

2.4.4 Jerarquías de clases

A veces resulta natural clasificar las entidades en un conjunto de entidades en subclases. Por ejemplo, puede que se desee hablar del conjunto de entidades Empleados_temp y del conjunto de entidades Empleados_fijos para distinguir el modo en que se calcula su sueldo. Puede que se hayan definido los atributos *horas_trabajadas* y *sueldo_hora* definidos para Empleados_temp y el atributo *idcontrato* para Empleados_fijos.

Se desea que todas las entidades de cada uno de esos conjuntos sean también entidades de Empleados y, como tales, deberán tener definidos todos los atributos de empleados. Por tanto, los atributos definidos para una entidad Empleados_temp dada son los atributos de Empleados más los de Empleados_temp. Se dice que los atributos del conjunto de entidades Empleados se **heredan** por el conjunto de entidades Empleados_temp y que una entidad de Empleados_temp **ES** una entidad de Empleados. Además —y a diferencia de las jerarquías de clases de los lenguajes de programación como C++— hay una restricción para las consultas sobre los ejemplares de estos conjuntos de entidades: las consultas que pidan todas las entidades Empleados también deben tomar en consideración las entidades Empleados_temp y Empleados_fijos. La Figura 2.12 ilustra la jerarquía de clases.

El conjunto de entidades Empleados también se puede clasificar según un criterio diferente. Por ejemplo, se puede identificar un subconjunto de empleados como Empleados_veteranos. Se puede modificar la Figura 2.12 para que refleje esta modificación añadiendo un segundo nodo ES como hijo de Empleados y haciendo a Empleados_veteranos hijo de ese nodo. Se puede seguir clasificando cada uno de esos conjuntos de entidades y crear una jerarquía ES multinivel.

Las jerarquías de clases se pueden considerar desde dos puntos de vista:

- Empleados está **especializada** en subclases. La especialización es el proceso de identificación de subconjuntos de un conjunto de entidades dado (la **superclase**) que comparten

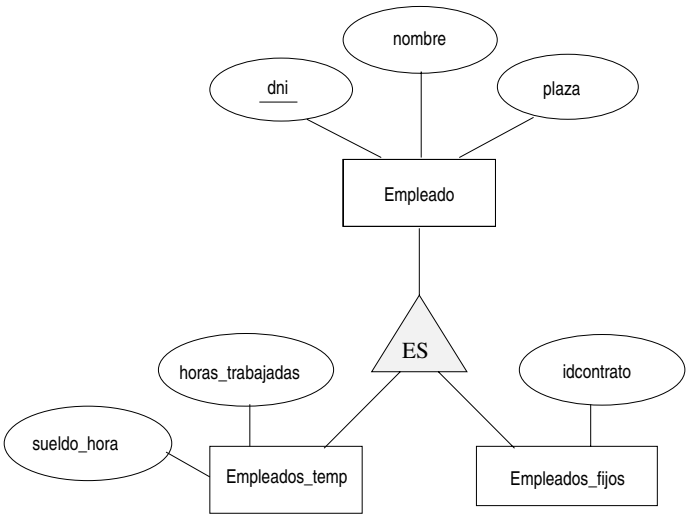

Figura 2.12 Jerarquía de clases

alguna característica distintiva. Generalmente se define en primer lugar la superclase, a continuación se definen las subclases y luego se añaden los atributos y los conjuntos de relaciones específicos de cada subclase.

- Empleados_temp y Empleados_fijos se **generalizan** mediante Empleados. Como ejemplo adicional, se pueden generalizar los conjuntos de entidades Motoras y Coches en el conjunto de entidades Vehículos_motorizados. La generalización consiste en identificar alguna característica común de un conjunto de conjuntos de entidades y crear un nuevo conjunto de entidades que contenga entidades que posean esas características comunes. Generalmente se definen en primer lugar las subclases, a continuación se define la superclase y luego se definen los conjuntos de relaciones que implican a la superclase.

Se pueden especificar dos tipos de restricciones con respecto a las jerarquías ES: las restricciones de *solapamiento* y de *cobertura*. Las **restricciones de solapamiento** determinan si se permite que dos clases contengan la misma entidad. Por ejemplo, ¿puede Avelino ser una entidad Empleados_temp y una entidad Empleados_fijos a la vez? De manera intuitiva, no. ¿Puede ser a la vez una entidad Empleados_fijos y una entidad Empleados_veteranos? De manera intuitiva, sí. Esto se denota escribiendo "Empleados_fijos `SOLAPA A` Empleados_Veteranos". En ausencia de una afirmación de este tipo, se da por supuesto de manera predeterminada que se restringe a los conjuntos de entidades a no solaparse.

Las **restricciones de cobertura** determinan si las entidades de las subclases incluyen de manera colectiva a todas las entidades de la superclase. Por ejemplo, ¿tienen que pertenecer todas las entidades Empleados a alguna de sus subclases? De manera intuitiva, no. ¿Tienen que ser todas las entidades Vehículos_motorizados una entidad Motoras o una entidad Coches? De manera intuitiva, sí; una propiedad característica de las jerarquías de generalización es que todos los ejemplares de una superclase son, a su vez, ejemplares de una subclase. Esto se denota escribiendo "Motoras `Y` Coches `CUBREN` Vehículos_motorizados". En ausencia de una afirmación de este tipo se supone de manera predeterminada que no hay ninguna restricción de cobertura; se pueden tener vehículos de motor que no sean ni motoras ni coches.

Hay dos motivos básicos para identificar subclases (por especialización o por generalización):

1. Puede que se desee añadir atributos descriptivos que sólo tengan sentido para las entidades de una subclase dada. Por ejemplo, *sueldo_hora* no tiene sentido para una entidad Empleados_fijos, cuya paga se determina mediante un contrato individual.

2. Puede que se desee identificar el conjunto de entidades que participan en una relación dada. Por ejemplo, puede que se desee definir la relación Dirige de modo que los conjuntos de entidades participantes sean Empleados_veteranos y Departamentos, para garantizar que sólo los empleados veteranos puedan ser encargados. Como ejemplo adicional, puede que Motoras y Coches tengan atributos descriptivos diferentes (por ejemplo, tonelaje y número de puertas) pero, como entidades de Vehículos_motorizados, deben estar matriculados. La información sobre la matrícula se puede capturar mediante una relación Matriculado_por entre Vehículos_motorizados y un conjunto de entidades denominado Propietarios.

2.4.5 Agregación

Como se ha definido hasta ahora, un conjunto de relaciones es una asociación entre conjuntos de entidades. A veces hay que modelar las relaciones entre un conjunto de entidades y de *relaciones*. Supóngase que se tiene un conjunto de entidades denominado Proyectos y que cada entidad de Proyectos está patrocinada por uno o varios departamentos. El conjunto de relaciones Patrocina captura esa información. El departamento que patrocina un proyecto puede asignar empleados para que controlen el patrocinio. De manera intuitiva, Controla debería ser un conjunto de relaciones que asocie relaciones Patrocina (en vez de entidades Proyectos o Departamentos) con entidades Empleados. Sin embargo, las relaciones que se han definido asocian dos o más *entidades*, no relaciones.

Para definir un conjunto de relaciones como Controla se introduce una nueva característica del modelo ER, denominada *agregación*. La **agregación** permite indicar que un conjunto de relaciones (identificado mediante un cuadro discontinuo) participa en otro conjunto de relaciones. Esto se ilustra en la Figura 2.13, con un cuadro discontinuo alrededor de Patrocina (y sus conjuntos de entidades participantes) para denotar la agregación. Esto permite tratar de manera efectiva Patrocina como un conjunto de entidades a los efectos de definir el conjunto de relaciones Controla.

¿Cuándo se debe emplear la agregación? De manera intuitiva se emplea cuando hace falta expresar una relación entre relaciones. ¿Pero no se pueden expresar relaciones que impliquen a otras relaciones sin emplear la agregación? En el ejemplo anterior, ¿por qué no hacer de Patrocina una relación ternaria? La respuesta es que realmente hay dos relaciones diferentes, Patrocina y Controla, cada una de las cuales tendrá sus propios atributos. Por ejemplo, la relación Controla tiene el atributo *hasta* que registra la fecha hasta la que un empleado ha sido nombrado controlador del patrocinio. Compárese este atributo con el atributo *desde* de Patrocina, que es la fecha en que el patrocinio entró en vigor. El empleo de la agregación en lugar de una relación ternaria también puede deberse a determinadas restricciones de integridad, como se explica en el Apartado 2.5.4.

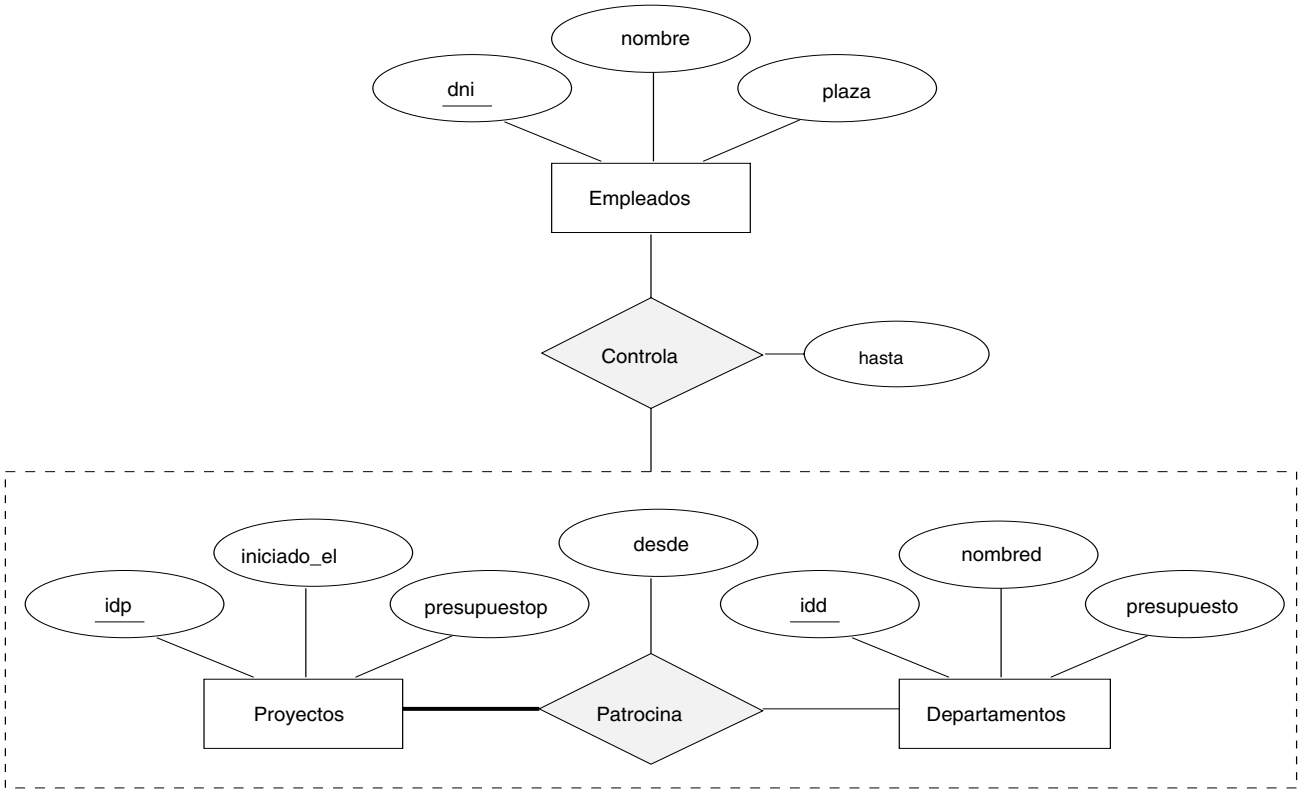

Figura 2.13 Agregación

2.5 DISEÑO CONCEPTUAL DEL MODELO ER

El desarrollo de diagramas ER supone escoger entre varias opciones, como las siguientes:

- ¿Un concepto dado se debe modelar como entidad o como atributo?
- ¿Un determinado concepto se debe modelar como entidad o como relación?
- ¿Cuáles son los conjuntos de relaciones y sus correspondientes conjuntos de entidades participantes? ¿Se deben emplear relaciones binarias o ternarias?
- ¿Se debe emplear la agregación?

Ahora se discutirán los problemas relacionados con la adopción de estas decisiones.

2.5.1 Entidades y atributos

Cuando se identifican los atributos de un conjunto de entidades no resulta a veces evidente si una determinada propiedad se debe modelar como atributo o como conjunto de entidades (y relacionarse con el primer conjunto de entidades mediante un conjunto de relaciones). Por ejemplo, considérese añadir información sobre el domicilio al conjunto de entidades Empleados. Una posibilidad es emplear el atributo *domicilio*. Esta opción resulta adecuada si sólo hace falta registrar un domicilio por empleado y basta con pensar en el domicilio como si fuera una cadena de caracteres. Una alternativa es crear un conjunto de entidades denominado Domicilios y registrar las asociaciones entre empleados y domicilios mediante una relación (por ejemplo, Tiene_domicilio). Esta alternativa más compleja resulta necesaria en dos situaciones:

- Hay que registrar más de una dirección por empleado.
- Se desea capturar la estructura de los domicilios en el diagrama ER. Por ejemplo, se puede descomponer el domicilio en ciudad, provincia, país y código postal, además de una cadena de caracteres para la información sobre la calle. Al representar el domicilio en forma de entidad con esos atributos, se pueden soportar consultas como "Buscar todos los empleados con domicilio en Madrid".

Como ejemplo adicional de la conveniencia de modelar un concepto como conjunto de entidades en vez de como atributo, considérese el conjunto de relaciones (denominado Trabaja_en4) que puede verse en la Figura 2.14.

Sólo se diferencia del conjunto de relaciones Trabaja_en de la Figura 2.2 en que tiene los atributos *desde* y *hasta*, en lugar de sólo *desde*. De manera intuitiva, registra el intervalo durante el que cada empleado trabaja para un departamento dado. Supóngase ahora que cada empleado puede trabajar en un departamento dado en más de un periodo.

Esta posibilidad queda descartada por la semántica del diagrama ER, ya que cada relación queda identificada de manera unívoca por sus entidades participantes (recuérdese del Apartado 2.3). El problema es que se desean registrar varios valores de los atributos descriptivos de cada ejemplar de la relación Trabaja_en4. (Esta situación es análoga a desear registrar varios domicilios para cada empleado.) Se puede abordar este problema mediante la introducción

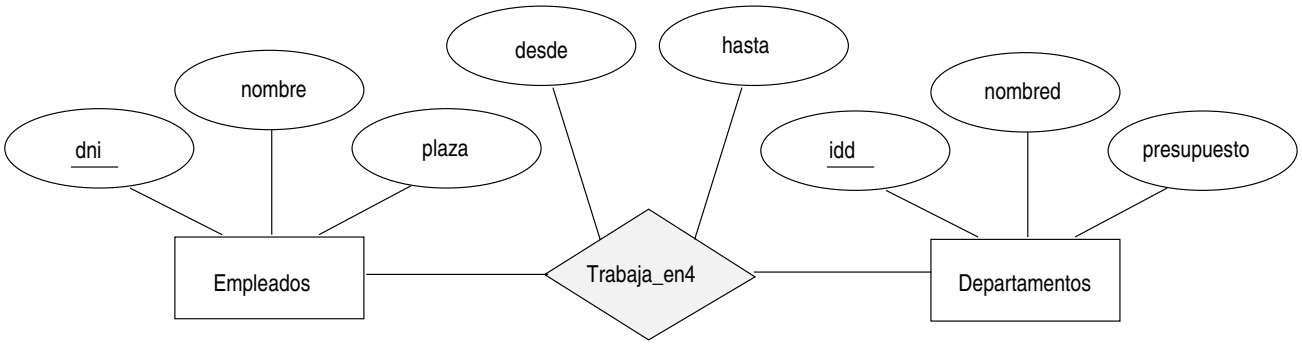

Figura 2.14 El conjunto de relaciones Trabaja_en4

de un conjunto de entidades denominado Permanencia, por ejemplo, con los atributos *desde* y *hasta*, como puede verse en la Figura 2.15.

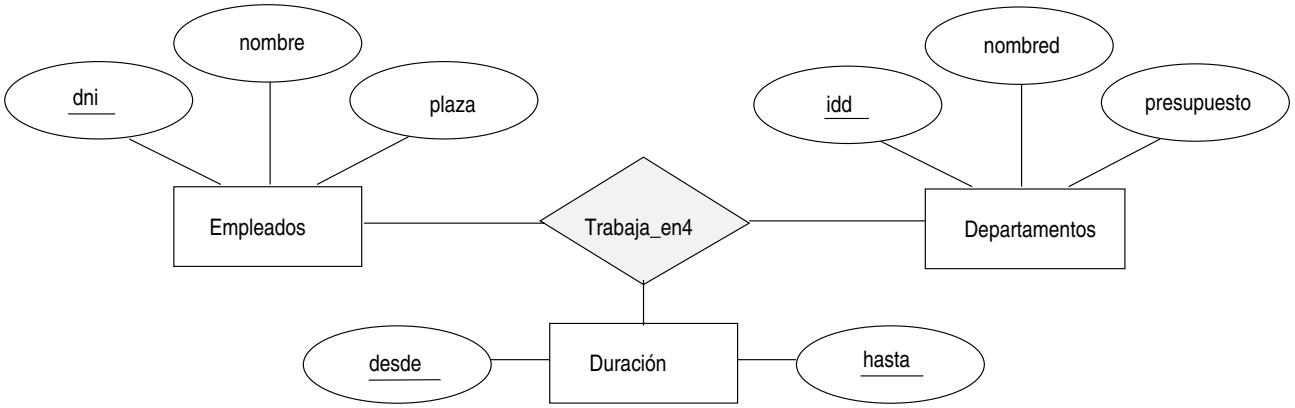

Figura 2.15 El conjunto de relaciones modificado Trabaja_en4

En algunas versiones del modelo ER se permite que los atributos adopten conjuntos como valores. Dada esta característica, se podría hacer de Permanencia un atributo de Trabaja_en4, en vez de un conjunto de entidades; asociado con cada relación Trabaja_en4 tendríamos un conjunto de intervalos. Este enfoque es, quizás, más intuitivo que el modelado de Permanencia como conjunto de entidades. Pese a todo, cuando esos atributos de tipo conjunto se trasladan al modelo relacional, que no los soporta, el esquema relacional resultante es muy parecido a lo que se obtiene considerando Permanencia como conjunto de entidades.

2.5.2 Entidades y relaciones

Considérese el conjunto de relaciones denominado Dirige de la Figura 2.6. Supóngase que se concede a cada encargado de departamento el presupuesto discrecional (*presupuestod*), como puede verse en la Figura 2.16, en la que también se ha renombrado el conjunto de relaciones como Dirige2.

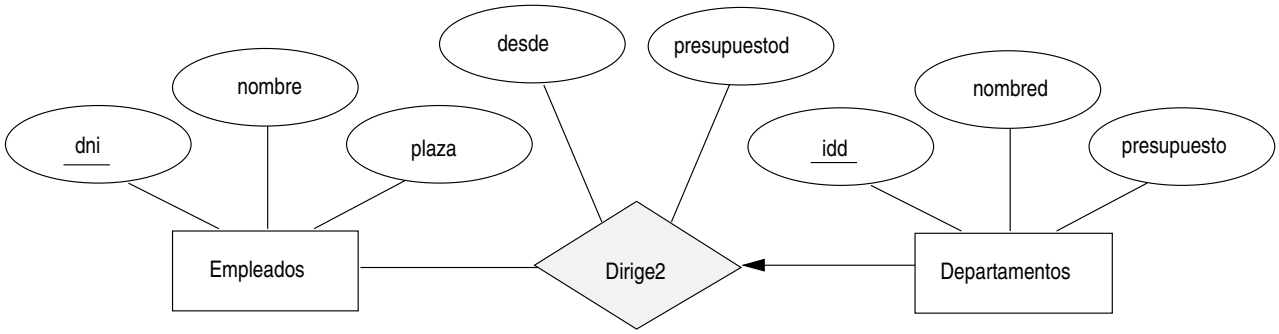

Figura 2.16 Entidades y relaciones

Dado un departamento, se conoce su encargado, su fecha de nombramiento y el presupuesto de ese departamento. Este enfoque resulta natural si se da por supuesto que cada encargado recibe un presupuesto discrecional diferente para cada departamento que dirige.

¿Pero qué ocurre si el presupuesto discrecional es una suma que abarca a *todos* los departamentos dirigidos por ese empleado? En ese caso, cada relación Dirige2 que implique a un empleado dado tendrá el mismo valor del campo *presupuestod*, lo que lleva al almacenamiento redundante de la misma información. Otro problema de este diseño es que induce a error; sugiere que el presupuesto está asociado a la relación cuando, en realidad, está asociado al encargado.

Estos problemas se pueden abordar mediante la introducción de un nuevo conjunto de entidades denominado Encargados (que puede ubicarse por debajo de Empleados en la jerarquía ES, para mostrar que todos los encargados son también empleados). Los atributos *desde* y *presupuestod* describen ahora a entidades encargado, como se pretendía. A modo de variación, aunque cada encargado tiene un presupuesto, cada uno de ellos puede tener una fecha de nombramiento (como encargado) diferente para cada departamento. En ese caso, *presupuestod* es un atributo de Encargados, pero *desde* es un atributo del conjunto de relaciones entre los encargados y los departamentos.

La naturaleza imprecisa del modelado ER puede, por tanto, dificultar el reconocimiento de las entidades subyacentes, y puede que se asocien atributos con relaciones en lugar de con las entidades correspondientes. En general, esos errores llevan al almacenamiento redundante de la misma información y pueden provocar muchos problemas. La redundancia y sus problemas asociados se discuten en el Capítulo 12, y se presenta una técnica denominada *normalización* para eliminar las redundancias de las tablas.

2.5.3 Relaciones binarias y ternarias

Considérese el diagrama ER de la Figura 2.17. Modela una situación en la que cada empleado puede tener varias pólizas, cada póliza puede ser propiedad de varios empleados y cada beneficiario puede estar cubierto por varias pólizas.

Supóngase que se tienen los requisitos adicionales siguientes:

- Dos o más empleados no pueden poseer conjuntamente una póliza.

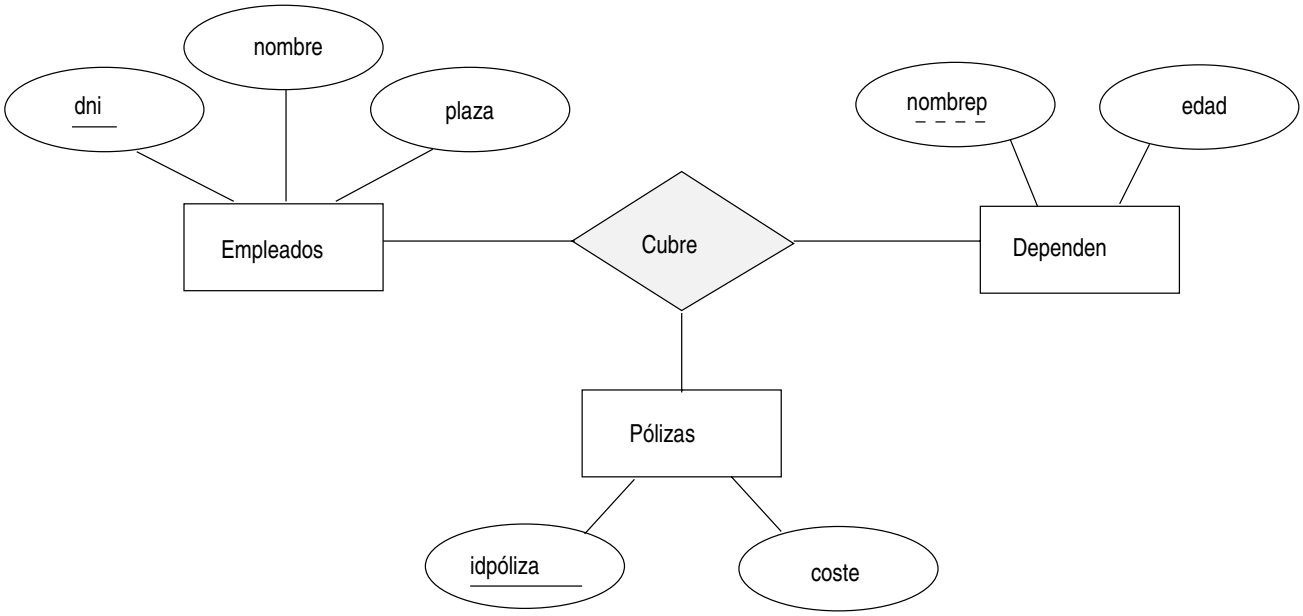

Figura 2.17 Las pólizas como conjunto de entidades

- Cada póliza debe ser propiedad de algún empleado.
- Dependientes es un conjunto de entidades débiles, y cada entidad dependiente queda identificada de manera unívoca empleando *nombrep* junto con *idpóliza* de una entidad póliza (que, de manera intuitiva, cubre a ese beneficiario).

El primer requisito sugiere que se ha impuesto una restricción de clave a Pólizas con respecto a Cubre, pero esa restricción tiene el efecto secundario no deseado de que cada póliza sólo pueda cubrir a un beneficiario. El segundo requisito sugiere que se ha impuesto una restricción de participación total sobre Pólizas. Esta solución resulta aceptable si cada póliza cubre, por lo menos, a un beneficiario. El tercer requisito nos obliga a introducir una relación identificadora que sea binaria (en esta versión de los diagramas ER, aunque hay versiones en las que no es ése el caso).

Aun ignorando el tercer requisito, la mejor manera de modelar esta situación es emplear dos relaciones binarias, como puede verse en la Figura 2.18.

Este ejemplo tiene realmente dos relaciones que implican a Pólizas, e intentar emplear una sola relación ternaria (Figura 2.17) resulta inadecuado. Hay situaciones, no obstante, en las que una relación asocia de manera inherente a más de dos entidades. Se ha visto un ejemplo de ello en las Figuras 2.4 y 2.15.

Como ejemplo típico de relación ternaria, considérense los conjuntos de entidades Repuestos, Proveedores y Departamentos, y el conjunto de relaciones Contratos (con el atributo descriptivo *cant*) que los implica a todos ellos. Un contrato dado especifica que un determinado proveedor suministrará (una determinada cantidad de) un repuesto concreto a un cierto departamento. Esta relación no puede capturarse de manera adecuada mediante un conjunto de relaciones binarias (sin el empleo de la agregación). Con las relaciones binarias se puede denotar que un proveedor "puede suministrar" determinados repuestos, que un departamento

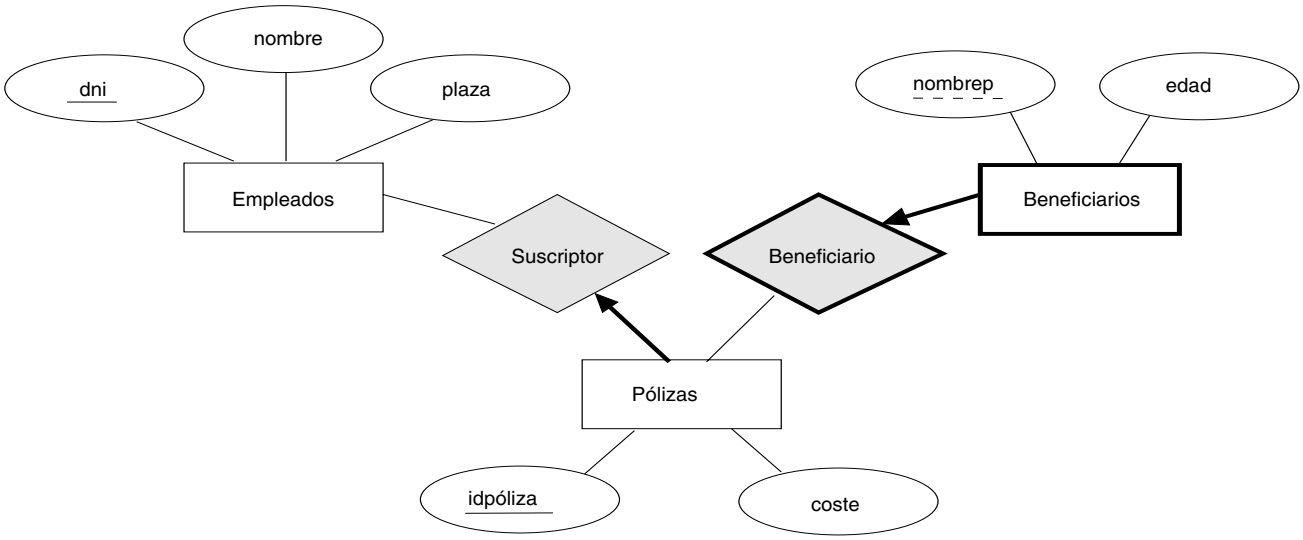

Figura 2.18 Las pólizas de nuevo

"necesita" ciertos repuestos o que un departamento "trata con" un proveedor dado. Ninguna combinación de estas relaciones expresa de manera adecuada el significado de un contrato, al menos por dos razones:

- El hecho de que el proveedor P pueda suministrar el repuesto R, de que el departamento D necesite el repuesto R y de que D compre a P no implica necesariamente que el departamento D compre realmente el repuesto R al proveedor P.
- No se puede representar adecuadamente el atributo *cant* de los contratos.

2.5.4 Agregación y relaciones ternarias

Como se señaló en el Apartado 2.4.5, la decisión de emplear la agregación o una relación ternaria viene determinada principalmente por la existencia de una relación que vincule un *conjunto de relaciones* con un conjunto de entidades (o un segundo conjunto de relaciones). Puede que la decisión también se guíe por determinadas restricciones de integridad que se deseen expresar. Por ejemplo, considérese el diagrama ER mostrado en la Figura 2.13. De acuerdo con ese diagrama, cada proyecto puede ser patrocinado por varios departamentos, cada departamento puede patrocinar uno o varios proyectos y cada patrocinio está controlado por uno o varios empleados. Si no hace falta registrar el atributo *hasta* de Controla, puede resultar razonable emplear una relación ternaria como, por ejemplo, Patrocina2, como puede verse en la Figura 2.19.

Considérese la restricción de que cada patrocinio (de un proyecto por un departamento) esté controlado, como máximo, por un empleado. No se puede expresar esa restricción en términos del conjunto de relaciones Patrocina2. Por otro lado, esa restricción se puede expresar fácilmente trazando una flecha desde la relación agregada Patrocina a la relación Controla de la Figura 2.13. Por tanto, la presencia de una restricción así es un motivo más para el empleo de la agregación en lugar de un conjunto de relaciones ternarias.

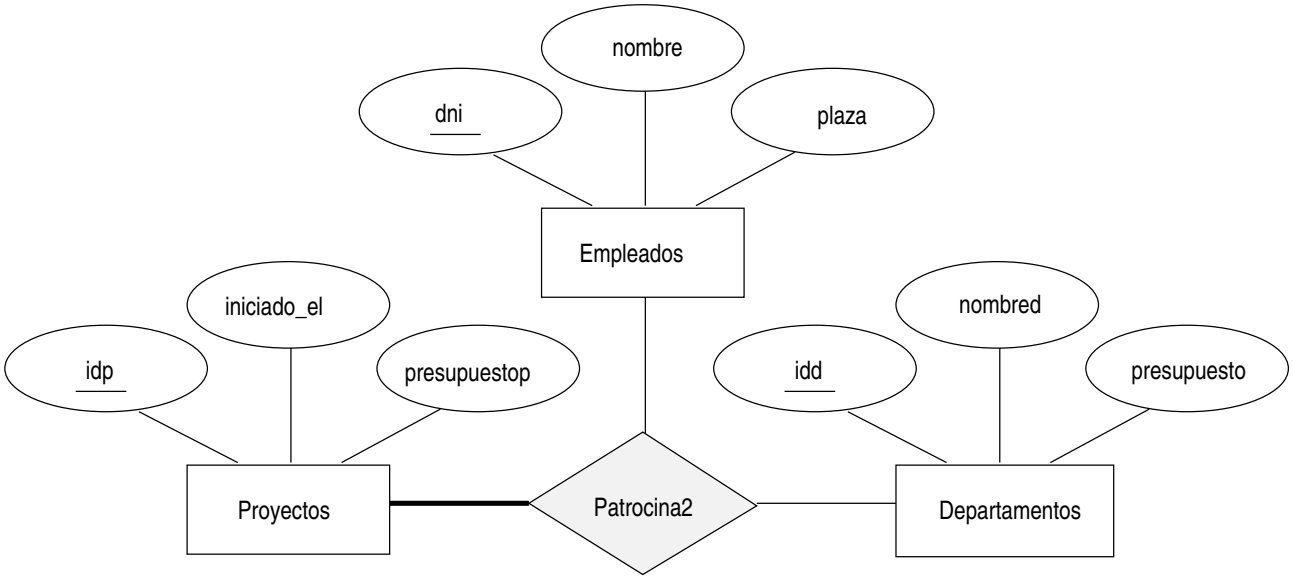

Figura 2.19 Empleo de una relación ternaria en lugar de la agregación

2.6 DISEÑO CONCEPTUAL PARA GRANDES EMPRESAS

Hasta ahora nos hemos concentrado en las estructuras disponibles en el modelo ER para la descripción de diversos conceptos y relaciones de las aplicaciones. El proceso de diseño conceptual consiste en algo más que la mera descripción de pequeños fragmentos de la aplicación en términos del diagrama ER. Para las grandes empresas puede que el diseño exija el esfuerzo de más de un diseñador y abarque datos y código de aplicaciones empleado por varios grupos de usuarios. El empleo de un modelo de datos semántico de alto nivel, como los diagramas ER, para el diseño conceptual en un entorno de este tipo ofrece la ventaja adicional de que el diseño de alto nivel se puede representar mediante diagramas y lo puede comprender fácilmente la gran cantidad de personas que deben ofrecer aportaciones al proceso de diseño.

Un aspecto importante del proceso de diseño es la metodología empleada para estructurar el desarrollo del diseño global y garantizar que tenga en cuenta todos los requisitos de usuario y sea consistente. El enfoque habitual es que se tomen en consideración los requisitos de diversos grupos de usuarios, se resuelvan de algún modo los que entren en conflicto y se genere un conjunto único de requisitos globales al final de la fase de análisis de requisitos. La generación de un conjunto único de requisitos globales es una labor difícil, pero permite que la fase de diseño conceptual se continúe con el desarrollo de un esquema lógico que abarque todos los datos y a todas las aplicaciones de la empresa.

Un enfoque alternativo es el desarrollo de esquemas conceptuales independientes para los diferentes grupos de usuarios y su posterior *integración*. Para integrar varios esquemas conceptuales hay que establecer correspondencias entre las entidades, las relaciones y los atributos, y hay que resolver numerosos tipos de conflictos (por ejemplo, de denominaciones, incoherencias de dominios, diferencias en las unidades de medida, etcétera). Esta tarea es

intrínsecamente difícil. En algunas situaciones la integración de los esquemas no se puede evitar; por ejemplo, cuando una organización se fusiona con otra, puede que haya que integrar las bases de datos existentes. La integración de esquemas también aumenta de importancia a medida que los usuarios van exigiendo el acceso a orígenes de datos *heterogéneos*, a menudo mantenidos por organizaciones diferentes.

2.7 EL LENGUAJE UNIFICADO DE MODELADO

Hay muchos enfoques del diseño integral de software, que abarcan todas las etapas desde la identificación de los requisitos empresariales a las especificaciones finales para una aplicación completa, incluidos el flujo de trabajo, las interfaces de usuario y muchos aspectos de los sistemas de software que superan ampliamente las bases de datos y los datos que se guardan en ellas. En este apartado se tratará brevemente un enfoque que está ganando popularidad, denominado **enfoque del lenguaje unificado de modelado** (unified modeling language, **UML**).

UML, al igual que el modelo ER, presenta la atractiva característica de que sus estructuras se pueden dibujar en forma de diagrama. Abarca un espectro del proceso de diseño de software más amplio que el modelo ER:

- **Modelado de la empresa.** En esta fase el objetivo es describir los procesos empresariales implicados en la aplicación de software que se está desarrollando.
- **Modelado del sistema.** La comprensión de los procesos empresariales se emplea para identificar los requisitos de la aplicación de software. Una parte de esos requisitos son los de la base de datos.
- **Modelado conceptual de la base de datos.** Esta etapa corresponde a la creación del diseño ER para la base de datos. Con ese objetivo, UML ofrece muchas estructuras análogas a las del modelo ER.
- **Modelado físico de la base de datos.** UML ofrece también representaciones gráficas de las opciones de diseño físico de la base de datos, como la creación de espacios de tablas y de índices. (El diseño físico de la base de datos se discute en capítulos posteriores, pero no así las estructuras de UML correspondientes.)
- **Modelado del sistema de hardware.** Los diagramas UML se pueden emplear para describir la configuración de hardware empleada para la aplicación.

Hay muchas clases de diagramas UML. Los diagramas de **casos de uso** describen las acciones llevadas a cabo por el sistema en respuesta a las solicitudes de los usuarios, y las personas implicadas en esas acciones. Estos diagramas especifican la funcionalidad externa que se espera que soporte el sistema.

Los diagramas de **actividad** muestran el flujo de acciones en los procesos comerciales. Los diagramas de **estado** describen las interacciones dinámicas entre los diferentes objetos del sistema. Estos diagramas, empleados en el modelado de la empresa y en el del sistema, describen el modo en que se va a implementar la funcionalidad externa, de manera consistente con las reglas del negocio y con los procesos de la empresa.

Los diagramas de **clases** son parecidos a los diagramas ER, aunque son más generales en el sentido de que se pretende que modelen entidades de *aplicación* (de manera intuitiva, componentes importantes del programa) y sus relaciones lógicas, además de las entidades de datos y sus relaciones.

Tanto los conjuntos de entidades como los de relaciones pueden representarse en UML como clases, junto con las restricciones de las claves, las entidades débiles y las jerarquías de clase. El término *relación* se emplea en UML de manera ligeramente diferente, y las relaciones de UML son binarias. Esto conduce a veces a confusión sobre si los conjuntos de relaciones de los diagramas ER que implican a tres o más conjuntos de entidades se pueden representar directamente en UML. La confusión desaparece una vez que se comprende que todos los conjuntos de relaciones (en el sentido de ER) se representan en UML como clases; las "relaciones" binarias de UML son, en esencia, precisamente los enlaces mostrados en los diagramas ER entre los conjuntos de entidades y los de relaciones.

Los conjuntos de relaciones con restricciones de las claves se suelen omitir de los diagramas UML, y la relación se indica enlazando directamente los conjuntos de entidades implicados. Por ejemplo, considérese la Figura 2.6. La representación UML de ese diagrama ER tendría una clase para Empleados y otra para Departamentos, y la relación Dirige se mostraría mediante un enlace entre ambas clases. El enlace se puede etiquetar con un nombre y la correspondiente información de cardinalidad para mostrar que cada departamento sólo puede tener un encargado.

Como se verá en el Capítulo 3, los diagramas ER se traducen en el modelo relacional mediante la transformación de cada conjunto de entidades o de relaciones en una tabla. Además, como se verá en el Apartado 3.5.3, las tablas correspondientes a los conjuntos de relaciones de una a varias suelen omitirse mediante la inclusión de alguna información adicional sobre cada una de esas relaciones en la tabla correspondiente a alguno de los conjuntos de entidades implicados en las mismas. Por tanto, los diagramas de clases en UML se corresponden estrechamente con las tablas creadas mediante la transformación de los diagramas ER.

En realidad, cada clase de los diagramas UML se transforma en una tabla del diagrama UML de la base de datos correspondiente. Los **diagramas de bases de datos** en UML muestran el modo en que se representan las clases en la base de datos y contienen detalles adicionales sobre la estructura de la base de datos, como las restricciones de integridad y los índices. Los enlaces ("relaciones" de UML) entre las clases de UML conducen a diversas restricciones de integridad entre las tablas correspondientes. Muchos detalles específicos del modelo relacional (por ejemplo, las *vistas*, las *claves externas*, los *campos que pueden tener valores nulos*) y que reflejan opciones del diseño físico (por ejemplo, los campos indexados) se pueden modelar en los diagramas de bases de datos en UML.

Los diagramas de **componentes** en UML describen los aspectos de almacenamiento de las bases de datos, como los *espacios de tablas* y las *particiones de las bases de datos*, así como las interfaces con las aplicaciones que tienen acceso a la base de datos. Finalmente, los diagramas de **implantación** muestran los aspectos hardware del sistema.

El objetivo de este libro es concentrarse en los datos almacenados en las bases de datos y en los aspectos de diseño relacionados. Para ello se adoptará una visión simplificada de las otras etapas del diseño y desarrollo del software. Más allá de la discusión concreta de UML, se pretende que el material de este apartado sitúe los aspectos de diseño que se tratan en el contexto más amplio del diseño de software. Los autores esperan que esto ayude a los lectores

interesados en una discusión más general del diseño de software a complementar este estudio mediante referencias a otros materiales de su enfoque preferido al diseño global de sistemas.

2.8 ESTUDIO DE UN CASO: LA TIENDA EN INTERNET

Como ilustración se introducirá ahora el estudio de un caso de diseño desde su comienzo hasta su fin que se empleará como ejemplo permanente a lo largo del libro. TiposBD S.A., una conocida empresa de consultoría de bases de datos, ha sido requerida para ayudar a Benito y Norberto (B&N) con el diseño y la implementación de su base de datos. B&N es una gran librería especializada en los libros sobre carreras de caballos, y ha decidido abrirse a Internet. TiposBD comprueba primero que B&N desee y pueda pagar sus elevados honorarios y luego convoca una comida de trabajo —que, naturalmente, se facturará a B&N— para realizar el análisis de requisitos.

2.8.1 Análisis de requisitos

El propietario de B&N, a diferencia de muchas de las personas que necesitan una base de datos, ha meditado ampliamente lo que desea y ofrece un resumen conciso:

"Me gustaría que los clientes pudieran examinar el catálogo de libros y realizar pedidos por Internet. Actualmente acepto pedidos por teléfono. Tengo, sobre todo, clientes corporativos que me llaman y me dan el código ISBN del libro y la cantidad que desean comprar; a menudo pagan con tarjeta de crédito. Luego preparo el envío, que contiene los libros pedidos. Si no dispongo de suficientes copias en el almacén, encargo copias adicionales y retraso el envío hasta que llegan; prefiero enviar todo el pedido de cada cliente de una sola vez. Mi catálogo incluye todos los libros que vendo. Para cada libro, el catálogo incluye su código ISBN, título, autor, precio de adquisición, precio de venta y año de publicación. La mayor parte de mis clientes son habituales, y dispongo de un registro con su nombre y dirección. Los clientes nuevos tienen que llamarme primero y abrir una cuenta antes de poder utilizar mi Web.

En la nueva Web los clientes se deberían identificar antes de nada por su número de identificación de cliente, que debe ser único. Luego deberían poder examinar el catálogo y formular pedidos en línea."

Los consultores de TiposBD están un poco sorprendidos por la rapidez con que se ha completado la fase de requisitos —suelen hacer falta semanas de discusiones (y muchas comidas y muchas cenas) llevarla a buen puerto— pero vuelven a su oficina para analizar esta información.

2.8.2 Diseño conceptual

En la etapa de diseño conceptual, TiposBD desarrolla una descripción de alto nivel de los datos en términos del modelo ER. El diseño inicial se puede ver en la Figura 2.20. Los libros y los clientes se modelan como entidades y se relacionan mediante los pedidos que formulan los clientes. Pide es un conjunto de relaciones que conecta los conjuntos de entidades Libros y Clientes. Para cada pedido se almacenan los atributos siguientes: cantidad, fecha del pedido

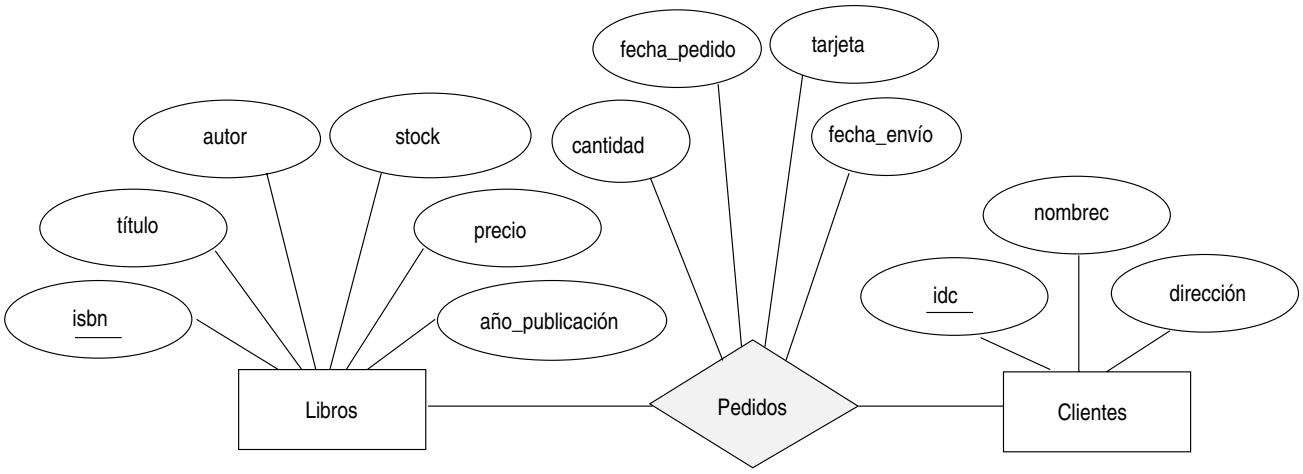

Figura 2.20 Diagrama ER del diseño inicial

y fecha de envío. En cuanto se envía un pedido, se establece la fecha de envío; hasta entonces, la fecha de envío se define como *nula*, lo que indica que ese pedido no se ha enviado todavía.

TiposBD tiene en este momento una reunión interna para la revisión del diseño, y se suscitan varias dudas. Para proteger sus identidades, nos referiremos al jefe del equipo de diseño como Tipo 1 y al revisor del diseño como Tipo 2.

Tipo 2: ¿Qué ocurre si un cliente realiza dos pedidos del mismo libro el mismo día?

Tipo 1: Se trata el primer pedido mediante la creación de una relación Pide y el segundo mediante la actualización del valor del atributo cantidad de esa relación.

Tipo 2: ¿Qué ocurre si un cliente realiza dos pedidos para libros diferentes el mismo día?

Tipo 1: No hay problema. Cada ejemplar del conjunto de relaciones Pide relaciona al cliente con un libro diferente.

Tipo 2: Ah, ¿pero qué ocurre si un cliente realiza dos pedidos del mismo libro en días diferentes?

Tipo 1: Se puede emplear el atributo fecha de pedido de la relación Pide para distinguir los dos pedidos.

Tipo 2: Oh, no, no se puede. Los atributos de Clientes y de Libros deben contener una clave de Pide conjuntamente. Por tanto, este diseño no permite que un cliente realice pedidos del mismo libro en días diferentes.

Tipo 1: Pues vaya, tienes razón. Oh, bueno, probablemente no le importe a B&N; ya veremos.

TiposBd decide pasar a la fase siguiente, el diseño lógico de la base de datos; volveremos a verlos en el Apartado 3.8.

2.9 PREGUNTAS DE REPASO

Las respuestas a las preguntas de repaso pueden encontrarse en los apartados indicados.

- Indíquense las principales etapas del diseño de bases de datos. ¿Cuál es el objetivo de cada etapa? ¿En qué etapa se emplea principalmente el modelo ER? **(Apartado 2.1)**
- Defínanse estos términos: *entidad*, *conjunto de entidades*, *atributo*, *clave*. **(Apartado 2.2)**

- Defínanse estos términos: *relación*, *conjunto de relaciones*, *atributos descriptivos*. **(Apartado 2.3)**
- Defínanse los siguientes tipos de restricciones y dese un ejemplo de cada una: *restricción de la clave*, *restricción de participación*. ¿Qué es una *entidad débil*? ¿Qué son las *jerarquías de clases*? ¿Qué es la *agregación*? Dese una situación de ejemplo que motive el empleo de cada una de estas estructuras de diseño del modelo ER. **(Apartado 2.4)**
- ¿Qué directrices se emplearían para cada una de estas opciones en la realización de un diseño ER? emplear un atributo o un conjunto de entidades, una entidad o un conjunto de relaciones, una relación binaria o una ternaria, o la agregación. **(Apartado 2.5)**
- ¿Por qué resulta especialmente difícil diseñar una base de datos para una gran empresa? **(Apartado 2.6)**
- ¿Qué es UML? ¿Cómo encaja el diseño de bases de datos en el diseño global de un sistema de software con empleo intensivo de los datos? ¿Cómo se relaciona UML con los diagramas ER? **(Apartado 2.7)**

EJERCICIOS

Ejercicio 2.1 Explíquense brevemente los términos siguientes: *atributo, dominio, entidad, relación, conjunto de entidades, conjunto de relaciones, relación de una a varias, relación de varias a varias, restricción de participación, restricción de solapamiento, restricción de cobertura, conjunto de entidades débiles, agregación* e *indicador de papel*.

Ejercicio 2.2 La base de datos de una universidad contiene información sobre los profesores (identificados por su documento nacional de identidad, o DNI) y las asignaturas (identificadas por idasignatura). Los profesores imparten las asignaturas; cada una de las situaciones siguientes concierne al conjunto de relaciones Imparte. Para cada situación, dibújese un diagrama que la describa (suponiendo que no se aplica ninguna otra restricción).

1. Los profesores pueden impartir la misma asignatura en varios semestres, y cada ocasión debe registrarse.
2. Los profesores pueden impartir el mismo curso en varios semestres, y sólo hace falta registrar la ocasión más reciente. (Supóngase que esta condición es aplicable a las preguntas siguientes.)
3. Cada profesor debe impartir alguna asignatura.
4. Cada profesor imparte exactamente una asignatura (ni más ni menos).
5. Cada profesor imparte exactamente una asignatura (ni más ni menos), y cada asignatura debe impartirla algún profesor.
6. Supóngase ahora que determinadas asignaturas puede impartirlas conjuntamente un equipo de profesores, pero que es posible que ningún profesor de ese equipo pueda impartir esa asignatura. Modélese esta situación, introduciendo más conjuntos de entidades y de relaciones si fuera necesario.

Ejercicio 2.3 Considérese la siguiente información sobre la base de datos de una universidad:

- Cada profesor tiene DNI, nombre, edad, rango y especialidad de investigación.
- Cada proyecto tiene número de proyecto, nombre de patrocinador (por ejemplo, el CSIC), fecha de comienzo, fecha de finalización y presupuesto.
- Cada alumno de posgrado tiene DNI, nombre, edad y programa de posgrado (por ejemplo, magíster o doctorado).

- Cada proyecto está dirigido por un profesor (conocido como investigador principal de ese proyecto).
- En cada proyecto trabajan uno o varios profesores (conocidos como investigadores de ese proyecto).
- Cada profesor puede dirigir varios proyectos o trabajar en ellos.
- En cada proyecto trabajan uno o varios alumnos de posgrado (conocidos como ayudantes de investigación de ese proyecto).
- Cuando los alumnos de posgrado trabajan en un proyecto, su trabajo debe supervisarlo un profesor. Cada alumno de posgrado puede trabajar en varios proyectos, en cuyo caso puede tener un supervisor diferente en cada uno de ellos.
- Cada departamento tiene número de departamento, nombre de departamento y despacho principal.
- Cada departamento tiene un profesor (conocido como director) que lo dirige.
- Cada profesor trabaja en uno o varios departamentos, y por cada departamento en que trabaja se asocia un porcentaje de tiempo a su trabajo.
- Cada alumno de posgrado tiene un departamento principal en el que trabaja en su titulación.
- Cada alumno de posgrado tiene otro alumno de posgrado más veterano (conocido como asesor del alumno) que lo aconseja sobre las asignaturas en las que debe matricularse.

Diséñese y dibújese un diagrama ER que capture la información sobre la universidad. Empléense únicamente el modelo ER básico, es decir, las entidades, las relaciones y los atributos. Asegúrese de indicar las claves y las restricciones de participación que pueda haber.

Ejercicio 2.4 La base de datos de una empresa necesita almacenar información sobre los empleados (identificados por su *dni*, con *sueldo* y *teléfono* como atributos); los departamentos (identificados por *númd*, con *nombred* y *presupuesto* como atributos); y los hijos de los empleados (con *nombre* y *edad* como atributos). Los empleados *trabajan* en los departamentos; cada departamento está *dirigido por* un empleado; cada hijo debe quedar identificado de manera unívoca por su *nombre* una vez conocidos su padre o su madre (que es uno de los empleados; supóngase que sólo uno de los padres trabaja en la empresa). No se está interesado en la información relativa al hijo una vez que el padre deja la empresa.

Dibújese un diagrama ER que capture esta información.

Ejercicio 2.5 Discos Sinpueblo (Notown Records) ha decidido guardar información sobre los músicos que intervienen en sus discos (así como otros datos de la empresa) en una base de datos. Sabiamente, la empresa ha decidido contratarlo a usted como diseñador de la base de datos (por su tarifa habitual de 2500 €diarios).

- Cada músico que graba en Sinpueblo tiene DNI, nombre, dirección y número de teléfono. Los músicos que cobran poco suelen compartir la misma dirección, y ninguna dirección tiene más de un teléfono.
- Cada instrumento que se emplea en las canciones grabadas en Sinpueblo tiene nombre (por ejemplo, guitarra, sintetizador o flauta) y una clave musical (por ejemplo, re, si bemol, mi bemol).
- Cada disco que se graba en el sello Sinpueblo tiene título, fecha de copyright, formato (por ejemplo, CD o MC) e identificador del disco.
- Cada canción grabada en Sinpueblo tiene título y autor.
- Cada músico puede tocar varios instrumentos, y cada instrumento pueden tocarlo varios músicos.
- Cada disco tiene varias canciones, pero ninguna canción puede aparecer en más de un disco.
- Cada canción la interpretan uno o varios músicos, y cada músico puede interpretar varias canciones.
- Cara disco tiene exactamente un músico que actúa como productor. Por supuesto, cada músico puede producir varios discos.

Diséñese un esquema conceptual para Sinpueblo y dibújese un diagrama ER para ese esquema. La información siguiente describe la situación que debe modelar la base de datos de Sinpueblo. Asegúrese de indicar todas las restricciones de clave y cardinalidad, y cualquier suposición que se haga. Identifíquense todas las restricciones que no se puedan capturar en el diagrama ER y explíquese brevemente el motivo de no poder expresarlas.

Ejercicio 2.6 Los miembros del Departamento de Informática que vuelan a menudo se han quejado al personal del Aeropuerto de Daganzo de la mala organización de ese aeropuerto. En consecuencia, el personal del aeropuerto ha decidido que toda la información relativa al aeropuerto se organice mediante un SGBD, y se le ha contratado para diseñar la correspondiente base de datos. Su primera tarea es organizar la información relativa a los aviones que se albergan y realizan su mantenimiento en el aeropuerto. La información relevante es la siguiente:

- Cada avión tiene un número de registro y es de un modelo concreto.
- El aeropuerto puede atender a varios modelos de aeroplano, y cada modelo está identificado por un número de modelo (por ejemplo, DC-10) y tiene una capacidad y un peso dados.
- En el aeropuerto trabajan varios técnicos. Hay que guardar el nombre, DNI, dirección, número de teléfono y sueldo de cada uno de ellos.
- Cada técnico es experto en uno o varios modelos de avión, y su maestría puede solaparse con la de otros técnicos. También hay que guardar esta información sobre los técnicos.
- Los controladores aéreos deben pasar un examen médico anual. Para cada controlador aéreo hay que guardar la fecha del examen más reciente.
- Todos los empleados del aeropuerto (incluidos los técnicos) pertenecen a un sindicato. Hay que guardar el número de afiliación al sindicato de cada empleado. Se puede suponer que cada empleado queda identificado de manera unívoca por el número de su documento nacional de identidad.
- El aeropuerto tiene varias pruebas que se emplean periódicamente para garantizar que los aviones siguen estando en condiciones de volar. Cada prueba tiene un número de examen de la Dirección General de Aviación Civil (DGAC), un nombre y una puntuación máxima posible.
- La DGAC exige que el aeropuerto registre el momento en que cada avión es examinado por un técnico dado siguiendo una prueba concreta. Para cada examen la información necesaria es la fecha, el número de horas que el técnico ha empleado en realizarlo y la puntuación que el avión ha conseguido.

1. Dibújese un diagrama ER para la base de datos del aeropuerto. Asegúrese de indicar los diferentes atributos de cada entidad y de cada conjunto de relaciones; especifíquense también las restricciones de clave y participación de cada conjunto de relaciones. Especifíquense las restricciones de solapamiento y de cobertura necesarias (en su lengua materna).
2. La DGAC ha aprobado una norma que obliga a que los exámenes de los aviones los realicen técnicos expertos en cada modelo. ¿Cómo se expresaría esa restricción en el diagrama ER? Si no se puede expresar, explíquese el motivo brevemente.

Ejercicio 2.7 La cadena de farmacias Recetas Rayos X le ha ofrecido abastecerlo gratuitamente de medicamentos de por vida si diseña su base de datos. Dado el creciente coste de la atención sanitaria, usted acepta. Ésta es la información que logra reunir:

- Los pacientes se identifican mediante su DNI y hay que registrar su nombre, dirección y edad.
- Los médicos se identifican mediante su DNI. Para cada médico hay que registrar el nombre, la especialidad y los años de ejercicio.
- Cada empresa farmacéutica se identifica por el nombre y tiene un número de teléfono.
- Para cada medicamento hay que registrar el nombre comercial y la fórmula. Cada medicamento lo vende una empresa farmacéutica dada, y el nombre comercial identifica ese medicamento de manera unívoca entre los productos de esa empresa. Si se borra una empresa farmacéutica, ya no hace falta realizar el seguimiento de sus productos.
- Cada farmacia tiene nombre, dirección y número de teléfono.
- Cada paciente tiene un médico de cabecera. Cada médico tiene, como mínimo, un paciente.
- Cada farmacia vende varios medicamentos y tiene un precio para cada uno de ellos. Cada medicamento se puede vender en varias farmacias, y el precio puede variar de una a otra.

- Los médicos recetan drogas a sus pacientes. Cada médico puede recetar una o varias drogas a varios pacientes, y cada paciente puede conseguir recetas de varios médicos. Cada receta tiene una fecha y una cantidad asociadas con ella. Se puede suponer que, si un médico receta el mismo medicamento al mismo paciente más de una vez, sólo hay que guardar la última receta.
- Las empresas farmacéuticas tienen contratos de larga duración con las farmacias. Cada empresa farmacéutica puede contratar con varias farmacias, y cada farmacia puede contratar con varias empresas farmacéuticas. Para cada contrato hay que guardar la fecha de inicio, la fecha de finalización y el texto íntegro.
- Las farmacias nombran un supervisor para cada contrato. Siempre debe haber un supervisor por contrato, pero el supervisor de un contrato puede cambiar durante su periodo de validez.

1. Dibújese un diagrama ER que capture esta información. Identifíquense las restricciones que no queden capturadas por el diagrama ER.
2. ¿Cómo cambiaría el diseño si cada medicamento debiera venderlo todas las farmacias al mismo precio?
3. ¿Cómo cambiaría el diseño si los requisitos de diseño cambiaran de la manera siguiente? Si un médico receta el mismo medicamento al mismo paciente más de una vez, puede que haya que guardar varias de esas recetas.

Ejercicio 2.8 Aunque siempre ha deseado ser artista, acabó siendo experto en bases de datos porque le encantan los animales y, de alguna manera, confundió "base de datos" con "base de gatos". Ese antiguo amor sigue ahí, sin embargo, por lo que ha montado una empresa de bases de datos, BaseArt, que crea productos para galerías de arte. El corazón del producto es una base de datos con un esquema que captura toda la información que necesitan guardar las galerías. Las galerías guardan información de los artistas: nombre (que es único), lugar de nacimiento, edad y estilo artístico. Para cada pieza de arte hay que guardar el autor, el año de realización, el título (que es único), la rama artística (por ejemplo, pintura, litografía, escultura o fotografía) y el precio. Las piezas de arte también se clasifican en grupos de diferentes tipos como, por ejemplo, retratos, bodegones, obras de Picasso u obras del siglo diecinueve; cada pieza puede pertenecer a más de un grupo. Cada grupo se identifica mediante un nombre (como los anteriores) que lo describe. Finalmente, las galerías guardan información sobre sus clientes. Para cada cliente las galerías guardan el nombre (que es único), la dirección, el total de euros que se ha gastado en esa galería (¡muy importante!) y los artistas y grupos de arte que más le gustan.

Dibújese el diagrama ER de esta base de datos.

Ejercicio 2.9 Respóndanse las siguientes preguntas.

- Explíquese brevemente los siguientes términos: *UML, diagramas de casos de uso, diagramas de estado, diagramas de clases, diagramas de bases de datos, diagramas de componentes* y *diagramas de implantación.*
- Explíquese la relación entre los diagramas ER y UML.

NOTAS BIBLIOGRÁFICAS

Hay varios libros que ofrecen un buen tratamiento del diseño conceptual; entre ellos están [43] (que también contiene un estudio de las herramientas comerciales para el diseño de bases de datos) y [442].

El modelo ER fue propuesto por Chen [107] y se han propuesto ampliaciones del mismo en varios trabajos posteriores. La generalización y la agregación se introdujeron en [417]. [254] y [354] contienen buenos estudios sobre los modelos semánticos de datos. Los aspectos dinámicos y temporales de los modelos semánticos de datos se tratan en [457].

[443] estudia una metodología de diseño basada en el desarrollo del diagrama ER y su posterior traducción al modelo relacional. Markowitz considera la integridad referencial en el contexto de la transformación del

modelo ER al relacional y estudia el soporte ofrecido en varios sistemas comerciales (hasta la fecha de su publicación) en [318, 319].

Las actas de la conferencia entidad-relación contienen numerosos trabajos sobre diseño conceptual, con cierto énfasis en el modelo ER como, por ejemplo, [421].

La integración de las vistas se discute en varios trabajos, como [57, 85, 116, 156, 334, 338, 337, 411, 420, 456]. [44] es un estudio de varios enfoques a la integración.