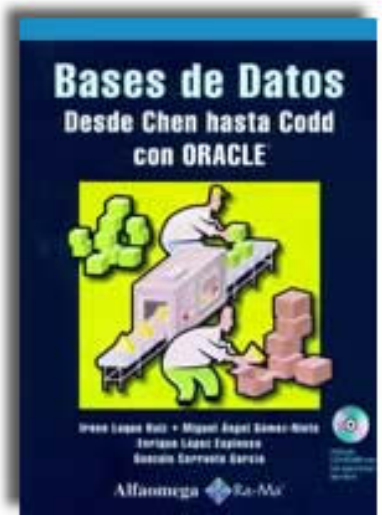
BASES DE DATOS



Desde Chen hasta Codd con ORACLE
LUQUE, Irene; GÓMEZ-NIETO, Miguel; LÓPEZ, Enrique y CERRUELA, Gonzalo
448 págs.
Rústica 17 x 23 cm
ISBN 970-15-0760-6
Coedición: Alfaomega-Rama

Una obra de consulta dirigida a quienes se acercan por primera vez a las bases de datos y a los sistemas de gestión de bases de datos y, también a los profesionales que encontrarán los fundamentos teóricos de esta tecnología (sin un formalismo matemático excesivo), así como los problemas y heurísticas a seguir en el proceso de traducción del mundo real a una representación entendible para estos sistemas.

Brinda una serie de problemas reales y resueltos paso a paso; expuestos así: enunciado del problema, esquema conceptual y lógico, su validación, proposición y resolución.

Contiene los ejercicios desarrollados en el libro y adicionales

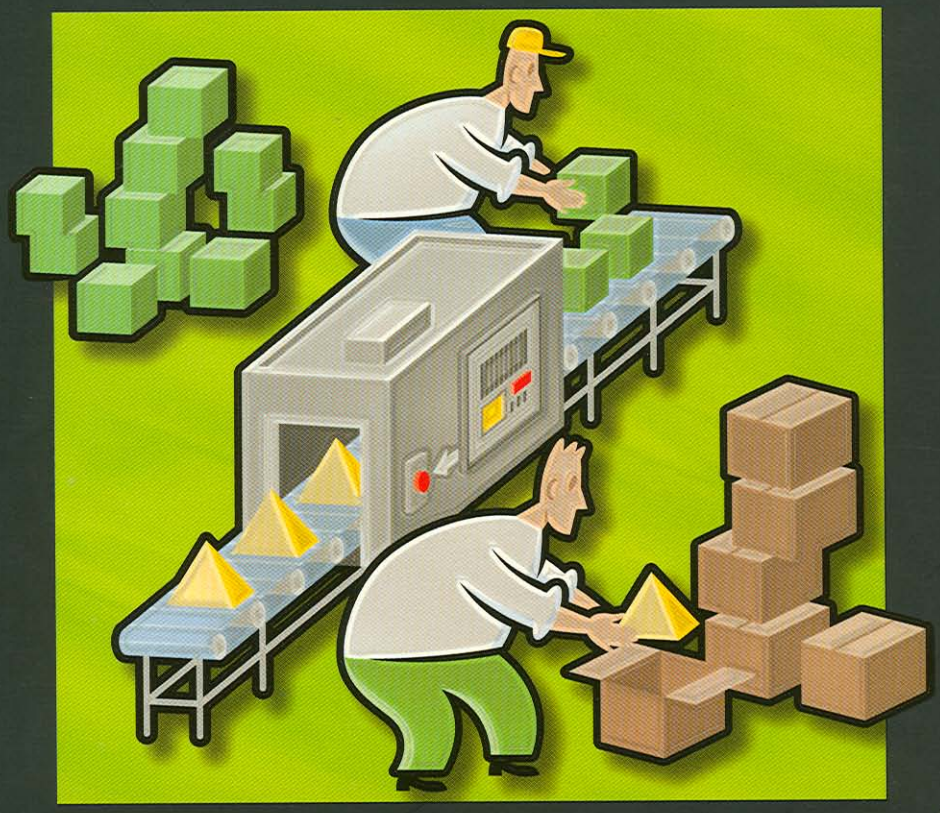
Resumen del contenido:

Introducción a las bases de datos - Representación de los problemas del mundo real - El modelo de datos relacional - El álgebra relacional. SQL - Traducción de esquemas E-R a esquemas relacionales - El catastro municipal - Los residuos tóxicos - Dietas ganaderas - Colecciones de mariposas - Venta y consumo de cigarrillos - Pesca deportiva - Proyecciones de películas - La fiesta nacional - Comidas a domicilio - Explotaciones mineras - ORACLE en Internet.

<< Regresar

Bases de Datos

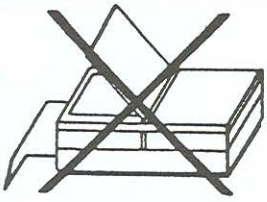
Desde Chen hasta Codd con ORACLE®



Irene Luque Ruiz • Miguel Ángel Gómez-Nieto
Enrique López Espinosa
Gonzalo Cerruela García

Incluye CD-ROM con los ejercicios del libro

Ra-Ma®

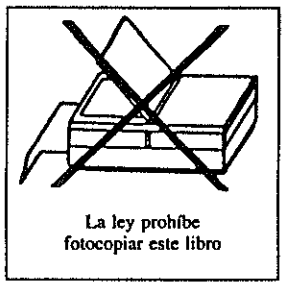


La ley prohíbe
fotocopiar este libro

Editado por:
RA-MA Editorial
Ctra. de Canillas, 144
28043 MADRID
Teléfono: 91 381 03 00
Fax: 91 381 03 72
Correo electrónico: editorial@ra-ma.com
Internet: **www.ra-ma.es** y **wwww.ra-ma.com**
ISBN: 84-7897-478-4
Depósito Legal: M-23913-2001
Autoedición: Autores
Filmación: Albadalejo, S.L.
Imprime: Franjograf, S.L.
Impreso en España
Primera impresión: Junio 2001

Bases de Datos

Desde Chen hasta Codd con ORACLE®



La ley prohíbe
fotocopiar este libro

Bases de Datos: Desde Chen hasta Codd con ORACLE®

Editado por:
RA-MA Editorial
Ctra. de Canillas, 144
28043 MADRID
Teléfono: 91 381 03 00
Fax: 91 381 03 72
Correo electrónico: editorial@ra-ma.com
Internet: www.ra-ma.es y www.ra-ma.com
ISBN: 84-7897-478-4
Depósito Legal: M-23913-2001
Autoedición: Autores
Filmación: Albadalejo, S.L.
Imprime: Franjograf, S.L.
Impreso en España
Primera impresión: Junio 2001

CONTENIDO

AGRADECIMIENTOS

La realización de esta obra ha sido posible gracias a la financiación concedida por la *Universidad de Córdoba* al proyecto número **NP-0006** presentado en la *II Convocatoria de Proyectos de Innovación y Mejora de la Calidad de la Enseñanza* para el curso 2000/2001.

Para el desarrollo de los ejercicios prácticos se ha hecho uso de los productos de *Oracle*, lo cual ha sido posible gracias al acuerdo entre *Oracle* y la *Universidad de Córdoba* a través de la *Iniciativa Académica de Oracle* (OAI).

Los autores desean expresar también su agradecimiento a las distintas promociones de alumnos de los estudios de *Informática* de la *Universidad de Córdoba*, a los cuales se les ha impartido, e imparte actualmente, el contenido de esta obra, lo que ha servido para que pudieran validar la misma. En particular, deseamos agradecer a *D. Javier Fuentes Alventosa* y *D. Rafael Carlos Vázquez Carmona*, su valiosa colaboración en la construcción del CD que acompaña la obra.

Nuestro más sincero agradecimiento a la editorial *Ra-Ma* por la confianza que ha depositado en los autores de esta obra, y en su contenido, avalada con la publicación de la misma.

PREFACIO

Bases de Datos es uno de los términos más populares dentro y fuera del *mundillo* de la informática, aunque si tuviéramos que realizar un *ranking* de popularidad, el de *Internet* podría ser el más *popular* hoy en día. A pesar de ello, la mayoría de las personas que de alguna forma interaccionan con un ordenador *conocen, o creen conocer* el significado de este término y, casi con toda seguridad, han utilizado, utilizan y utilizarán algún producto software en el que la *base de datos* y su gestión sea el *corazón del mismo*.

Efectivamente, tras más de treinta años desde la aparición de esta tecnología, las bases de datos se han popularizado hasta llegar a ser en nuestros días una herramienta más que utilizan desde los profesionales del área hasta los usuarios completamente legos en informática, debido a los productos software que se han puesto a disposición de estos usuarios, los cuales les permite una comunicación lógica y sencilla con él mismo, ocultándole la complejidad teórica, conceptual y física en la cual se basa el producto que están utilizando.

A pesar de todo, es también cierto que todavía se sigue desarrollando mucho software en cuya especificación comercial se habla continuamente del término *Base de Datos*, cuando este software no satisface la mayor parte de las propiedades exigidas a estos sistemas, pero claro está, *¿quién puede decir que el soporte de almacenamiento en el cual se basa su sistema no es una base de datos?;* aunque se trate en realidad de una colección de ficheros planos y, posiblemente, pobremente relacionados.

Las *Bases de Datos* son utilizadas como solución, o como soporte a la solución, de la mayor parte de los problemas que se presentan en el mundo real. Por ello, el

conocimiento de esta tecnología, de la teoría en la que se soporta y de los modelos de sistemas encargados de la gestión de las mismas es de suma importancia.

Si bien existen productos software, orientados al usuario, que permiten la definición de la base de datos y el posterior tratamiento de los datos almacenados en la misma, estos sistemas no pueden (al menos hasta la fecha) ayudar al usuario en el proceso de traducción de las características del problema del mundo que quieren tratar a una representación entendible por el sistema informático y por el software que están utilizando. Este proceso de análisis del problema y diseño de la base de datos debe ser realizado por el usuario auxiliándose únicamente de sus conocimientos acerca del problema, de su experiencia en el desarrollo de tareas similares y, cómo no, de sus conocimientos acerca de las bases de datos y de los sistemas de gestión de bases de datos.

Los actuales sistemas software ayudan al usuario en el desarrollo de aplicaciones dedicadas a la manipulación de los datos, pero prestan poca o ninguna ayuda al usuario en los pasos previos que son de vital importancia en la garantía de la calidad del sistema construido, lo que se traducirá en sistemas con un pobre desempeño y, posiblemente, con una *vida* muy corta. Por ello, sigue siendo importante para cualquier usuario de las bases de datos el conocimiento (naturalmente, a distinto nivel de profundidad dependiendo del tipo de usuario) de la teoría en la cual se soporta esta tecnología y de las heurísticas a seguir en el proceso de traducción de un problema del mundo real a una representación entendible por los sistemas de gestión de bases de datos que van a gestionar los datos correspondientes al mismo.

Actualmente, debido tanto al avance tecnológico, como a la difusión de Internet, las bases de datos se encuentran con un nuevo desafío para satisfacer las necesidades de negocio existentes. Este desafío consiste en que puedan ser integradas en sistemas cliente/servidor que son explotados a través de Internet por todo tipo de usuarios, y presentando a los mismos una interfaz integrada en una página (por ejemplo, html) que se ejecuta en un navegador estándar, y que presenta información muy variada: texto, imágenes, vídeo, sonido, etc.

Estas nuevas necesidades han originado importantes cambios en los *SGBD* (*Sistemas de Gestión de Bases de Datos*) para que permitan representar problemas y dominios de información complejos. Estos sistemas utilizan tanto el modelo relacional clásico, como el modelo objeto-relacional, el cual está basado en una "simbiosis" entre el modelo relacional y el modelo orientado a objetos, lo que le permite la representación de dominios complejos de información bajo la simplicidad del modelo relacional.

Debido a esta circunstancia, este libro presenta una revisión actualizada del texto *Diseño y Uso de Bases de Datos Relacionales*, en la que se ha incluido la visión objeto-relacional de los datos, además del uso de las bases de datos a través de Internet.

OBJETIVOS DE LA OBRA

Este libro pretende aportar a una amplia gama de lectores una obra de consulta en la que se puedan apoyar tanto como una primera toma de contacto con las bases de

datos y los sistemas de gestión de bases de datos, como con las bases teóricas en la que se soporta esta tecnología, así como con los problemas y heurísticas a seguir en el proceso de traducción de los problemas del mundo real a una representación entendible por estos sistemas.

Si bien existen muchas obras relacionadas con el modelo lógico relacional, aunque no tantas sobre el modelo conceptual (*Entidad-Interrelación*), el lector debe recurrir a obras que, de forma independiente, tratan por un lado el modelo entidad-interrelación como medio de representación en el análisis de los problemas del mundo real, y por otro lado, a obras que tratan el modelo relacional como método de diseño de bases de datos para el manejo de la información correspondiente a estos problemas.

A los lectores interesados les sería de gran ayuda, para el desarrollo de una solución basada en las bases de datos, contar con referencias que:

1. Aporten una presentación clara y concisa del modelo entidad-interrelación para la representación de los problemas del mundo real.
2. Aporten una presentación formal del modelo lógico relacional como método de diseño para el manejo de la información correspondiente a estos problemas.
3. Establezca la heurística a seguir para la traducción del modelo conceptual al lógico, presentando las variantes principales y los problemas cotidianos que se pueden plantear con independencia del problema a tratar.
4. Aporten una primera toma de contacto con el uso del modelo objeto-relacional, y la aplicación de las bases de datos a sistemas que son explotados a través de Internet.

Esta obra pretende aportar esa información. Sin profundizar en las bases teóricas con un formulismo matemático excesivo sino, por el contrario, centrándonos en los conceptos y reglas fundamentales y, basándose en estos conocimientos, presentar los pasos a seguir en el proceso de análisis de los problemas orientados a su solución con el uso de base de datos, apoyados e ilustrados con casos prácticos que van avanzando en complejidad a lo largo de la obra.

Por ello, la obra incluye un amplio conjunto de ejemplos con problemas del mundo real completamente resueltos. Ejemplos en los que partiendo de una especificación o enunciado del problema, se desarrolle el modelo conceptual (se explique el proceso de obtención del mismo) y, partiendo de éste, se derive el modelo relacional, se normalice, se defina sintácticamente bajo un estándar (*Oracle*) este esquema, todo ello en un proceso paso a paso ampliamente explicado con detalle, para por último incluir ejemplos de manipulación de la información de cada una de las bases de datos ejemplos.

Además, la obra presenta ejercicios resueltos, tanto en el texto, como en el CD incluido, que hacen uso tanto del modelo relacional clásico, como del nuevo modelo objeto-relacional que incorpora las nuevas versiones de *Oracle*. Y se ha incluido un problema resuelto haciendo uso del producto *WebDB* de *Oracle*, mediante el cual se pueden construir "*sitios*" Web en los que existe una íntima interrelación entre la base

de datos, las páginas que componen el sitio, y el contenido o conjunto de información que es difundido en ese sitio.

ORIENTACIÓN A LOS LECTORES

La necesidad del conocimiento profundo de la representación conceptual y lógica (relacional) de problemas para el almacenamiento y manipulación de la información está justificada por:

DOCENCIA: El primer paso en el desarrollo de cualquier sistema y, sobre todo, en los sistemas de información, es el análisis del mismo, en el que es preciso el desarrollo de un modelo conceptual que describa los objetos y relaciones existentes en el dominio del problema. Además, hoy en día, las bases de datos más utilizadas son las que trabajan sobre *SGBD* relacionales, por lo que en la fase de diseño es necesaria la traducción del análisis a un modelo relacional perfectamente normalizado.

Este conocimiento debe ser y es impartido a los alumnos en áreas de *Bases de Datos, Ingeniería del Software, Diseño lógico, Sistemas de Información*, etc., por lo que una obra en la que se presenten estos conocimientos apoyados por un buen número de ejemplos resueltos resulta de interés.

DESARROLLO: Muchos de los sistemas informáticos existentes, en desarrollo y que se desarrollarán en el futuro en la empresa pública y privada, hacen uso de los *SGBD* relacionales (y entre ellos, muchos en *Oracle*). Estos sistemas requieren un proceso de diseño del esquema, por lo que el analista encargado de esta labor agradecerá la existencia de obras en las que se pueda apoyar (gracias a la base teórica y a los ejemplos) para el desarrollo de su trabajo con calidad.

La obra, por tanto, va dirigida tanto a los alumnos de informática (estudios medios y superiores), profesores de estas asignaturas (como libro de apoyo en la preparación de sus clases teóricas, prácticas y de problemas), y a profesionales del área, así como a todo usuario y desarrollador de sistemas de información.

CONTENIDO DE LA OBRA

PRIMERA PARTE *Introducción teórica*

Esta parte se dedica a desarrollar los conocimientos básicos sobre las *Bases de Datos*. El capítulo inicial se dedica a introducir el concepto de bases de datos, las ventajas e inconvenientes y los principales componentes de los *SGBD*. En el Capítulo 2 se presenta el modelo *E-R* como instrumento básico de representación conceptual de los problemas del mundo real y que servirá como base para el diseño de esquemas lógicos, independientemente del *SGBD*. En los Capítulos 3 y 4 se presenta el modelo relacional y los *SGBD* relacionales basados en el álgebra relacional, los lenguajes de definición y manipulación de datos, así como los principios básicos de traducción de esquemas conceptuales a relacionales. Por último, en el Capítulo 5 se presentan las

reglas a seguir en el proceso de traducción de los esquemas conceptuales a esquemas entendibles por los *SGBD* relacionales.

Los contenidos de esta primera parte pretenden abarcar todos los conocimientos necesarios y básicos para el conocimiento de las bases de datos, de los *SGBD* relacionales y de los lenguajes de definición y manipulación de la información. Asimismo, pretende introducir al lector al problema de diseño de bases de datos, paso previo y de vital importancia en el tratamiento, desempeño, integridad, etc. de las bases de datos y a las reglas de traducción de los esquemas conceptuales a relacionales.

SEGUNDA PARTE *Casos resueltos*

Esta sección tiene como objetivo asentar las bases teóricas introducidas en la primera parte del manuscrito, presentando una colección de casos de problemas del mundo real completamente resueltos. Diez ejemplos, completamente resueltos, que partiendo de un enunciado del problema, una especificación clara y precisa del mismo, se explica *paso a paso* el proceso de obtención de la representación conceptual utilizando diagramas *E-R*. A partir de estos diagramas, y utilizando los principios teóricos vistos anteriormente, se deriva el modelo relacional. Para cada caso resuelto se analizan los problemas de normalización y las dependencias funcionales, y se presentan una serie de interrogantes resueltos sobre los esquemas relacionales obtenidos haciendo uso de *SQL, y del lenguaje PL/SQL*, analizando la mejor solución, de entre las posibles, para la manipulación de la información propuesta en el ejercicio.

La estructura general del contenido para cada uno de los casos resueltos es: (a) enunciado del problema, (b) esquema conceptual, (c) esquema lógico, (d) validación del esquema lógico, (e) proposición, resolución y estudio, así como problemas resueltos sobre la manipulación de los datos del esquema lógico correspondiente. Si bien no aparecen todos estos epígrafes en todos los casos ni se presentan en el mismo orden.

En algunos de los problemas resueltos se ha hecho uso del modelo objeto-relacional para la construcción de la base de datos. De esta forma se han definido clases de objeto que son representadas y manipuladas como tablas dentro del esquema relacional propuesto. El uso de este modelo ha sido moderado, aplicándolo sólo en casos en los que el lector pueda fácilmente comprenderlos, lo que permite su aprendizaje en el mismo.

Los casos resueltos presentados en la obra no pretenden representar sistemas o problemas completos, pues en ese caso se necesitaría un manuscrito independiente para cada uno de ellos por muy simples que fueran. Si bien los casos planteados consideran problemas conocidos por cualquier tipo de lector (al menos eso hemos pretendido), las características y, por tanto, definición de los mismos, se han simplificado para poder presentar una serie de casos que, avanzando en complejidad a lo largo de la obra, están lo suficientemente restringidos en cuanto a sus características para ser abordados en una obra de este tipo.

Al final de la obra, el Capítulo 16, está dedicado a la integración de las *Bases de Datos* e *Internet*. En este capítulo se presenta el producto *WebDB* de *Oracle* apoyándonos en un nuevo problema que se ha resuelto haciendo uso de este software. Una vez planteado y resuelto el problema, se presenta la manipulación de la base de datos a través de páginas Web, haciendo uso del *WebDB*. Las páginas y procedimientos desarrollados se incluyen en el CD de forma que el usuario, además de consultar los ficheros fuentes, puede ejecutarlos lo que le ayudará a mejorar sus conocimientos en el uso de bases de datos a través de Internet.

Por último, en la obra se aportan una serie de referencias bibliográficas de fácil acceso a los lectores. El acceso a estas referencias, ya clásicas en esta área, garantizará al lector un conocimiento profundo de aquellas partes teóricas y prácticas no tratadas en esta obra. Si bien el número de referencias aportadas es reducido, consideramos que es más que suficiente, además de no causar al lector dispersión en cuanto a cuál referencia debe o necesita adquirir en un momento dado. Será la bibliografía aportada en estas otras obras la que, en caso de necesidad, permita al lector dirigirse a obras más específicas sobre un determinado tema o área de interés.

CONTENIDO DEL CD

Junto con la obra se incluye un CD en el que el lector puede encontrar tanto los problemas resueltos en el texto de la obra, como un amplio número de nuevos ejercicios adicionales. De esta forma, el lector contará con un conjunto de ejercicios de uso y manipulación de las bases de datos correspondientes a los problemas resueltos (Capítulos 4, y 6 a 15) que le ayudarán al asentamiento de los conocimientos.

Todos estos ejercicios pueden ser consultados por el lector a través de cualquier editor de ficheros texto, o bien a través de la interfaz construida a tal fin e incluida en el CD. A través de esta interfaz que se ejecuta con cualquier navegador estándar, el lector podrá consultar estos ejercicios, acceder a una serie de ejercicios propuestos, e instalar y ejecutar la aplicación desarrollada con *WebDB* de *Oracle* y descrita en el Capítulo 16.

CAPÍTULO 1

INTRODUCCIÓN A LAS BASES DE DATOS

El término de *Bases de Datos* no apareció hasta mediados de los años sesenta, época en la cual la información era representada haciendo uso de un conjunto de ficheros, generalmente planos. Estos ficheros no estaban relacionados entre sí, y los datos almacenados representaban las relaciones existentes en la información que representaban mediante referencias simbólicas y/o físicas. La redundancia era grande y la integridad de la información representada dejaba mucho que desear.

Aun así, muchos desarrolladores de software *bautizaban* a sus sistemas de ficheros como *Bases de Datos*, sin preocuparse de que cumplieran o no una serie de propiedades que deben acompañar al uso de este término, cualidades que se describirán más adelante en este capítulo.

Para que se denomine a una *Base de Datos* como tal, debe satisfacer una serie de propiedades, las cuales fueron incorporándose a estos sistemas a medida que el software para la administración de la información que se desarrolló fue siendo más eficaz. Hay que tener en cuenta que, hoy en día, no todas las bases de datos satisfacen estas propiedades *ideales*, por lo que el analista de sistemas se ve obligado a una harmonización de las cualidades deseables de una base de datos, a menudo contrapuestas.

Los procedimientos y estructuras para el almacenamiento y mantenimiento de la información correspondientes a un determinado dominio de un problema han evolucionado a medida que lo ha hecho la tecnología. Inicialmente los dispositivos de almacenamiento sólo permitían un acceso serial a la información, por lo que las

estructuras de datos, mediante las que se podía representar la información, debían ser muy simples (archivos con organización de apilo o secuenciales) y los procedimientos de acceso a esta información requerían un alto tiempo de cómputo puesto que eran puramente seriales. Además, los procedimientos encargados del mantenimiento de la información eran totalmente dependientes del hardware utilizado para ello, lo que suponía una continua modificación del software encargado de esta tarea cuando el hardware cambiaba.

Con la aparición de los dispositivos de almacenamiento que permitían el acceso directo, generalmente denominado aleatorio, las estructuras mediante las cuales se podía representar la información se fueron haciendo más complejas. En esta época (histórica) se desarrollaron procedimientos de acceso directo a la información, si bien seguían siendo estos procedimientos los encargados de describir la estructura del almacenamiento de la información que trataban. Si la estructura cambia debido a cualquiera de las múltiples razones posibles (cambios en los requisitos del cliente del sistema o cambios en el entorno del sistema, lo que puede suponer la modificación de la estructura de los registros que se almacenan en los archivos), los procedimientos debían ser modificados para reconocer la nueva estructura de la información. En esta época, aunque el almacenamiento de la información sí era independiente del dispositivo hardware utilizado, la estructura de la información no era independiente de los procedimientos que la manejaban.

Como los sistemas son dinámicos, los requisitos cambian con el tiempo, la información a ser tratada en cada problema también cambia y, por tanto, es necesario, de alguna manera, independizar la estructura de la información (los archivos encargados de almacenarla) de los procedimientos encargados de su tratamiento, si no se estaría siempre abocado a la dedicación de una gran cantidad de esfuerzo a la modificación de todos aquellos procedimientos encargados del mantenimiento de la información.

Cuando se reconoce que los sistemas evolucionan y que, por tanto, la información y la estructura de la misma no es estática sino que va cambiando con el tiempo, es cuando aparece el concepto de las *Bases de Datos*. Si se desea que cualquier modificación en la cantidad, contenido y estructura de la información que se desea mantener acerca de un determinado problema no afecte a los procedimientos desarrollados previamente para el mantenimiento de la misma, es necesario tener en cuenta que existe una *independencia de los datos con respecto a los procedimientos*. El software, por tanto, debe referenciar los datos al nivel de ítem de datos; es decir, a nivel de atributo o propiedad de los objetos que forman parte o intervienen en el problema, y no a nivel de objeto. Así, la descripción lógica de un registro (un objeto abstracto de interés procedimental) puede contener, para un procedimiento, ítems de datos que son distintos a los que aparecen para otro procedimiento para ese mismo registro.

La independencia de los datos con respecto a los procedimientos supone, como se verá a lo largo de este capítulo, que la visión conceptual de los datos, tal y como se perciben de la observación del problema del mundo real, no tiene por qué ser la misma

que la visión física de los mismos, la estructura de los archivos utilizados para su almacenamiento. Si los procedimientos encargados del mantenimiento de la información sólo "*ven*" la estructura física de los datos y si ésta se realiza a nivel de ítem de datos, un cambio en la visión conceptual no tiene por qué afectar, en principio, a estos procedimientos.

Esta independencia de la información con respecto a los procedimientos que la maneja debe satisfacerse a dos niveles de abstracción para que sea efectiva; por tanto, se habla de:

Independencia lógica de los datos, por la que la modificación de la representación lógica general del dominio del problema no afecta a los programas de aplicación que la manipulan, siempre que esta modificación no elimine ninguno de los ítems de datos que estos programas requieran.

Independencia física de los datos, por la que la distribución de los datos en las unidades de almacenamiento y la estructura física de la información almacenada es independiente de los cambios de la estructura lógica general de la información y, por tanto, de los procedimientos que manejan la misma.

1.1. CARACTERÍSTICAS DE LAS BASES DE DATOS

La información que forma parte de una base de datos puede organizarse de múltiples formas pero con independencia de la arquitectura de la base de datos, ésta debe cumplir una serie de características para ser considerada como tal, algunas de las cuales se describirán a continuación:

Versatilidad para la representación de la información: Si bien la información que forma parte del dominio de un problema es única y caracteriza a ese problema o sistema, pueden existir diferentes visiones de esa información. Visiones parciales en las que sólo se tiene en cuenta parte del dominio del problema y/o visiones globales que observan el problema desde diferentes puntos de vista.

Un procedimiento, un programa de aplicación, que maneja la información correspondiente a un problema puede, por tanto, tener en cuenta sólo parte del conjunto de información, mientras que otro procedimiento puede considerar a otro conjunto diferente, o no, de información del mismo problema.

Si se considera que un procedimiento "*ve*" la información que maneja como un registro, la organización de la información en la base de datos debe permitir que diferentes procedimientos puedan construir diferentes registros a partir de la información existente en la base de datos. Estos registros (lógicos) estarán formados por ítems de datos que forman parte del dominio del problema y que son derivados del conjunto de los ítems de datos existentes en ese problema y, además, estos registros lógicos construidos por los procedimientos deben ser independientes de los registros físicos existentes en la base de datos para almacenar la información.

Desempeño: Las bases de datos deben asegurar un tiempo de respuesta adecuado en la comunicación hombre-máquina, permitiendo el acceso simultáneo al mismo o distinto conjunto de ítems de datos por el mismo o distinto procedimiento.

Mínima redundancia: Una de las principales razones por las que surgió la tecnología de las bases de datos fue el evitar la alta redundancia que se presentaba en los sistemas de procesamiento de la información debido al uso de archivos con estructuras planas. Sin embargo, las bases de datos no evitan totalmente la redundancia en la información debido a que es necesario representar todas las relaciones que existen entre las entidades que forman parte del dominio del problema.

La existencia de redundancia es nefasta debido a la posibilidad de inconsistencia en la información almacenada en la base de datos. La redundancia implica la existencia de varias copias de un mismo ítem de datos, las cuales pueden, en un momento dado, tener distintos valores. Además, hay que tener en cuenta que esta duplicación implica unas necesidades de almacenamiento innecesarias que, aunque el coste del almacenamiento por bit resulte cada vez menor, siempre implicarán un coste innecesario.

Un objetivo principal de las bases de datos es eliminar la redundancia siempre que ello no implique una complejidad de la misma y/o una disminución en el desempeño. Si existen diversas copias de un mismo ítem de datos, es necesario establecer procedimientos que garanticen la consistencia de la información cuando estas copias sean utilizadas por diferentes procedimientos al mismo tiempo. Por otro lado, si sólo existe una copia de un ítem de datos es necesario establecer procedimientos que permitan el acceso a esta copia por varios procedimientos, para garantizar un desempeño aceptable, lo que complica el software de gestión y la representación del dominio (a cualquier nivel) del problema en la base de datos.

El diseñador de la base de datos debe adoptar una solución de compromiso, aunque siempre con el objetivo de que exista una redundancia mínima en la base de datos.

Capacidad de acceso: Los usuarios de la base de datos reclaman a ésta continuamente información sobre los datos almacenados. Estos interrogantes *contra* la base de datos, que pueden ser conocidos de antemano o no, cuando se diseñó la misma, solicitan información correspondiente a distintos ítems de datos, así como sus relaciones, representadas en la base de datos y, por añadidura, agrupados, formateados, etc., de múltiples formas.

Una base de datos debe ser capaz de responder, en un tiempo aceptable, a cualquier consulta sobre la información que mantiene, sin restricciones graves en cuanto a los ítems, relaciones, formato, etc., solicitados en la misma, y respondiendo al usuario rápidamente.

Esta característica va a depender directamente de la organización física de los datos en la base de datos. De nuevo una solución de compromiso deberá ser

adoptada por el diseñador de la misma. Una organización física *muy completa* garantiza una respuesta rápida a las consultas, aunque requiere un mayor coste computacional en actualizaciones (entre otras razones debido a la redundancia que se añade) y viceversa.

Simplicidad: La base de datos representa el dominio de un problema que se necesita tratar computacionalmente. La naturaleza de este problema puede ser muy variada y, por tanto, existir en el mismo un número de objetos variable que se relacionan de múltiples formas. Por ello, es la naturaleza del problema un factor de complejidad de partida de las bases de datos que es conveniente eliminar para que se garanticen otras de las características que se le requieren.

Las bases de datos deben estar basadas en representaciones lógicas simples que permitan la verificación en la representación del problema que representan y, más aún, la modificación de los requisitos en el mismo, de tal forma que la inclusión y/o modificación de nuevos ítems de datos y relaciones no ocasione una complejidad excesiva.

Integridad: La integridad de una base de datos hace referencia a la veracidad de los datos almacenados con respecto a la información existente en el dominio del problema que trata la misma. Como los datos de la base de datos son manejados por muchos usuarios haciendo uso de muchos procedimientos que tratan los mismos datos de muchas formas, es necesario garantizar que estos datos no sean destruidos ni modificados de forma anómala (naturalmente, este control debe tenerse en cuenta tanto con el valor de los ítems de datos como de las relaciones existentes entre ellos).

Durante el procesamiento se pueden producir fallos de muy diversa naturaleza: errores del sistema general, del hardware, software, etc. Así, los procedimientos que manejan la información (en inserción, borrado y actualización) deben asegurar que el sistema pueda garantizar la integridad de la información a pesar de los errores que se puedan producir, temporalmente, a causa de los fallos con independencia de su naturaleza.

Pero además de garantizarse la integridad de la base de datos con respecto a este tipo de fallos, esta integridad debe garantizarse con respecto a la veracidad de los ítems de datos y sus relaciones con respecto al dominio del problema. Así, en una base de datos deben establecerse procedimientos que verifiquen que los valores de los datos se ajustan a los requisitos y restricciones extraídas del análisis del problema.

Seguridad y Privacidad: La seguridad de una base de datos hace referencia a la capacidad de ésta para proteger los datos contra su pérdida total o parcial por fallos del sistema o por accesos accidentales o intencionados a los mismos. Mientras que la privacidad de una base de datos hace referencia a la reserva de la información de la misma a personas no autorizadas.

La información de la base de datos es de vital importancia y valor para la organización/empresa responsable de la misma. A medida que esta información es

más importante, tanto más valioso se hace para la empresa el mantener su seguridad y privacidad. Para conseguir estas características, una base de datos debe satisfacer, al menos, los siguientes requisitos:

- Seguridad contra la destrucción de los datos causada por el entorno: fuego, robo, inundaciones y cualquier otra forma.
- Seguridad contra la destrucción de los datos causada por fallos del sistema (hardware y software), de forma que los datos puedan reconstruirse.
- Seguridad contra accesos no autorizados a la base de datos.
- Seguridad contra accesos indebidos a los datos.

Por lo que deben existir en la base de datos tanto procedimientos de recuperación de la información perdida total o parcialmente por cualquier causa, como procedimientos que supervisen el acceso a los datos por los usuarios de la base de datos.

Afinación: La afinación hace referencia a la organización física de la información de la base de datos, la cual determina directamente el tiempo de respuesta de los procedimientos que operan sobre la misma.

Si una de las características que debe tener una base de datos es un buen desempeño, la organización física de los datos debe ser tal que ésta pueda ser alcanzada. Pero la base de datos evoluciona con el tiempo, el volumen de información va haciéndose cada vez más importante y, por añadidura, tanto los ítems de datos como las relaciones entre ellos pueden ampliarse y/o modificarse. Esto implica que una buena organización física de los datos en un momento dado, puede no ser tan buena en otro.

Por ello, la base de datos debe ser flexible a la modificación de esta organización física, lo que puede suponer además una migración de los datos según evolucione la base de datos, sin que por ello se vean afectados los procedimientos u otras representaciones de los datos pero, sin embargo, se consiga un desempeño más alto.

Interfaz con el pasado y el futuro: Es aceptado que el dominio de un problema, el problema en sí, cambia evolucionando con el tiempo. Las necesidades de la organización cambian continuamente y, por lo tanto, cambia la información correspondiente al subsistema o dominio del problema de la misma. Una base de datos debe estar abierta a estos cambios de forma que no afecten, o afecten lo mínimo posible, a los procedimientos existentes para manejar la información que mantiene.

Por otro lado, una base de datos debe estar abierta a reconocer información organizada físicamente por otro software (de base de datos o no) de distinta forma a la que utiliza la base de datos. Información existente antes de la implantación, por la empresa, de la base de datos (actual) y que es valiosa para la organización. Además, el avance tecnológico hará que la base de datos actual pueda ser cambiada con el tiempo, por lo que esta necesidad, aunque no es inminente para la empresa, sí será una característica importante a exigir a estos sistemas.

1.2. LAS DIFERENTES VISIONES DE LOS DATOS EN LAS BASES DE DATOS

Para que una base de datos pueda satisfacer las características antes señaladas, y otras más, es necesario que los usuarios de la misma tengan una visión abstracta de los datos almacenados. Es decir, el usuario, a diferencia de lo que ocurría con el uso de las organizaciones clásicas para el almacenamiento de la información, no tiene necesidad de conocer cómo se organizan los datos físicamente en la base de datos.

El usuario, experto o lego en informática, conoce las características del problema que representa la base de datos. El usuario conoce la organización, todo o parte del sistema, la información que se maneja, las relaciones existentes entre esta información, cómo debe ser tratada esta información y en qué tiempo. En definitiva, el usuario conoce el dominio del problema que va a ser tratado con una base de datos. Si es ésta la visión de los datos que la base de datos presenta al usuario, éste podrá utilizar el sistema e integrarse en él adecuadamente.

Si por el contrario, la base de datos presenta al usuario la visión de cómo está organizada la información físicamente, tanto los datos como las múltiples relaciones que pueden existir entre ellos, éste no reconocería el problema de la organización que está tratando si no tiene una alta formación para reconocer, en ocasiones, estructuras físicas muy complejas.

Dependiendo de quién acceda o use la base de datos, ésta debe presentarle una visión de los datos que sea capaz de reconocer, interpretar y manejar. Además, si una base de datos debe satisfacer las características antes expuestas, la organización física de los datos debe ser lo más independiente posible de los procedimientos que manejan la información y de los posibles cambios que surjan en el dominio del problema.

Se puede entonces hablar de que existen tres visiones de los datos en una base de datos:

Visión externa: Es la visión de los datos que tienen los usuarios finales de una base de datos. Un usuario (un operador de terminal) trata sólo una visión parcial de la información, sólo aquella que interviene en el dominio de actividad (el subsistema de la organización en el que interviene). Este usuario debe *ver* la información que maneja como un registro, una ficha de datos (un formulario) con independencia de a qué objeto pertenecen los ítems de datos, correspondientes a ese registro, en el dominio del problema (sistema) y en qué relaciones se ven implicados esos datos.

Por otro lado, otro usuario (del mismo o cualquier otro subsistema) verá también su *registro particular* de información cuyos ítems de datos podrán ser comunes, o no, al de otros *registros particulares* de otros usuarios.

Estas *visiones particulares* de los usuarios son proporcionadas por los procedimientos o programas de aplicación que sólo manejan parte de la información de la base de datos.

Visión conceptual: Es la visión o representación del problema tal y como éste se presenta en el mundo real. Una base de datos representa la información que es observada en el mundo real con respecto a un determinado problema. En esta observación, en el análisis del problema, se determinan los objetos o entidades que intervienen en el mismo, las propiedades o características de estas entidades y las relaciones o dependencias que existen entre ellos.

La visión conceptual de una base de datos es una representación abstracta del problema e independiente, en principio, de cómo va a ser tratada esta información, de qué visiones externas pueda tener y de cómo esta información pueda ser almacenada físicamente. Así, la visión conceptual de una base de datos no cambia a no ser que cambie la naturaleza del problema.

Visión Física: La visión física de una base de datos es la representación de cómo la información es almacenada en los dispositivos de almacenamiento. Esta visión describe las estructuras u organizaciones físicas, dispositivos, volúmenes, ficheros, tipos de datos, punteros, etc., estructuras de mayor o menor complejidad que representan el dominio del problema de una forma entendible por el sistema informático.

Si los usuarios de las bases de datos pueden percibir tres visiones diferentes de los datos, es debido a que en una base de datos existen, al menos, tres formas diferentes de descripción de la información que almacena. Pero, como el dominio del problema que representa una base de datos es el mismo, entonces estas tres formas de descripción son, en realidad, tres niveles de abstracción diferentes que describen un mismo problema (véase la figura 1.1).

En cada nivel se describen aquellos objetos de interés que pueden ser entendidos por los usuarios, de ese nivel, de la base de datos[1]. Así, el usuario final sólo entiende de registros (visión o nivel de abstracción externo) o formas de comunicación similares a los documentos externos que son manejados en la organización, mientras que el diseñador o analista de sistemas sólo entiende de tipos de entidades o clases de objetos que intervienen en el dominio del problema que la organización desea que se trate mediante una base de datos, de las relaciones existentes entre ellas y de los procedimientos que son llevados a cabo en la organización para la solución del problema que se está tratando (nivel de abstracción conceptual) y, por otro lado, es el administrador de la base de datos el encargado de describir el nivel físico para determinar aquella organización física que pueda garantizar el desempeño óptimo del sistema (nivel de abstracción físico o interno).

[1] Como se describirá más adelante, una base de datos tiene diferentes tipos de usuarios, los cuales sólo observan aquella parte de la base de datos que les interesa y a un nivel de abstracción que son capaces de entender.

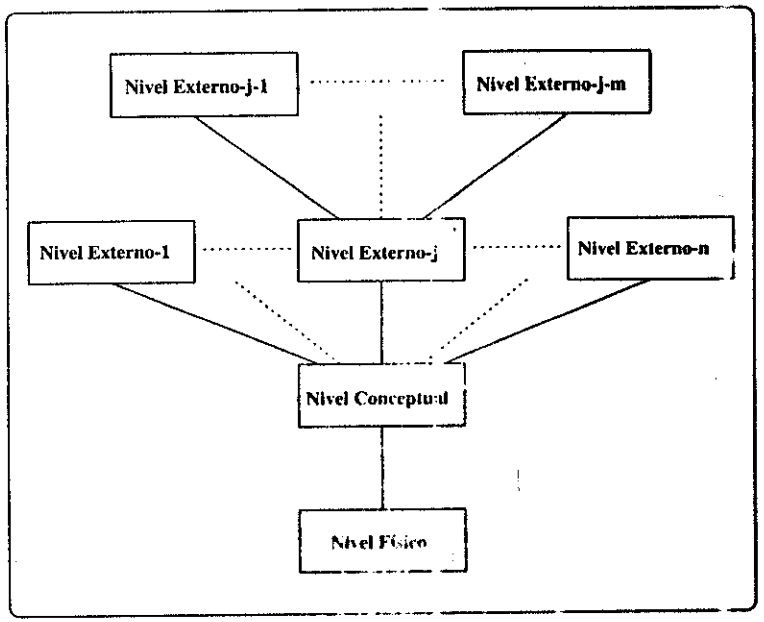


Figura 1.1 Tres visiones diferentes de las bases de datos

La descripción de los datos a estos tres niveles de abstracción diferentes garantiza (no del todo, como veremos más adelante) la independencia de los datos, uno de los objetivos principales de las bases de datos. Es decir:

- Que pueda ser modificada la organización física de los datos sin que por ello haya cambiado o deba cambiar la descripción conceptual, y sin que por ello tengan que ser modificados los programas de aplicación (nivel de abstracción externo) que manipula esta información.
- Que pueda ser modificada la representación conceptual del problema que está siendo representado en la base de datos, debido a la consideración de nuevas entidades, o relaciones, o cambios en las características de las mismas, y sin que por ello tenga que ser modificada la estructura física de la información (nivel interno) (naturalmente, si se consideran nuevos objetos, éstos deberán ser reflejados), ni los programas de aplicación (nivel de abstracción externo), naturalmente siempre y cuando no se eliminen de la representación conceptual objetos necesarios o requeridos en estos otros niveles.
- Evidentemente, las visiones externas pueden cambiar conforme nuevos requisitos o necesidades funcionales o de operación son incorporados al dominio del problema por la organización y/o su entorno, y sin que por ello deba ser modificada ninguna de las descripciones de los datos a ninguno de los restantes niveles de abstracción.

Para que exista esta independencia tan deseada de los datos los tres niveles de abstracción mediante los cuales los datos son descritos deben ser completamente independientes, lo cual no es del todo cierto en la mayoría de las bases de datos. Sin embargo, puede conseguirse una buena independencia de los datos si:

- La representación interna de los datos no es una *traducción* dependiente de la representación conceptual. Una misma representación conceptual puede representarse de varias formas físicamente. Serán los requisitos funcionales y de desempeño los que determinen esta representación.
- Si bien las representaciones externas son dependientes de la representación conceptual por el hecho de que los registros (representaciones externas) deben estar formados por ítems de datos existentes y, por tanto, representados en el nivel conceptual (lo mismo se puede decir entre el nivel físico y conceptual), la estructura de estos registros (número de ítems y disposición de los mismos) debe ser independiente de cómo estos ítems han sido representados en el nivel conceptual y de las relaciones que mantienen en el mismo.

1.2.1. Independencia del nivel de descripción conceptual

El nivel de descripción conceptual es seguramente el más importante, o por lo menos aquel en el que se apoyan en menor o mayor grado los otros niveles y, con seguridad, en el que, en base a su calidad, se garantiza que la base de datos solucione el problema de la organización.

El nivel conceptual describe el sistema de la organización, o aquella parte del sistema, el dominio del problema, que se desea tratar. Una descripción conceptual de calidad describirá todas y cada una de las entidades o clases de objetos que intervienen en el problema, sus propiedades y atributos, así como las características de las relaciones existentes entre las mismas. En este nivel se describe cada uno de los ítems de datos o elementos de información que intervienen en el comportamiento del sistema y cuya información es necesario considerar.

Pero existen muchas formas de describir un sistema, todas ellas más o menos válidas y correctas para obtener una visión conceptual de un determinado problema, aunque cualquier procedimiento no puede reconocer e interpretar cualquier descripción. Se puede hacer una descripción severa de un problema utilizando técnicas como los diagramas de estructuras, de contexto, tablas de cualquier tipo (procesos/datos, historia de la vida de la entidad, procesos/procesos, entidad/procesos, etc.), diagramas de entidades y relaciones, árboles, tablas, redes, etc. y, sin embargo, no todos los procedimientos son capaces de interpretar cualquier clase de representación conceptual.

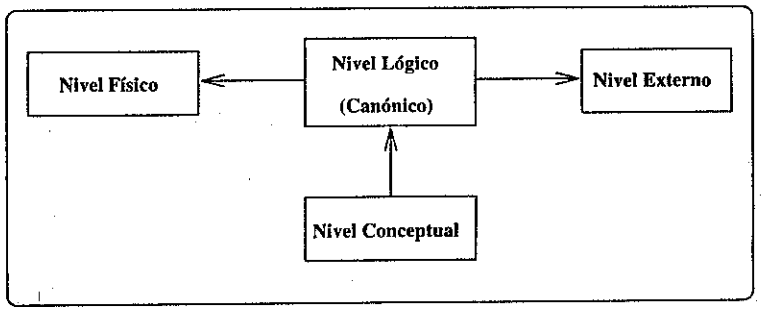


Figura 1.2 Niveles de abstracción de la base de datos

De hecho, como se tratará más adelante, existen muchas formas de representar de forma abstracta un fenómeno (un problema) observado del mundo real. Una representación abstracta (conceptual) de un problema supone la aplicación de una serie de reglas que restringen y dirigen la forma en que ese problema es representado. Pero, por otro lado, el fenómeno del mundo real o problema que se está representando debe ser y, de hecho lo es, independiente de la forma en que el *ser humano* sea capaz de representarlo.

Por ello, se puede hablar de un cuarto nivel de abstracción en la representación de la información en una base de datos (ver figura 1.2), el *nivel lógico o canónico*.

La descripción conceptual es independiente de las descripciones externas e interna, puesto que sólo depende del problema del mundo real objeto de la representación. Si el problema no cambia, no cambia la representación conceptual, aunque cambien los mecanismos por los cuales el problema será tratado y, por tanto, alguna de las otras representaciones.

La representación canónica, sin embargo, sí es dependiente de la forma, mecanismos, o procedimientos por los cuales la información correspondiente al problema va a ser manipulada. La descripción canónica es derivada de la descripción conceptual en base a la aplicación de una serie de reglas y restricciones que tienen en cuenta cómo la información representada puede ser tratada por los procedimientos que van a manejar y definir la información en base a las otras representaciones. Así, pueden existir muchas representaciones lógicas de una misma representación conceptual, al igual que de una representación lógica pueden ser derivadas muchas representaciones externas.

La inclusión de este nuevo nivel de representación del dominio del problema, un nivel dependiente del software encargado de manipular la información, va a garantizar la independencia de la información en una base de datos.

1.2.2. Granularidad y ligadura

Si bien, como se ha descrito hasta ahora, es posible la descripción de los datos a niveles de abstracción diferentes para que se garantice la independencia de la información con respecto a los procedimientos que la manipulan a cualquiera de los niveles (externo e interno), es obvio pensar que esta independencia tendrá unas restricciones que limitan su ámbito.

Si la descripción externa (los registros que *ven* los usuarios a través de los programas de aplicación) debe ser derivada de la descripción lógica, el nivel de detalle de la descripción lógica limitará, por tanto, la independencia por la cual pueden ser las descripciones externas derivadas.

Si, por ejemplo, el nivel de detalle con que se pueda representar la información en el nivel lógico es al nivel de entidad, entonces las descripciones externas deberán considerar para la descripción de cada registro a entidades descritas en el nivel lógico; es decir, un registro que deba considerar a dos entidades descritas en el nivel lógico deberá considerar a todos los ítems de datos que forman parte de estas entidades con independencia de si son manejados por el programa de aplicación que considera ese registro.

Mientras que si en el nivel lógico la información puede describirse al nivel de agregado de dato, o ítem de dato, los registros descritos en el nivel externo podrán considerar sólo aquellos ítems de datos de las entidades descritas en el nivel lógico que sean de interés para el programa de aplicación.

Al nivel de detalle en que pueden ser descritas las representaciones externas derivadas de la representación lógica se le denomina *Granularidad*. A mayor granularidad de una representación externa (menor información a considerar) mayor será la independencia, y viceversa. Es decir, una representación externa será más independiente de la representación lógica si pueden *construirse* registros en base a ítems de datos en lugar de en base a agregados de datos o en base a entidades definidas o existentes en la representación lógica. Una mayor granularidad proporciona una mayor independencia, pero al mismo tiempo una mayor complejidad en el software utilizado para realizar estas representaciones y, por añadidura, en la organización de la información para que estas representaciones puedan ser almacenadas físicamente.

Suponiendo que en una base de datos pueda existir una independencia completa entre las distintas representaciones de los datos con respecto a los procedimientos que manejan la información para cada representación, es obvio que si se desea mantener la integridad de la representación de un problema, la integridad de la base de datos y los procedimientos que manejan cada representación no pueden realizar acciones que puedan dar lugar a una inconsistencia entre el problema representado en cada nivel, simplemente porque el problema es el mismo, con independencia del nivel de abstracción al cual se represente.

Para garantizar la integridad de la base de datos es, por tanto, necesario que en *algún instante* esos procedimientos que manipulan las representaciones a un determinado nivel de abstracción tengan en cuenta cómo se representa la información en los otros niveles. En ese *instante* las diferentes representaciones de los datos se vinculan entre sí y, por tanto, la independencia entre ellas se pierde. A este proceso de vinculación de las diferentes representaciones de la información en la base de datos se le denomina *ligadura*, y ésta es de dos tipos:

Ligadura lógica, correspondiente al proceso de vinculación que se produce entre las representaciones externas y la lógica.

Ligadura física, correspondiente al proceso de vinculación entre la representación lógica y la física.

Si cuando se produce la ligadura se vinculan las representaciones de los datos en los diferentes niveles de abstracción, ello implica que la independencia de los datos se pierde y, por tanto, sería conveniente retrasar lo más posible el proceso de vinculación. El proceso de vinculación entre las distintas representaciones puede realizarse en cualquiera de las siguientes fases: *Compilación, enganche, ejecución y acceso a la base de datos.*

La independencia de los datos será mayor cuanto más tardía se realice la ligadura. Así, aquellos sistemas que realicen la ligadura en la fase de acceso a los datos mantendrán esta independencia hasta el último momento y sólo vincularán aquellos datos a los que se pueda acceder y se vean involucrados en cada uno de los accesos. Mientras que los sistemas que realizan la ligadura en la fase de compilación pierden la independencia de los datos en el momento en que los programas de aplicación se traducen a código objeto.

Si la ligadura se realiza en una fase muy temprana (compilación, enganche) implica que los programas de aplicación deberán ser recompilados (o reenlazados) cada vez que se produzca una modificación de las distintas representaciones de los datos (lógica y física), aunque por otro lado el desempeño de los mismos será alto debido a que el tiempo de cómputo que conlleva el proceso de ligadura se consume sólo una vez y no en el arranque del programa o en cada acceso a los datos.

Por otro lado, el que se realice la ligadura en una fase tardía (en cada acceso a los datos) supone que se podrán modificar las representaciones lógica y física de los datos sin que por ello deban traducirse de nuevo a código máquina los programas de aplicación. Sin embargo, el desempeño de estos programas será en principio menor debido a que necesitan un coste computacional añadido para realizar el proceso de ligadura, bien en el arranque del programa o en cada uno de los accesos a los datos.

La solución de compromiso adoptada por muchos sistemas es que el proceso de ligadura se realice en la fase de ejecución. Esto supone un coste computacional añadido sólo en el arranque del programa de aplicación, pero libera de una continua

recompilación de las aplicaciones cada vez que se alteren las representaciones de los datos.

1.3. BASES DE DATOS Y SISTEMAS DE GESTIÓN DE BASES DE DATOS

En las secciones previas se han descrito las características que debe tener una base de datos y las diferencias con los sistemas de almacenamiento convencionales, luego el lector puede, en estos momentos, tener una idea clara de lo que es una base de datos. Aunque al principio los organismos de estandarización[2] intentaron dar una definición a este término, ésta no fue totalmente aceptada, surgiendo a lo largo de los años diferentes definiciones que intentaron adaptarse a los cambios tecnológicos y nuevas características o propiedades que se les ha ido exigiendo a las bases de datos.

Así, han sido propuestas innumerables definiciones para las bases de datos las cuales intentan exponer en una corta definición, tarea bastante difícil, todas aquellas características exigibles a las bases de datos.

Los autores de esta obra proponen, cómo no, la siguiente definición de una base de datos, la cual con toda seguridad es una recopilación de muchas otras ya propuestas:

> ***Una Base de Datos** es una colección de archivos relacionados que almacenan tanto una representación abstracta del dominio de un problema del mundo real cuyo manejo resulta de interés para una organización, como los datos correspondientes a la información acerca del mismo. Tanto la representación como los datos están sujetos a una serie de restricciones, las cuales forman parte del dominio del problema y cuya descripción está también almacenada en esos ficheros.*

De la que podemos extraer las siguientes consideraciones:

- Se trata de una colección de archivos relacionados; es decir, a diferencia de los sistemas clásicos que manejan organizaciones clásicas de almacenamiento, en una base de datos los archivos no son independientes entre sí, la base de datos puede ser vista como un único depósito en el cual se almacena toda la información correspondiente al dominio de un problema.
- En estos archivos se encuentra almacenada tanto la *representación abstracta del dominio del problema*: es decir, la visión física, lógica y cada una de las visiones externas de la información, como *los datos* conocidos acerca del mismo en un momento dado.
- El que tanto la representación como los datos están sujetos a una serie de restricciones implica:

[2] Inicialmente Codasyl a través de su grupo DBTG (Data Base Task Group), y después ANSI a través de su grupo X3/SPARC.

  - Que las restricciones innatas al problema están representadas. Restricciones acerca de las propiedades de las entidades, datos y relaciones existentes en el dominio del problema.
  - Que el acceso a la información almacenada está sujeto a una serie de restricciones que garantizan la integridad de la misma. Estas restricciones impiden que algún procedimiento viole las reglas que vinculan los datos entre los diferentes niveles de representación.

Es importante conocer la diferencia entre lo que es una base de datos y lo que es un *Sistema de Gestión de Bases de Datos*, términos que se confunden muy a menudo cuando se está trabajando con la información haciendo uso de esta tecnología.

Hasta estos momentos se ha estado tratando únicamente el término de bases de datos, aportando una definición en esta misma sección. Cuando se habla de bases de datos se habla de información que está almacenada cumpliendo toda una serie de características y restricciones como las que se han expuesto en este capítulo.

Pero para que la información pueda ser almacenada como se ha descrito y el acceso a la misma satisfaga las características exigidas a una base de datos para ser denominada como tal, es necesario que exista una serie de procedimientos (un sistema software) que sea capaz de llevar a cabo tal labor. A este sistema software es a lo que se le denomina *Sistema de Gestión de Bases de Datos*[3].

Así, un *SGBD* es una colección de programas de aplicación que proporcionan al usuario de la base de datos los medios necesarios para realizar las siguientes tareas:

- Definición de los datos a los distintos niveles de abstracción (físico, lógico y externo).
- Manipulación de los datos en la base de datos. Es decir, la inserción, modificación, borrado y acceso o consulta a los mismos.
- Mantenimiento de la integridad de la base de datos. Integridad en cuanto a los datos en sí, sus valores y las relaciones entre ellos.
- Control de la privacidad y seguridad de los datos en la base de datos.
- Y, en definitiva, los medios necesarios para el establecimiento de todas aquellas características exigibles a una base de datos.

1.4. COMPONENTES DE LOS SGBD

Para realizar todas las funciones descritas en la sección anterior, y otras más, es necesario que el *SGBD* cuente con una serie de componentes cuya función sea el desarrollo de las mismas de forma que satisfaga los requisitos impuestos para estos sistemas.

[3] Término al cual nos referiremos, únicamente por razones de simplificación, como *SGBD*.

Cuando se utiliza el término de *componentes* de un *SGBD* se está realizando una, y quizás peligrosa, generalización, puesto que estos componentes son muy variados. Así, un *SGBD* cuenta tanto con herramientas software como con personal humano especializado en la realización de las tareas y acciones necesarias para la gestión adecuada de la información[4].

1.4.1. El lenguaje de definición de datos

Si para garantizar la independencia de los datos es necesaria la definición de éstos a diferentes niveles de abstracción es, por tanto, necesario que el *SGBD* cuente con un componente que permita la realización de esta tarea. El lenguaje de definición de los datos —*Data Definition Language (DDL)*— es un lenguaje artificial basado en un determinado modelo de datos que permite la representación lógica de los datos.

Generalmente, los *DDL* de los diferentes *SGBD* son lenguajes simples basados en una gramática sencilla que cuenta con un conjunto muy reducido de morfemas, lo que garantiza la definición no ambigua de los datos. Esta definición debe ser compilada para dar lugar a una representación orientada a la máquina que es la que utiliza el *SGBD* en tiempo de procesamiento.

La representación de los datos obtenida en este proceso de compilación es almacenada en otro componente del *SGBD* denominado *Diccionario de Datos*.

1.4.2. El lenguaje de definición del almacenamiento de los datos

En la mayoría de los *SGBD* el mismo lenguaje *DDL* permite la definición de los datos en el nivel de representación físico, si bien en otros es un subcomponente de éste denominado lenguaje de definición del almacenamiento de los datos —*Data Storage Definition Language (DSDL)*—. En cualquier caso, haciendo uso del *DDL* o un subcomponente de éste, el *DSDL*, se definen los datos correspondientes al dominio de un problema a los dos niveles de abstracción, y a esta definición de los datos se le denomina *Esquema de la Base de Datos*.

El esquema de la base de datos es una representación de los datos correspondientes al dominio de un problema mediante un lenguaje de definición de datos, el cual está basado en un modelo de datos (véase el Capítulo 2 de esta misma obra). En el esquema estarán definidas:

- Las características del problema a un nivel de descripción lógico o intencional. Esta definición no variará a no ser que cambie el problema.
  - Cada una de las clases de objetos, y sus propiedades, que formen parte del dominio del problema o sistema que se desea tratar con el *SGBD*.
  - Cada una de las relaciones, y sus propiedades, existentes entre estas clases de objetos.
  - Todas aquellas restricciones concernientes tanto a las clases de objetos y sus propiedades como a las relaciones entre ellos.
- Las características del problema desde un punto de vista físico u operacional. Esta descripción puede variar con el tiempo en base a las necesidades o requisitos operacionales, y contemplará, por ejemplo:
  - Las unidades físicas en las cuales los datos van a ser almacenados.
  - Los volúmenes y archivos utilizados.
  - Las características físicas y lógicas de los medios de almacenamiento y métodos de acceso a la información: clusters, bloques, índices, tablas hash, etc.

Además del *DSDL*, el *DDL* cuenta con un sublenguaje encargado del control y seguridad de los datos, el cual se denomina lenguaje de control de datos —*Data Control Language (DCL)*— y permite el control del acceso a la información almacenada en el diccionario de datos (definición de privilegios y tipos de acceso), así como el control de la seguridad de los datos.

1.4.3. El lenguaje de manipulación de datos

Otro componente esencial de los *SGBD* es el lenguaje de manipulación de los datos —*Data Manipulation Language (DML)*—. El *DML* es un lenguaje artificial mediante el cual se realizan dos funciones bien diferentes en la gestión de los datos:

1. La definición del nivel externo o de usuario de los datos.
2. La manipulación de los datos; es decir, la inserción, borrado, modificación y recuperación de los datos almacenados en la base de datos.

Al igual que el *DDL*, el *DML* está basado en un modelo de datos y, por tanto, los *SGBD* basados en distintos modelos de datos tienen diferente *DML*. Se trata también de un lenguaje basado en una gramática completa, sencilla y, generalmente, fácil de entender por usuarios no expertos.

Dependiendo del modelo de datos en el cual se soportan y, por supuesto, del *SGBD*, existen dos tipos de *DML*:

Procedimentales: los cuales requieren que en las sentencias del lenguaje se especifique qué datos se van a manipular, qué se desea obtener y qué acciones/operaciones deben realizarse para ello.

No Procedimentales: los cuales sólo requieren que en las sentencias del lenguaje se especifique qué datos se van a manipular y qué se desea obtener, siendo el propio *DML* el encargado de determinar los procedimientos más efectivos para ello.

[4] Sólo trataremos estas dos categorías de componentes de los *SGBD*, pues si bien el hardware es de vital importancia en estos sistemas y de hecho existen ordenadores especializados para esta tecnología, su estudio no es el objetivo de esta obra.

Naturalmente, estos últimos son mucho más fáciles de entender y manejar por usuarios no expertos, si bien cuando los requisitos en cuanto al tiempo de respuesta del *SGBD* en los procesos de manipulación de los datos son importantes, será necesaria la modificación del código que generan estos lenguajes para asegurar un desempeño adecuado del sistema.

Como se ha comentado anteriormente, el *DML* tiene también la función de describir la visión externa de los datos. Efectivamente, mediante el *DML* se definen las *vistas* o visiones parciales que los usuarios tienen del esquema de la base de datos definido mediante el *DDL*. Estas vistas de los datos son denominadas *Subesquemas* (véase el Capítulo 2 de esta misma obra) y pueden realizarse de varias formas:

- Haciendo uso única y exclusivamente del *DML*. Así, con sentencias propias de este lenguaje se definen distintas visiones parciales (orientadas al usuario final) del esquema de la base de datos. Estas visiones parciales (vistas) son, generalmente, almacenadas en el diccionario de datos.
- Haciendo uso de un lenguaje huésped (host) como *C*, *COBOL*, *FORTRAN*, etc., mediante el cual se realizan los programas de aplicación que permiten al usuario manipular los datos de la base de datos. En el código fuente de estos programas están presentes sentencias del *DML*, que son las encargadas de estos procesos, mientras que las sentencias realizadas en el lenguaje huésped tienen como objetivo el control del flujo de la información y la interfaz de usuario.

  En este caso, los programas fuentes deben de ser precompilados, antes de ser compilados, con el compilador correspondiente, para generar el código máquina, y convertir las sentencias *DML* inmersas en el código fuente a un código entendible por el compilador del lenguaje huésped.

1.4.4. El diccionario de datos

El diccionario de datos es uno o un conjunto de archivos que contienen información acerca de los datos que pueden ser almacenados en la base de datos. Se trata de una *metabase de datos*; es decir, una base de datos (intencional) que contiene información sobre otra base de datos (extensional).

En el diccionario de datos se almacenan todas las definiciones realizadas por el *DDL* sobre el problema que va a ser tratado por el *SGBD* y, algunas (las que se deseen) de las realizadas por el *DML*. Así, en el diccionario de datos se encuentra almacenado:

- El esquema lógico de la base de datos.
- El esquema físico de la base de datos.
- Los subesquemas de la base de datos.

Es decir, la representación de los datos a los tres niveles de abstracción. Pero además, en el diccionario de datos se encuentra mucha más información almacenada; información correspondiente con:

- Las restricciones de privacidad y acceso a los datos almacenados en la base de datos. Estas restricciones han sido definidas haciendo uso del *DDL* y/o su sublenguaje, el *DCL*.
- Las reglas, normas o restricciones referentes a la seguridad de los datos.
- Otra serie de información que permite garantizar la integridad de los datos almacenados en la base de datos.

Si los datos se definen a tres niveles de abstracción, es necesario que en los procedimientos de acceso a estos datos se haga referencia en algún momento a las distintas representaciones de un mismo dato (el proceso de ligadura o vinculación). Por tanto, en el diccionario de datos, además de almacenarse la representación de los datos al nivel externo, lógico y físico, se almacena un conjunto de reglas que permite vincular los mismos datos desde un nivel de abstracción y representación a otro. A este conjunto de reglas se le denomina *Mapa de reglas (Mapping rules)* y consiste en un conjunto de parámetros que definen qué procedimientos deben realizarse para vincular o transformar un mismo dato desde un nivel de representación a otro. A este conjunto de procedimientos o funciones que realizan la transformación de los datos entre los diferentes niveles de representación se le denomina *Mapa de datos (Data mapping)*.

Como existen tres niveles de representación de los datos, existirán:

- Una serie de reglas para definir la correspondencia entre las representaciones física y canónica de los datos.
- Una serie de reglas para definir la correspondencia entre la representación canónica de los datos y cada una de las representaciones externas de los mismos.

1.4.5. El gestor de la base de datos

El gestor de la base de datos, a veces denominado *monitor*, es un componente software encargado de garantizar el correcto, seguro, íntegro y eficiente acceso y almacenamiento de los datos. Este componente es el encargado de proporcionar una interfaz entre los datos almacenados y los programas de aplicación que los manejan.

El que este componente realice sus funciones asignadas correctamente dependerá de muchos factores, entre los que se pueden citar: el volumen de la base de datos, las estructuras físicas definidas para el almacenamiento de los mismos, los procedimientos desarrollados para la manipulación de los datos, las características del hardware y la calidad del propio gestor.

Puede verse al gestor de la base de datos como un intérprete entre el usuario (de cualquier tipo) y los datos. Toda operación que se quiera realizar *contra* la base de datos debe ser previamente permitida por el gestor de la misma, el cual, una vez interpretada y validada, o bien realiza la operación devolviendo el resultado de la misma al programa/procedimiento que la solicitó, o bien la rechaza. Así, el gestor de la base de datos es el responsable de:

- Garantizar la privacidad de los datos, permitiendo sólo el acceso a los mismos a los usuarios autorizados.
- Garantizar la seguridad de los datos, realizando los procedimientos necesarios para que los datos puedan ser recuperados tras un fallo que ocasione una pérdida o deterioro temporal de los mismos.
- Garantizar la integridad de los datos, gestionando que los datos que se almacenan en la base de datos satisfagan las restricciones definidas en el esquema de la misma.
- Garantizar el acceso concurrente a la base de datos de forma que varios usuarios puedan acceder al mismo o distinto dato sin que ello ocasione una pérdida de la integridad de la base de datos.
- Interaccionar con el sistema operativo y, en particular, con el gestor de archivos del mismo, de forma que los procedimientos *DML* puedan ser entendidos por el sistema operativo para el correcto almacenamiento y recuperación de la información. Para ello, el gestor de la base de datos cuenta con un subcomponente denominado *procesador de consultas*.

Es uno de los componentes más complejos del *SGBD*, aunque su complejidad dependerá del propio *SGBD* y de las características del sistema operativo y hardware en el cual se implante. Así, en aquellos sistemas en los que existe la restricción de que no puedan acceder varios usuarios simultáneamente a la base de datos, o la seguridad de los datos sea llevada únicamente por el usuario de la base de datos, este componente será más simple.

1.4.6. El administrador de la base de datos

Otro de los componentes de los *SGBD* es el administrador de la base de datos —*Data Base Administrator (DBA)*—. Se trata de un componente humano de suma importancia en el resultado que el uso de las bases de datos va a tener en la resolución de un determinado problema. El *DBA* tiene una serie de responsabilidades en cuanto a la definición, administración, seguridad, privacidad e integridad de la información que es tratada, así como en el desempeño del *SGBD* en el procesamiento de la misma.

Entre las tareas asignadas al *DBA* se encuentran:

- La definición del esquema canónico o lógico de la base de datos. Es decir, la codificación mediante sentencias del *DDL* del conjunto de definiciones que representan las características del problema que va a ser tratado haciendo uso del *SGBD*. En esta definición se incluyen aquellas especificaciones necesarias para que el *SGBD* pueda mantener la integridad de los datos almacenados en la base de datos.
- La definición del esquema físico de la base de datos. Es decir, la especificación de las estructuras de almacenamiento y los métodos de acceso a la información almacenada en los dispositivos físicos de almacenamiento. Esta definición se realiza haciendo uso del *DSDL* (o el propio *DDL*) mediante un conjunto de sentencias que son compiladas y traducidas a una especificación entendible por la máquina y que es almacenada en el diccionario de datos junto con el esquema canónico.
- La definición de los subesquemas o visiones externas o de usuario de la base de datos. Aquellas vistas parciales de la base de datos que son almacenadas en el diccionario de datos son definidas por el *DBA*, el cual es el único que tiene acceso y, por tanto, privilegios suficientes para la gestión de este componente.
- El control de la privacidad de los datos, mediante la concesión de privilegios a usuarios o grupos de éstos para el acceso a la información almacenada en la base de datos. Esta tarea se realiza en base al esquema de la base de datos y en base a las operaciones básicas que pueden realizarse con los datos (consulta, inserción, modificación y borrado), concediéndose privilegios para una o varias de estas acciones a grupos de datos definidos en el esquema de la base de datos.
- Mantenimiento de los esquemas. Así, el *DBA* es el responsable de:
  - Introducir las modificaciones necesarias en el esquema lógico; modificaciones producidas por un cambio en el problema tratado por el *SGBD* o una ampliación del mismo.
  - Introducir las modificaciones necesarias en la representación física de los datos, de forma que esta representación evolucione paralelamente a la extensión de la base de datos y a la introducción de nuevos requisitos funcionales y/o de desempeño.
  - Introducir las modificaciones y nuevas definiciones de los subesquemas o visiones externas o de usuario, aportando una explotación efectiva de la base de datos.
- La especificación de los procedimientos necesarios para el mantenimiento de la seguridad de los datos almacenados en la base de datos. Es decir, cuándo, cómo y de qué forma se deben realizar y definir los procesos que garanticen que los datos puedan ser recuperados aun después de un fallo que dé lugar a una pérdida temporal de los mismos.

1.4.7. Los usuarios de la base de datos

Tradicionalmente se ha considerado a los usuarios de las bases de datos como un componente más de los *SGBD*, posiblemente debido a que estos sistemas fueron de los primeros en considerar a éstos como una parte importante del correcto, adecuado y necesario funcionamiento de los mismos.

Naturalmente, en un sistema de esta complejidad existen muchos tipos de usuarios que acceden e interaccionan con el mismo, así se pueden considerar:

Usuarios terminales: aquellos usuarios que. a través de programas de aplicación, interaccionan con la base de datos. Son usuarios no especializados que tienen la visión del problema que les proporcionan las visiones externas o subesquemas

que utilizan los programas de aplicación a los cuales tienen privilegios de ejecución.

Usuarios técnicos: aquellos que desarrollan los programas de aplicación que van a ser utilizados por los usuarios terminales de la base de datos. Son profesionales informáticos que, haciendo uso de lenguajes de programación, preparan procedimientos que son invocados desde una interfaz orientada al usuario para realizar las operaciones necesarias en la gestión del problema.

Usuarios especializados: aquellos que utilizan el *SGBD* como una herramienta en el desarrollo de otros sistemas más o menos complejos. Estos usuarios necesitan una buena gestión de la información que es procesada por otro sistema comercial o desarrollado por ellos y, por tanto, utilizan al *SGBD* como un submódulo de sus sistemas particulares, interaccionando con él en la medida que le es necesario. Por ejemplo, en el desarrollo de sistemas expertos, el *SGBD* puede ser el encargado de la gestión de la base y metabase de conocimiento del mismo.

Usuarios críticos: cuya denominación, aunque algo *ruda*, consideramos la más acertada. Estos usuarios pueden tener desde mucho, hasta ningún conocimiento técnico de la tecnología de base de datos, y/o del *SGBD* en el cual se soporta la base de datos con la cual interactúan pero, independientemente de ello, requieren de la base de datos información en un formato, detalle y bajo unos requisitos que generalmente no están previstos de antemano (en el proceso de análisis del problema y diseño del esquema) y en un tiempo mínimo.

Se trata de aquellos usuarios gerenciales o pertenecientes al *staff* de las empresas en las cuales se ha instalado la base de datos, los cuales, en base a expectativas de gestión, administración, mercado, marketing o simplemente por interés personal, realizan consultas no previstas sobre la información almacenada en la base de datos.

Como hoy en día la mayoría de los *SGBD* cuenta con un lenguaje de cuarta generación integrado en su producto, las interacciones de estos usuarios contra la base de datos pueden realizarse sin que los usuarios técnicos tengan que estar preparándoles continuamente procedimientos nuevos, y de un corto tiempo de vida, para la realización de las mismas. Sin embargo, al tratarse de consultas complejas, y debido a que el lenguaje de cuarta generación no genera un código objeto efectivo para interaccionar con la base de datos, y al no estar previstas las estructuras físicas para este tipo de interrogantes, el desempeño de estos procesos es bajo, ocasionando la *crítica* generalizada de estos usuarios, los cuales son, generalmente, los que tienen poder decisorio sobre la inversión y contratación en la empresa (de ahí el nombre que los autores han dado a los mismos).

CAPÍTULO 2

REPRESENTACIÓN DE LOS PROBLEMAS DEL MUNDO REAL

La interpretación de los fenómenos que ocurren en la naturaleza es una actividad innata del ser humano. Desde el principio, el hombre ha intentado explicar las causas del comportamiento del entorno en el cual se encuentra. Para la realización de esta actividad el ser humano se ha basado en su capacidad de abstracción; capacidad mediante la cual podemos simplificar el proceso de interpretación, simplificando o reduciendo el número de parámetros y relaciones existentes en el fenómeno natural que se desea interpretar.

En efecto, el proceso de interpretación de un fenómeno consiste en primer lugar en la caracterización del mismo; es decir, en la propuesta de las propiedades o parámetros que caracterizan a ese fenómeno natural y que, por lo tanto, lo hacen diferente de cualquier otro, proponiendo un modelo inicial que intenta representar al mismo. En este modelo, para cada una de estas propiedades se establece la forma en la cual será medida; es decir, el dominio en el cual estas propiedades pueden ser definidas y, por último, las relaciones existentes entre todas las propiedades que definen el fenómeno en estudio.

Al conjunto de las propiedades que caracterizan un fenómeno se le denomina *datos*, y al conjunto de valores que estas propiedades pueden presentar para un determinado fenómeno, junto con el conjunto de las relaciones o dependencias entre las mismas, se le denomina *información*. Así pues, el contenido de la información es el conocimiento que aporta la información que se tiene de un determinado fenómeno a través de los datos obtenidos del mismo, y que es utilizado por el ser humano en la toma de decisiones.

Un modelo es entonces una representación abstracta y holística de un determinado fenómeno natural. Esta representación informa de qué sucesos deben ser medidos y de qué forma para interpretarlo y, por añadidura, obtener el conocimiento acerca del mismo.

Si no nos limitamos a la interpretación de un fenómeno natural, sino que consideramos que se desea interpretar cualquier fenómeno o suceso real o abstracto, el proceso sigue siendo el mismo y, en ambos casos, los datos que proporcionan la información deben representar valores medibles de las propiedades que caracterizan el fenómeno concreto o abstracto objeto de la descripción.

La medición de los valores correspondientes a los datos es realizada por el hombre haciendo uso, de nuevo, del principio de la abstracción. Para ello, se definen unos *tipos o clases abstractas de datos básicos* (entero, real, booleano, etc.) los cuales pueden tomar un conjunto de valores predefinido de antemano. Cualquier dato que se desee medir para un determinado fenómeno será definido en alguno de estos tipos de datos básicos (dominios).

De esta forma, un dato correspondiente a un fenómeno real o abstracto vendrá dado por la agregación de la representación de ese dato y su valor en el mundo real; es decir, el suceso que representa bien sea real o abstracto, y el valor medido de ese dato en función de alguna de las clases de datos básicas predefinidas. Por ejemplo, la declaración *el peso es 67 kilogramos*, está informando de la representación del dato *peso en kilogramos*, que se ha medido basándose en la clase de datos básica *números enteros* con un valor de *67* para un determinado fenómeno.

La información correspondiente a un determinado fenómeno debe ser almacenada usando para ello un método concreto que permita la comunicación de esta información. El hombre ha utilizado y utiliza un gran número de métodos para la comunicación de la información: imágenes, signos, escritura, pintura, así como un gran número de soportes para cada uno de estos métodos de comunicación: lenguajes, papel, madera, etc. En muchos casos es necesario almacenar para cada uno de los datos el valor medido y la representación del dato o significado del mismo (es necesario almacenar el valor de *67*, y que representa el *peso en kilogramos*), de forma que esta información pueda ser interpretada. Pero si junto con los datos son almacenados de forma conjunta sus valores e interpretaciones, los datos están representando estados de un determinado fenómeno.

En efecto, si un fenómeno puede ser descrito mediante un conjunto de datos o propiedades, estos datos toman un determinado valor en un instante dado del fenómeno y, por tanto, cuando se almacenan estos datos, se está almacenando ese instante. Pero la mayoría de los fenómenos no son estáticos (por ejemplo, el peso de una persona varía con el tiempo) y, por tanto, es necesario representar también la evolución o la dinámica del fenómeno en estudio.

Un *Modelo de Datos* es una unidad de abstracción mediante la cual puede describirse un fenómeno real o abstracto. Mediante el uso de un modelo de datos se describen las propiedades que caracterizan el fenómeno y que lo diferencian de otros fenómenos que se puedan o no describir, las relaciones entre estas propiedades, y cómo las propiedades y las relaciones pueden evolucionar con el tiempo. En definitiva, mediante un modelo de datos se describen las características estáticas y dinámicas de un fenómeno. Un modelo de datos es, por tanto, un conjunto de reglas de acuerdo a las cuales puede ser descrito un fenómeno, aunque como veremos más adelante, no proporciona una interpretación completa de cómo puede utilizarse la información descrita por el mismo, sino únicamente a qué operaciones o a qué tratamiento puede ser sometida esta información.

2.1. LOS PROBLEMAS DEL MUNDO REAL

Como se ha comentado anteriormente, mediante el uso de un modelo de datos puede ser representado cualquier fenómeno real o abstracto; es decir, cualquier *problema* sobre el que se desea obtener información para su conocimiento y/o solución. En el mundo real se presentan un gran número de diferentes problemas a solucionar, problemas de la naturaleza, problemas sociales, económicos, de organización, etc. Por ejemplo, el comportamiento orbital de los astros, el comportamiento de la luz, el crecimiento de las plantas, el crecimiento de la población, las mareas, el cambio de moneda, la contabilidad de una empresa, la gestión de personal, el proceso de comunicación, la determinación del color o textura de un objeto, etc.; en definitiva, cualquier hecho, real o no, que ocurre o parece que ocurre en el mundo real.

El primer paso en la representación del conocimiento acerca de un problema del mundo real es la caracterización del mismo. Es decir, la determinación de los límites del problema. En esta fase, se deben determinar qué datos son los que intervienen en el problema y cómo pueden ser medidos. De esta forma se establece la frontera del problema con respecto al resto de los innumerables problemas o fenómenos existentes o considerados.

Como muestra la figura 2.1, el problema que se desea representar puede ser visto como un sistema en el que intervienen una serie de parámetros o propiedades del mismo. Estas propiedades pueden representar a uno o una colección de datos que deben ser medidos en un determinado tipo de datos básico. Cada propiedad puede mantener un número de interdependencias con el resto de las propiedades que forman parte del sistema de trabajo. Al mismo tiempo existirán otros sistemas, otros problemas, que afecten al problema que se desea representar, o bien el sistema representado puede afectar a otros sistemas.

El término *sistema* es ampliamente utilizado en todas las áreas del conocimiento para identificar un conjunto de elementos cuyas propiedades e interdependencias dan lugar al comportamiento de ese conjunto, comportamiento que lo diferencia de otros sistemas. En el mundo real todo puede ser visto como un sistema, desde los sistemas biológicos, sociales, comerciales, etc., habiéndose desarrollado una teoría general para el estudio de los sistemas independientemente de su naturaleza.

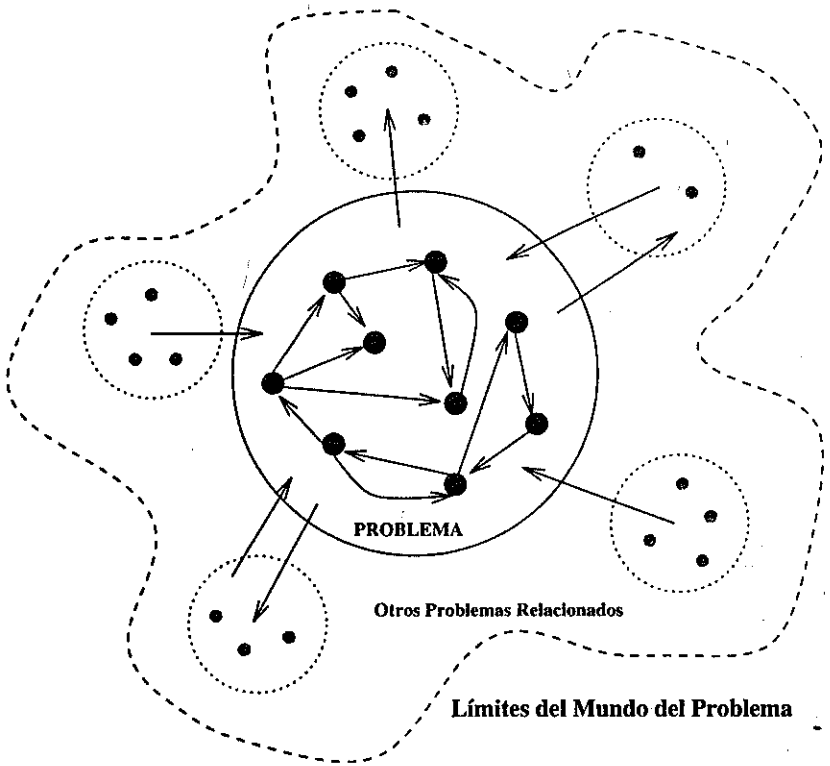


Figura 2.1 Los límites del problema

De forma general, para el estudio de un sistema es necesaria la simplificación del problema que representa el mismo. Esta simplificación comienza por la determinación de la frontera o límite del sistema. En este proceso se determinan aquellas propiedades de interés, dejando otras propiedades que pueden afectar o verse afectadas por el sistema como pertenecientes a otros sistemas o bien al mundo real o a la porción del mundo real que no se está considerando. Este proceso de simplificación es innato al proceso mental del ser humano y está basado en la capacidad de abstracción.

2.1.1. La abstracción

La abstracción es la capacidad mediante la cual una serie de objetos se categorizan en un nuevo objeto mediante una función de pertenencia. Al nuevo objeto se le denomina *clase o tipo de objeto*, y todos los elementos categorizados en esta clase tienen propiedades comunes, las cuales caracterizan la clase. Por ejemplo, el conjunto de los números enteros es una abstracción que nos permite agrupar en esta clase a todos los números naturales positivos, cero o negativos. La abstracción permite

ocultar los detalles, simplificando la descripción de un problema mediante la agrupación de elementos con propiedades comunes que intervienen en el mismo.

La abstracción es utilizada cotidianamente por el ser humano. Por ejemplo, el término vehículo es una clase de objeto mediante la cual representamos a aquellos objetos que tienen la propiedad de desplazarse sobre ruedas. Los elementos de esta clase pueden ser: automóviles, camiones, tractores, carros, etc. Al mismo tiempo, el término automóvil es otra clase que agrupa a los distintos tipos de automóviles, pero todos tienen la propiedad de ser vehículos, así como otro conjunto de propiedades particulares que lo diferencian del resto de los vehículos (por ejemplo, un límite en el peso).

En la definición de los datos, la abstracción es utilizada de dos formas: *generalización* y *agregación*. La generalización es la abstracción por la cual un conjunto de clases de objetos puede ser visto como una nueva clase de objetos más general. Por ejemplo, como muestra la figura 2.2, la clase *Silla*, *Mesa* y la clase *Cama* son generalizadas en una nueva clase denominada *Mueble*. A su vez, estas clases pueden ser generalizaciones de otras clases, como es el caso de la clase *Cama* que es vista como la generalización de las clase *Litera* y *Catre*. Por último, las clases son a su vez agrupaciones (generalizaciones) de objetos simples: todas las sillas, las mesas, las literas y los catres.

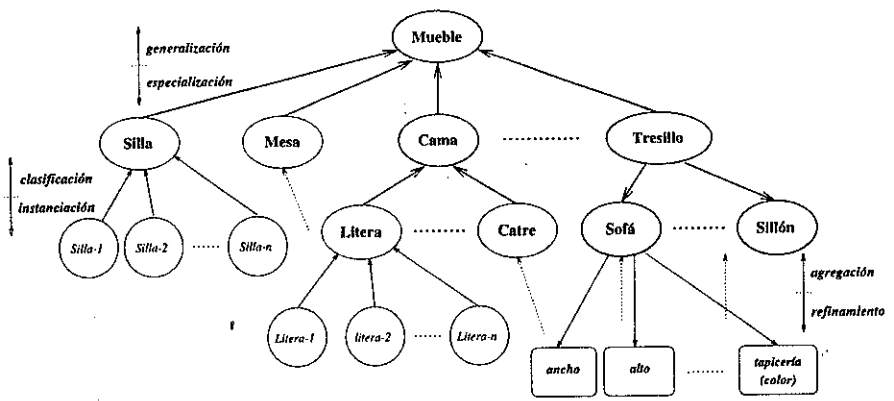


Figura 2.2 Un ejemplo del uso de la abstracción

La generalización de objetos simples en una clase (por ejemplo, cada una de las sillas en la clase *Silla*) es denominada *clasificación*, y se les denomina *especialización* e *instanciación* a los procesos inversos a la generalización y clasificación.

La *agregación*, por otra parte, es la capacidad de considerar un objeto basándose en los elementos que lo constituyen. Por ejemplo, la clase *Tresillo* en la figura 2.2 es vista como la agregación de otras dos clases, la clase *Sofá* y la clase *Sillón* (un *tresillo* está formado por un *sofá* y uno o varios *sillones*, dos generalmente).

De forma similar, un objeto puede verse (y, por tanto, la clase de objetos en la que se generaliza) como la agregación de un conjunto de propiedades que lo caracterizan. Así, en la figura 2.2, se muestra que la clase *Sofá* es vista como la agregación de las propiedades: *alto*, *ancho*, *tapicería*, etc., las cuales caracterizan a esta clase y, por tanto, a todos los elementos de la misma, diferenciándola de cualquier otra clase. El proceso inverso a la agregación se denomina *refinamiento*, mediante el cual se puede representar a aquellos objetos simples o propiedades que caracterizan a una clase de objetos.

La generalización puede asociarse con el concepto *ES_UN*, ya que mediante esta abstracción es posible representar a aquellos objetos o clases de objetos que pertenecen o pueden ser considerados como de un tipo o clase de objetos más general. Por otro lado, la agregación puede asociarse con el concepto *PARTE_DE*, puesto que mediante esta abstracción puede representarse que un objeto o clase de objetos está formado por una serie de objetos o clases constituyentes que lo caracterizan como objeto o clase.

2.1.2. Representación de los problemas del mundo real

La abstracción es una herramienta muy potente mediante la cual pueden ser estudiados los problemas del mundo real gracias a la simplificación que permite introducir en el proceso de representación de los mismos. La representación de un problema puede llevarse a cabo haciendo uso de la abstracción de forma ascendente o descendente en complejidad.

En la forma ascendente, inicialmente deben ser determinados aquellos objetos simples, aquellos datos o propiedades simples que intervienen en el problema o sistema en estudio, y para cada uno de estos datos se determina el tipo de datos básico mediante el cual puede ser medido. A continuación, estos datos son agregados en clases de objetos del sistema. Las clases de objetos estarán formadas por la agregación de un conjunto de propiedades del sistema, cada una de ellas medible en un tipo de datos básico y para los cuales se determinan los límites en el conjunto de valores que pueden tomar para la clase de objeto de la que forman parte. Por ejemplo, la clase *Sofá* puede estar definida como la agregación de un conjunto de propiedades: *ancho*, *alto*, *color de la tapicería*, etc. La propiedad color está definida en un tipo de datos básico, el tipo cadena de caracteres, por lo que cualquier conjunto de caracteres puede ser un valor, en principio, para esta propiedad. Pero en el caso particular de los valores que puede tomar la propiedad color de la tapicería para la clase *Sofá*, se define, por ejemplo, que sólo pueden ser medidos los colores: amarillo, negro y azul; mientras que la propiedad color también puede existir para la clase *Silla*, pero, en este caso, los únicos valores admitidos pueden ser: negro, blanco y madera.

Una vez definidas las clases de objetos que intervienen en el problema y haciendo uso de la abstracción, estas clases pueden ser generalizadas en clases de objetos de mayor categoría y que agrupan a varias clases más simples. Por ejemplo, la clase *Cama* como generalización de las clases *Litera* y *Catre*. Este proceso ascendente se repite hasta que la generalización es completa.

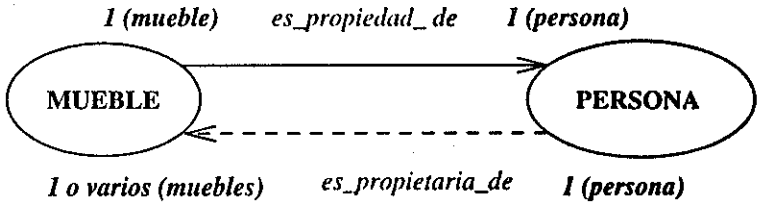


Figura 2.3 Relaciones entre los elementos del problema

Por otro lado, la representación del problema puede realizarse de forma descendente. Inicialmente se identifican las clases de objetos más generales y se procede a un proceso de especificación e instanciación de las mismas hasta alcanzar las propiedades o datos que intervienen en el problema.

Generalmente, es necesario utilizar ambas técnicas en la representación de un problema. Partiendo de un proceso ascendente o descendente se va saltando de uno a otro conforme se va alcanzando más conocimiento respecto del problema a representar, refinándose en cada paso la representación del mismo hasta alcanzar una representación correcta y simplificada del sistema.

Además de las relaciones *ES-UN* y *PARTE_DE* consideradas entre las clases de objetos que intervienen en un problema, en la mayor parte de los problemas existen otro tipo de relaciones o dependencias entre los elementos del mismo. Por ejemplo, consideremos que se desea representar los muebles que las personas tienen en su casa. De forma simplificada, y muy general, se puede deducir que en el problema están presentes dos clases de objetos: la clase *Mueble* y la clase *Persona*. La clase *Mueble*, a su vez, puede refinarse como muestra la figura 2.2 y la clase *Persona* podría refinarse de forma similar. Pero además, entre ambas clases de objetos existe una dependencia o relación, en nuestro caso la relación que identifica qué persona es propietaria de qué mueble, como muestra la figura 2.3.

Como muestra la figura 2.3, en la descripción de un problema es necesario también representar las interdependencias entre los elementos del mismo. Esta representación debe consistir en la información correspondiente al significado de la dependencia y la información correspondiente al número de objetos de cada una de las clases que se ven implicados, así como a la forma, debido a esta dependencia. En nuestro caso, el significado de la dependencia será *la propiedad del mueble por la persona*, y los objetos implicados significarán que un objeto *Persona* puede ser propietaria de uno o más objetos *Mueble*, mientras que un objeto *Mueble* sólo es propiedad de un objeto *Persona*.

2.1.3. Análisis de los problemas

La representación de los problemas es un proceso complejo que requiere una gran experiencia, capacidad y método por parte de la persona encargada de esta labor.

La representación de un sistema es la conclusión de un arduo y complejo proceso de observación del mismo. Un proceso en el cual se determinan las entidades del sistema, sus dependencias y, por tanto, el comportamiento del mismo, así como las interdependencias con otros sistemas.

A este proceso se le denomina *Análisis del Sistema*, e independientemente del problema a representar está basado en la *Teoría General de Sistemas*, la cual se basa en la consideración de todas las partes del sistema para el estudio de un problema; es decir, la consideración holística del problema y de su entorno y, por tanto, de todos los elementos que intervienen en el estudio de algún problema dentro del sistema. Esta teoría surgió con la intención ideal de proponer un método general mediante el cual pudiera ser representado cualquier problema. Es decir, una teoría que permitiera la representación del *saber total*. Esta teoría se basa en considerar que todos los problemas o sistemas están relacionados y que todos tienen un comportamiento básico similar. Si es posible proponer un modelo para representar este comportamiento básico, entonces sería posible el conocimiento de todos los problemas que se le planteen al ser humano.

Naturalmente, el tiempo demostró que los ambiciosos objetivos de estos investigadores no podían ser alcanzados, y la teoría de sistemas se adaptó al estudio particular de cada uno de los sistemas, adecuándose a las particularidades de los mismos utilizando un método general de trabajo[5].

Según la categoría del sistema que se desee estudiar se han desarrollado a lo largo de los años una serie de teorías o principios mediante los cuales este estudio debe ser realizado. Así, los problemas físicos, químicos, biológicos o de la empresa requieren para su estudio el uso de una teoría, que aunque basada en los principios generales de la teoría general de sistemas, se ha adaptado a la particularidad de cada problema o subproblema.

De forma general, la representación de un problema requiere el seguimiento de los siguientes pasos:

1. La *definición del problema*, mediante una descripción simple y concreta del problema que se desea estudiar y de cuál es la función u objetivo que el sistema intenta alcanzar. Si se trata de un sistema que puede ser observado, esta definición estará basada en la observación del comportamiento del mismo utilizando para ello un nivel de abstracción elevado. En esta fase interesa describir cómo el sistema se comporta a grandes rasgos; es decir, cómo el sistema es observado desde los límites del mismo. No interesa describir el comportamiento interno del sistema, sino el comportamiento externo. Si el sistema no puede ser observado, en este paso se describirá cómo se comporta o se deberá comportar (si no existe) como un todo.

[5] La idea de proponer un modelo general de representación de cualquier problema o sistema supone el pretender alcanzar la raíz última del conocimiento; es decir, el conocimiento absoluto, y éste es un objetivo *bastante ambicioso*.

2. La *definición de la arquitectura del problema*, mediante una descripción de las *partes* importantes del sistema. Basándonos en la abstracción, un sistema puede ser visto como un conjunto de otros sistemas o problemas de menor entidad que se interrelacionan entre sí. En este paso se describirán estos subsistemas en el mismo nivel de abstracción que en el paso anterior, prestando atención a cómo afecta cada uno de los subsistemas o se ve afectado por los otros.

   Se puede observar que, en estos dos pasos, se están definiendo los límites del sistema y de cada uno de los subsistemas que lo componen y, por tanto, se está diferenciando el problema, y sus correspondientes subproblemas, de cualquier otro conjunto de problemas que se puedan estudiar.

3. La *definición de la estructura del problema*, mediante la descripción de los elementos del sistema. En esta fase se determinan qué objetos, entidades, datos o variables son las que forman parte del problema en estudio. Estos objetos pueden formar parte del problema como un todo, o de alguno o varios de los subproblemas que se han determinado en el paso anterior. Para cada uno de estos objetos se determina:

   a) La *definición del objeto*, mediante una descripción de la función que desempeña el objeto dentro del problema en estudio. Esta descripción aporta una interpretación del objeto en el mundo real, independientemente de la existencia del sistema.

   b) La *medida del objeto*, mediante una descripción de los valores que pueden ser medidos o puede tomar el objeto para el problema en estudio. Para ello, a cada uno de los objetos se le asignará, mediante una función de pertenencia, un tipo de datos básico en el cual puede ser medido. Este paso se realiza mediante el uso de la *generalización*, de forma que los objetos del problema son generalizados mediante una relación *ES_UN* a una clase básica de un conjunto predefinido de antemano y que es común para todos los sistemas en estudio.

   c) Las *relaciones entre los objetos*, mediante una descripción de las interdependencias entre los objetos que intervienen en el problema. En este paso y, de nuevo haciendo uso de la abstracción, se agregarán objetos particulares en objetos más generales cuyo comportamiento está definido y limitado por el conjunto de estos objetos particulares, y se agruparán objetos en otros más generales, los cuales representan un comportamiento común a un conjunto de objetos.

      También, en este paso, se describirá cualquier otro tipo de relaciones que están presentes en el problema entre los objetos constituyentes del mismo. Todas las relaciones serán identificadas sobre la base de la función u objetivo que tienen las mismas para que el sistema alcance su objetivo global, y se determinará cómo participan los objetos que intervienen en estas relaciones y en qué número.

d) Por último, se definen las *restricciones* inherentes a los objetos para el problema en estudio. En este paso se describe, ya a un nivel muy bajo de abstracción, qué valores pueden ser medidos para cada uno de los objetos basándose en sus propiedades y a las relaciones que mantienen con el resto de los objetos del sistema.

4. En los pasos anteriores se ha realizado una definición del problema como una plantilla, un marco, formado por un conjunto de cuadrículas (los subproblemas), las cuales a su vez pueden estar formadas por otras cuadrículas más pequeñas, en las cuales están presentes una serie de zonas, marcas, ranuras o *slots* en los que aparecen datos que pueden tomar un conjunto de valores a veces ilimitado. Es decir, se ha definido la *estática* del problema, una descripción tal que nos permite representar el estado del sistema independientemente del instante en que se observe.

   El siguiente paso consiste en la *definición de la dinámica del problema*; es decir, la descripción de la evolución que el problema va a tener o tiene con el tiempo. Si se trata de un sistema existente, en este paso se realiza una interpretación de la evolución del mismo y se propone un modelo mediante el cual pueden preverse futuros estados del sistema. Si el sistema no existe, en este paso se propone, también mediante un modelo, qué operaciones o qué acciones pueden realizarse o ser realizadas sobre cada uno de los objetos del sistema; entendiendo por objetos a cualquier elemento simple o complejo (un dato, propiedad, objeto simple o clase de objeto a cualquier nivel de abstracción) que forme parte del problema.

5. Una vez definidas las características estáticas y dinámicas del sistema, el siguiente paso consiste en el *estudio del comportamiento* del modelo propuesto. Si el sistema existe, se analizará la adecuación del modelo al comportamiento real del sistema, y si el sistema no existe, se estudiará la adecuación del modelo a los objetivos deseados por el sistema. Cualquier desviación entre el comportamiento real y esperado dará lugar a una revisión en los pasos anteriores y, por tanto, a la modificación del modelo.

Es lógico pensar que no existe una única representación de un problema. Distintas personas pueden ver un mismo problema de distinta forma, y un mismo problema puede, a su vez, verse desde distintos puntos de vista, dependiendo de los aspectos del problema que se deseen representar y de la forma en que se representen. Además, hay que tener en cuenta las dependencias existentes entre el problema en estudio y otros problemas (fuera de los límites del sistema), ya que cualquier alteración en el entorno del problema puede alterar el comportamiento del sistema en estudio. Por ello, se pueden presentar varias alternativas en la representación de un problema, alternativas que deben ser estudiadas completamente, para al final desestimarse todas menos una basándose en un gran número de factores como: disponibilidad de medios, capacidad para su representación, coste y esfuerzo necesario, etc.

2.2. LOS MODELOS DE DATOS

El análisis de un problema tiene como objetivo el proponer un modelo del comportamiento y características del mismo. Este modelo está basado en una representación de los elementos del problema, de las relaciones entre los mismos y del comportamiento de estos elementos y sus relaciones en el tiempo. Los *modelos de datos* son abstracciones mediante las cuales puede realizarse una representación de los problemas en estudio. Están basados en el uso de la abstracción, permitiendo una descripción del sistema y sus elementos como un todo sin que sea necesario describir cada uno de los elementos particulares del sistema ni cada uno de los estados de los elementos y del sistema a lo largo del tiempo.

Mediante un modelo de datos el sistema es descrito como una clase de objeto que interacciona con otras clases de objetos (otros sistemas). El sistema está a su vez formado por otras clases, las cuales pueden a su vez ser clasificadas o refinadas. A su vez, mediante un modelo de datos se especifican las operaciones o acciones que los objetos pueden llevar a cabo. Así, un modelo de datos es una abstracción mediante la cual pueden ser descritas las características estáticas y dinámicas de un sistema.

Pero ningún modelo de datos puede describir al mismo tiempo la naturaleza estática y dinámica de un sistema. Por ello, un modelo de datos está a su vez formado por dos submodelos:

1. Un submodelo encargado de definir las propiedades estáticas del sistema; es decir, aquellas características del sistema que son invariantes con el tiempo y que identifican al sistema y a cada uno de los objetos constituyentes.
2. Un submodelo encargado de describir las propiedades dinámicas del sistema; es decir, qué acciones toma o puede tomar el sistema y cada uno de los objetos constituyentes, acciones que dan lugar a que el sistema evolucione y pase de un estado E_1 a un estado E_2.

Los modelos son utilizados en todas las áreas del saber para la descripción y representación de los problemas en estudio, y en las áreas relacionadas con el uso del ordenador no va a ser menos. Han sido propuestos numerosos modelos para la representación y solución de diferentes tipos de problemas haciendo uso del ordenador. Todos ellos están basados en el uso de la abstracción para la definición de *tipos* o *clases*, puesto que el uso del ordenador requiere la tipificación de los datos que son tratados.

Cada modelo de datos da soporte a una teoría para la representación y tratamiento de los problemas, o lo que es lo mismo, una teoría se soporta en un modelo. De forma general todos los modelos presentan:

- Un conjunto de reglas mediante las cuales puede ser representado gráficamente el problema. Este conjunto de reglas está soportado por un conjunto de símbolos por los que pueden ser representados cada uno de los objetos del sistema a los

diferentes niveles de abstracción y cada una de las relaciones o dependencias, en sus diferentes tipos, entre los objetos del sistema. Mediante el uso de estas reglas y con la simbología correspondiente son representadas también las restricciones existentes en el problema para los objetos y sus relaciones.

- Un pseudolenguaje, formado por un conjunto reducido de morfemas y una sintaxis perfectamente establecida, mediante el cual pueden describirse las propiedades estáticas y dinámicas del sistema.
- Un conjunto de restricciones que limitan el ámbito en el que el modelo puede ser utilizado. El conjunto de restricciones del modelo marca los límites de los sistemas a representar y, por tanto, de las características de los sistemas que pueden ser representados por un modelo.

Los modelos de datos que soportan las teorías para el tratamiento de la información mediante el uso del ordenador, los modelos de interés en nuestro contexto, permiten la representación del problema a tres niveles de abstracción diferentes. Tres visiones de un mismo problema, las cuales lo describen, de forma idealmente independiente, para su representación y tratamiento. Estas tres visiones son:

Nivel conceptual: a este nivel son representados los tipos o clases de objetos, y sus relaciones desde un punto de vista estructural. En el nivel conceptual se representa un modelo del sistema en el que se describen cada uno de los tipos de objetos o elementos del mismo. Para cada uno de estos tipos de objetos se describen sus propiedades y el dominio o tipo de datos básico en el cual pueden ser medidas, así como las restricciones o límites de los valores que pueden presentarse para cada una de estas propiedades. Además, son descritas las relaciones entre los tipos de objetos, relaciones jerárquicas o no, apoyándose para ello en los principios de la abstracción.

A este nivel se representa el problema tal y como es; es decir, se representa el mundo real del problema tal y como se percibe, sin tener en cuenta cómo este problema puede ser representado para que sea entendido por los programas de ordenador.

Un modelo conceptual de un problema o la visión conceptual de éste, es independiente de los procedimientos manuales o automatizados que se utilicen o se vayan a utilizar para el mantenimiento y tratamiento de la información correspondiente al problema. La visión conceptual sólo es dependiente de:

- Las características del problema o sistema que se desea representar.
- El detalle de la representación, el cual sí depende de la *parte* o *partes* o globalidad del problema que se desea representar para su posterior tratamiento.

Pero es independiente de las herramientas y mecanismos que se vayan a utilizar para esa representación y tratamiento.

Nivel lógico: en este nivel se representa el problema bajo las limitaciones impuestas por la representación y el tratamiento de la información que se vaya a realizar. Es decir, en este punto se introducen en la representación las limitaciones o restricciones que imponen los mecanismos y soportes que se van a utilizar para la representación y tratamiento de la información del problema.

Está claro que el tratamiento manual de la información predispone unas restricciones, en cuanto al soporte y forma en que ésta debe ser almacenada para su posterior tratamiento, muy diferentes al tratamiento automatizado y mediante un ordenador de la información correspondiente al mismo problema.

Igualmente, es fácil advertir que en función del hardware y software utilizado para el almacenamiento y tratamiento de esta información, la representación del problema se verá afectada considerablemente. Por ejemplo, el uso de un sistema de gestión de bases de datos relacionales supone la aplicación de una serie de reglas para la representación de un determinado problema del mundo real muy diferentes al uso de cualquier otro sistema de gestión de bases de datos (red, jerárquico, orientado a objetos, etc.) o estructuras clásicas de almacenamiento.

Mientras en el nivel conceptual el problema se representa tal y como es captado desde el mundo real, en el nivel lógico esta representación es filtrada o alterada para que se adapte a las limitaciones existentes para llevar a cabo este proceso. Por ejemplo, en el nivel conceptual describiremos un árbol tal y como lo apreciamos en el mundo real, mientras que en el nivel lógico describiremos el mismo árbol en base a cómo los datos que lo representan pueden ser almacenados para su posterior tratamiento (una imagen, una descripción textual, tabular, etc.).

Nivel físico: en este nivel, el principio de representación del problema está guiado tanto por el soporte utilizado para su representación como por los métodos o mecanismos que se van a utilizar para el tratamiento de la información correspondiente a éste.

En este nivel, el problema se representa en la forma en que es visto por el sistema de representación y tratamiento utilizado, y no cómo existe o es visto desde el mundo real. Por ejemplo, si se quiere acceder posteriormente a la información correspondiente a los árboles basándose en la ubicación de los mismos, se deben establecer mecanismos y, por tanto, representar la información de los árboles de forma muy diferente a cuando se desee acceder a la misma información basándose en la categoría o clase de árbol, de forma que el criterio de ubicación o clase facilite el acceso y posterior tratamiento de la información correspondiente a los árboles.

La visión o representación física de la información correspondiente a un problema será determinante en el desempeño del tratamiento de la misma. En los problemas tratados por programas de ordenador, la visión física determina las estructuras utilizadas para el almacenamiento de la información, al igual que en el tratamiento manual, la visión física determina los documentos, sus formatos, compaginación, ubicación y archivos utilizados para el almacenamiento de la información.

Como un modelo de datos no puede representar al mismo tiempo la estructura y comportamiento de un sistema, mediante el uso de cada submodelo se debe representar, a estos tres niveles de abstracción, la sustancialidad y el comportamiento que se va a realizar sobre el problema en estudio. Cada submodelo será utilizado para representar cada una de las características del sistema, las estáticas y las dinámicas, a estos tres niveles de representación.

Así, cuando se representa el comportamiento del sistema se describe éste a tres niveles de abstracción: *conceptual, organizativo* y *procedimental*. Mediante los cuales son descritas cada una de las acciones que los elementos del sistema realizan o pueden realizar, así como cada una de las acciones por las que estos elementos se ven afectados o afectan a otros sistemas (ver tabla 2.1).

NIVEL DE DESCRIPCIÓN	ESTRUCTURA	COMPORTAMIENTO
Modelo Conceptual	Descripción de los objetos del mundo real, de sus atributos o propiedades y de las relaciones entre los objetos	Descripción del comportamiento de los objetos: las acciones, operaciones y procesos que estos objetos realizan sobre otros objetos, así como las que son realizadas sobre los objetos del sistema
Modelo Lógico **Modelo Organizativo**	Descripción de los objetos, así como las relaciones existentes entre los objetos lógicos, identificando los atributos por los cuales estos objetos pueden ser identificados	Descripción de las tareas que se deben realizar para representar el comportamiento de los objetos. Estas tareas se agruparán en fases y procedimientos
Modelo Físico **Modelo Procedimental**	Descripción de los objetos físicos. La estructura y relaciones de los objetos son representadas de forma adecuada para su posterior almacenamiento, recuperación y tratamiento	Descripción de las acciones elementales que se deben realizar para representar el comportamiento de los objetos. Estas acciones son representadas bajo las limitaciones del lenguaje que se vaya a utilizar para su implementación en lenguajes de ordenador

Tabla 2.1 Niveles de abstracción en la representación de problemas

Ahora bien, la representación de un sistema en sus dos aspectos: estático y dinámico, y a estos tres niveles de abstracción, no es una tarea fácil debido a que no se trata ni de niveles o visiones independientes ni de aspectos independientes de un problema.

Está claro que el comportamiento de un sistema está mediatizado por su estructura, o dicho de otra forma, la estructura de un sistema determina y limita el

comportamiento del mismo. Por ejemplo, una mesa no puede tener acciones relacionadas o que impliquen su desplazamiento por el aire a elevadas velocidades, mientras que un avión sí. La diferencia en el comportamiento entre una mesa y un avión es debida, entre otras causas, a la diferente estructura de estos sistemas.

Al mismo tiempo, la visión lógica de la estructura de un problema va a depender de qué comportamiento de ese problema se desee estudiar; es decir, de la visión organizativa del problema. Por ejemplo, si se desea seguir la evolución del comportamiento de un coche con el tiempo, se almacenará la fecha de adquisición del coche, mientras que si no se desea conocer este comportamiento, esta información no deberá ser representada, pues es trivial, en la visión lógica e incluso conceptual y, por tanto, física de la estructura del sistema coche.

Por tanto, el proceso de representación de los problemas del mundo real, en sus dos aspectos, es un problema complejo que debe ser llevado a cabo siguiendo una serie de normas o reglas básicas:

1. El proceso de representación del problema debe ser realizado siguiendo una *Metodología* o *Plan* preestablecido de antemano, el cual estará soportado por uno o varios modelos. Una metodología es un conjunto de acciones y reglas que determinan las actuaciones a seguir y en qué orden, para realizar la representación de un problema y la construcción de los programas para el tratamiento de la información correspondiente al mismo. El área de las ciencias encargada de este estudio se denomina *Ingeniería del Software*, y está basada en la Teoría General de Sistemas.

   La ingeniería del software propone el uso de un método de ingeniería (el uso del método científico adaptado al uso del ordenador) para la representación de problemas en cuyo estudio, tratamiento y solución se haga uso del ordenador. Al ser un área relativamente reciente, hoy en día, se acepta el uso de una serie de metodologías diferentes y, por tanto, diferentes modelos en los que se soportan, para llevar a cabo esta tarea, cada una de ellas orientada a un tipo o grupo de problemas diferentes.

2. Inicialmente, debe realizarse un documento inicial en el que se describa el sistema (si éste existe) o el que se desee representar (si éste no existe), su comportamiento global, los objetivos del mismo, las relaciones con otros sistemas del entorno, etc. Esta descripción debe realizarse en un lenguaje natural y a un nivel de abstracción muy general, sin entrar en detalles del sistema en sí. Además, se deberán describir los objetivos que se persiguen, tanto en cuanto al sistema que se desea describir y/o construir como al alcance del mismo. Esta descripción deberá ser clara, exacta y concisa para que a partir de esta descripción pueda realizarse al final del proceso la revisión y validación del mismo.

3. La descripción del sistema debe realizarse a los tres niveles de abstracción anteriormente descritos, tanto para la estructura como para el comportamiento

del mismo. Estos procesos se realizarán haciendo uso de las técnicas de representación y estándares que proponga la metodología que se haya adoptado. Si es necesario se introducirán técnicas propuestas por otras metodologías o propias del diseñador con el único fin de clarificar la representación del sistema.

4. El proceso de representación en cada uno de los niveles de abstracción se realizará en etapas sucesivas, representando en cada nivel el problema desde visiones globales hacia visiones particulares de los elementos del mismo. De esta forma, en cada etapa se realizará un refinamiento de la representación realizada en la etapa previa. Este procedimiento por etapas implica el abordar las características del problema poco a poco, lo que facilita el entendimiento del mismo, de forma que no se atacan todos los problemas del sistema y de su representación al mismo tiempo, sino conforme se va alcanzando más conocimiento acerca de los mismos.
5. Para cada etapa o fase de representación realizada se deben generar una serie de documentos que permitan el entendimiento del sistema tanto a la persona que ha realizado la representación como a otras personas que puedan requerir esta información. La documentación es una actividad de suma importancia en el estudio y solución de problemas que requiere una gran cantidad de tiempo y esfuerzo para su desarrollo adecuado.
6. Los documentos y, por tanto, la representación del sistema realizada debe ser revisada conforme al documento inicial en el que se estipularon los objetivos del trabajo a realizar. Este proceso de revisión debe realizarse guiado por los criterios de validación y verificación que fueron descritos en el documento inicial.

2.2.1. Modelos de datos y sistemas de gestión de bases de datos

Se han descrito, en el capítulo anterior, los *SGBD* y los componentes que lo forman y que permiten, haciendo uso de ellos, la definición de los datos correspondientes al problema en estudio a los tres niveles de abstracción descritos: nivel externo, lógico e interno.

Cada *SGBD* está basado en el uso de un modelo de datos y en el uso de su teoría y, por tanto, heurística, para la descripción y manipulación de los datos. Se tienen, entonces, *SGBD* jerárquicos, en red, relacionales, orientados a objetos, etc., los cuales están basados y se apoyan en los modelos de datos correspondientes: jerárquico, plex, relacional (algebraicos o predicativos), orientados a objeto, etc.

Es el *DDL* del *SGBD* el encargado de la definición de los datos que va a considerar la base de datos correspondientes al problema que se desea tratar. Este componente del *SGBD*, haciendo uso del modelo de datos correspondiente, y como se describió en el capítulo anterior, permite describir los datos a los tres niveles de abstracción: externo, lógico y físico.

Por otro lado, el *DML* del *SGBD* es el encargado de la manipulación de los datos representados y almacenados en la base de datos. El *DML* de un *SGBD* está basado en el mismo modelo de datos, puesto que el tratamiento de la información es dependiente de la estructura de ésta; es decir, del modelo utilizado para la descripción de los datos correspondientes a un problema. Si bien, hoy en día, se está utilizando el *SQL*, el lenguaje de consulta de los *SGBD* relacionales algebraicos, en *SGBD* basados en otros modelos de datos (por ejemplo, orientados a objetos) debido a la estandarización del mismo y su facilidad de uso.

Pero ésta no es una regla general, cada *SGBD* tiene su propio *DML*, con su propio lenguaje de consulta cuya filosofía y sintaxis está basada en el modelo de datos en el cual se asienta el *SGBD*.

Ahora bien, aunque cada *SGBD* tiene su propio *DDL* y *DML* encargado de la definición y manipulación de los datos, no todos los *SGBD* basados en un mismo modelo tienen un *DDL* y/o *DML* basados en la misma sintaxis, aunque sí filosofía, para desarrollar sus funciones. Es el constructor del *SGBD* el que define la estructura de estos componentes basándose en estándares cuando éstos están definidos.

El lector puede observar este hecho en cualquiera de los *SGBD* existentes en el mercado. No todos los *DDL* de los *SGBD* relacionales utilizan la misma sintaxis para la definición de los datos a los diferentes niveles de abstracción, aunque en este caso sí utilizan todos el *SQL* como lenguaje de consulta[6].

Independientemente del *SGBD* que vaya a ser utilizado para la resolución de un problema, el primer paso será siempre el análisis y entendimiento del mismo, de forma que este problema pueda ser representado haciendo uso del *DDL* correspondiente.

Una práctica general es la representación de este problema, inicialmente, de forma independiente al *SGBD*. Recuérdese que el nivel lógico de los datos es dependiente del modelo de datos utilizado por el *SGBD*, puesto que cada *SGBD* está basado en un modelo de datos. Si se representa un problema, en la fase de su análisis, en este nivel lógico, se está restringiendo el uso de los *SGBD* que pueden ser utilizados para su tratamiento. Por ello, en la fase de análisis de los problemas que van a ser tratados haciendo uso de un *SGBD* es conveniente su representación al nivel conceptual. Un nivel de abstracción independiente del *SGBD*, pues sólo depende de las características del problema.

Una vez representado el problema y validada esta representación con los requisitos impuestos en el mismo, la representación conceptual se *traducirá* al nivel lógico de cualquiera de los *SGBD* con los cuales se desea tratar los datos correspondientes al problema.

---

[6] Esta diferencia se aprecia sobre todo en los *SGBD* orientados a objetos, en los que la definición y la manipulación de los datos son muy diferentes dependiendo del *SGBD* de que se trate.

La representación conceptual más utilizada en el análisis de los problemas es aquella que está basada en el modelo de datos denominado *Entidad-Interrelación*, el cual se describirá en la siguiente sección.

2.3. EL MODELO ENTIDAD-INTERRELACIÓN

El *Modelo Entidad-Interrelación* fue propuesto por *Peter Chen* a mediados de los años setenta para la representación conceptual de los problemas y como un medio para representar la visión de un sistema de forma global. Se trata de un modelo muy extendido que ha experimentado a lo largo de los años una serie de ampliaciones, lo que ha originado un medio muy potente para la representación de los datos correspondientes a un problema. Las características actuales de este modelo permiten la representación de cualquier tipo de sistema y a cualquier nivel de abstracción o refinamiento, lo cual da lugar a que se aplique tanto a la representación de problemas que vayan a ser tratados mediante un sistema computerizado como manual; siendo utilizado en muchas ocasiones simplemente para la representación estructural de las organizaciones y, por tanto, para el estudio de los sistemas gerenciales.

El modelo Entidad-Interrelación, el cual lo denotaremos como *E-R*, está soportado en la representación de los datos haciendo uso de grafos y de tablas. Mediante un conjunto de símbolos, y haciendo uso de un conjunto reducido de reglas, son representados los elementos que forman parte del sistema y las relaciones existentes entre ellos, siendo estos elementos descritos mediante un pseudolenguaje basado en una gramática sencilla. El modelo *E-R* propone el uso de tablas bidimensionales para la representación particular de cada uno y, por tanto, de los conjuntos de elementos particulares y sus relaciones existentes en el sistema.

Antes de empezar a describir el modelo *E-R*, es necesario introducir una serie de conceptos básicos que son utilizados por el mismo. Así, se define:

Conjunto: se denomina *conjunto*, en este contexto, al igual que en la *Teoría Clásica de Conjuntos*, a la agregación de una serie de objetos elementales mediante una función de pertenencia. La función de pertenencia caracteriza a los elementos como parte del conjunto, no siendo importante el orden de los elementos dentro del conjunto, ni duplicación de los mismos. Así, el conjunto {1, 2, 3} es igual al conjunto {2, 2, 1, 3, 1}.

Relación: se denomina *relación* a un conjunto que representa una correspondencia entre dos o más conjuntos. Una relación es, por tanto, un nuevo conjunto en el que cada elemento está formado por la agregación de los elementos de los conjuntos individuales que intervienen en la relación. Así, una relación binaria define un conjunto de pares ordenados $<c_1, c_2>$, que se forman en base a un criterio de correspondencia. El orden de la relación es importante, por lo que el par $<c_1, c_2>$ no tiene por qué ser igual al par $<c_2, c_1>$. Las relaciones pueden ser binarias, ternarias o n-arias, y pueden ser definidas como el *producto cartesiano* de los conjuntos que intervienen en la relación ($R \subseteq C_1 \times C_2 \times \ldots C_n$).

Intención y extensión: tanto los conjuntos como las relaciones pueden ser descritos en términos de *intención* y *extensión*. La intención o comprensión representa, en términos de abstracción, la clasificación de una serie de elementos individuales en un tipo o clase de objeto al que se ha denominado conjunto o relación. La extensión representa, también en términos de abstracción, la instanciación de un tipo o clase de objetos (el conjunto o la relación). Así, la intención es la descripción del tipo o clase de objeto (conjunto o relación), y la extensión es la descripción de los elementos individuales o instancias de objetos (los elementos del conjunto).

Consideremos, por ejemplo, los conjuntos C_1 y C_2, donde: $C_1 = \{c \mid c = letra\}$ y $C_2 = \{c \mid c = 0,1\}$, y la correspondencia entre los mismos definida por la relación $R \subseteq C_1 \times C_2$. Esta definición describe las propiedades intencionales de la relación R, que consiste en un conjunto cuyos elementos están formados por la agregación de una letra y un cero o un uno. Así, el conjunto $C_3 = \{<a, 1>, <b, 0>, <c, 1>\}$, representa una extensión de la relación R.

Puede advertirse que la intención es invariante con el tiempo, pues describe la estructura de un conjunto, mientras que la extensión varía con el tiempo, pues depende de los elementos del conjunto en un momento dado.

Dominio: se denomina *dominio* a los conjuntos homogéneos; es decir, a aquellos conjuntos cuyos elementos son homogéneos. Un dominio, en términos de abstracción, es una especialización de un conjunto. Por ejemplo, el conjunto de los números comprendidos entre el 10 y el 99 (los números de dos cifras), es un dominio del conjunto infinito de los números enteros.

Atributo: se denomina *atributo* de un dominio a la intención de ese dominio, y el valor del atributo será la extensión del dominio. Un atributo identifica la semántica de un dominio para la descripción de un problema; es decir, el significado de un dato, que forma parte del sistema, en el mundo real. Un atributo, un dato, es definido en función de un dominio (la intención), y el conjunto de posibles valores que pueden ser medidos para ese atributo determina el posible conjunto de extensiones de ese dominio. Por ejemplo, el atributo *edad* considerado en un determinado problema en el que se trate la edad de una serie de objetos *Personas*, puede ser definido sobre la base del dominio de los números enteros de dos cifras.

Entidad: una *entidad* es un tipo de objeto (un conjunto) definido en base a la agregación de una serie de atributos. Una entidad corresponde a la caracterización de objetos del mundo real, los cuales son definidos y diferenciados del resto de los objetos, sobre la base del conjunto de atributos que se agregan. Las entidades tienen, como los conjuntos, intenciones y extensiones. La intención de una entidad es denominada *Tipo de Entidad* y representa el posible conjunto de objetos definidos en base a la agregación de un mismo conjunto de atributos; es decir, en términos de abstracción, un tipo de entidad representa la clasificación de las entidades individuales. La extensión de un tipo de entidad es denominada

Conjunto de Entidades, y se corresponde con todos los valores que en un momento dado están asociados con cada atributo que define el tipo de entidad.

Por ejemplo, el tipo de entidad *Persona*, puede venir definida en base a la agregación de atributos tales como: *<nombre, edad, ciudad, estado-civil>*, y cada uno de estos atributos están definidos en un dominio determinado. La extensión del tipo de entidad *Persona* corresponde con todas las personas existentes en un momento dado en el problema, por ejemplo, *<José, 28, Madrid, casado>*, *<Antonio, 34, Lugo, soltero>*, *<Pedro, Córdoba, 25, soltero>*, etc.

Interrelación: la interpretación dada para las entidades puede ser igualmente propuesta para las interrelaciones. Así, una *interrelación* representa la relación existente entre entidades, denominándose *tipo de interrelación* a la intención de la relación existente entre dos tipos de entidad. Por ejemplo, el tipo de interrelación *Dueño_de* entre el tipo de entidad *Persona* y el tipo de entidad *Coche*, representa la relación existente entre las personas que *son dueñas* de algún coche. La extensión de un tipo de interrelación es denominada *Conjunto de Interrelaciones* y representa a cada una de las posibles correspondencias entre los conjuntos de entidades que intervienen en el tipo de interrelación. En términos de abstracción, un tipo de interrelación corresponde con la agregación de dos o más tipos de entidad (por ejemplo, el tipo de interrelación *Dueño_de* puede ser representado como la agregación de los tipos de entidad *Persona* y *Coche*). De igual forma, un tipo de interrelación puede representar la generalización de dos o más tipos de interrelación (por ejemplo, un tipo de interrelación *Poseedor_de* puede ser visto como la generalización de dos interrelaciones *Dueño-1_de* entre los tipos de entidad *Persona* y *Coche*, y el tipo de interrelación *Dueño-2_de* entre los tipos de entidad *Persona* y *Casa*).

El modelo de datos *E-R* utiliza los conceptos de tipo de entidad y tipo de interrelación como estructuras básicas para la representación de un problema del mundo real. En este modelo los tipos de entidad son denominados *conjuntos de entidades* y los tipos de interrelación son denominados *conjuntos de interrelaciones*; denominando *entidades* e *interrelaciones* a cada uno de los objetos que pertenecen a estos tipos respectivamente[7].

2.3.1. Entidades e interrelaciones en el modelo E-R

En el modelo *E-R* se considera que una entidad es un objeto real o abstracto que forma parte del sistema o problema en estudio y que cumple las siguientes propiedades:

[7] A lo largo del texto seguiremos utilizando el término *tipo de entidad o interrelación* en lugar de *conjunto de entidad o interrelación* respectivamente, e *instancia de entidad o interrelación* por ser más generales y representar más adecuadamente la realidad desde un punto de vista semántico.

- Tiene existencia propia. Es decir, desde el punto de vista en el cual se estudia el sistema y al nivel de abstracción en el cual es considerado, la entidad existe como un elemento que interviene en el comportamiento global del sistema.
- Es distinguible del resto de las entidades (objetos) que intervienen en el sistema.
- Las entidades de un mismo tipo están definidas en base a un mismo conjunto de atributos, cada uno de ellos definido en un mismo dominio.

Un tipo de interrelación es definido como una relación matemática entre *n* tipos de entidades. Así, el tipo de interrelación *R* puede ser definida como:

$$R = \{[e_1, e_2, ..., e_n] \mid e_1 \in E_1, e_2 \in E_2, ..., e_n \in E_n\},$$

donde E_1, E_2, ... E_n son tipos de entidad que intervienen en el tipo de interrelación.

Dos tipos de entidad son consideradas por el modelo *E-R*:

Tipo de entidades fuertes: cuya existencia no depende de la existencia de ningún otro tipo de entidad en la consideración del problema.

Tipo de entidades débiles: cuya existencia depende de la existencia de un tipo de entidad fuerte.

A su vez los tipos de entidad débiles pueden ser de dos tipos:

Debilidad por identificación: por lo que una entidad débil no puede ser identificada (reconocida y diferenciada del resto de las entidades del mismo u otro tipo) a no ser que se identifique una entidad fuerte por cuya existencia está presente la debilidad.

Debilidad por existencia: por lo que una entidad débil puede ser identificada sin necesidad de identificar la entidad fuerte por la cual existe.

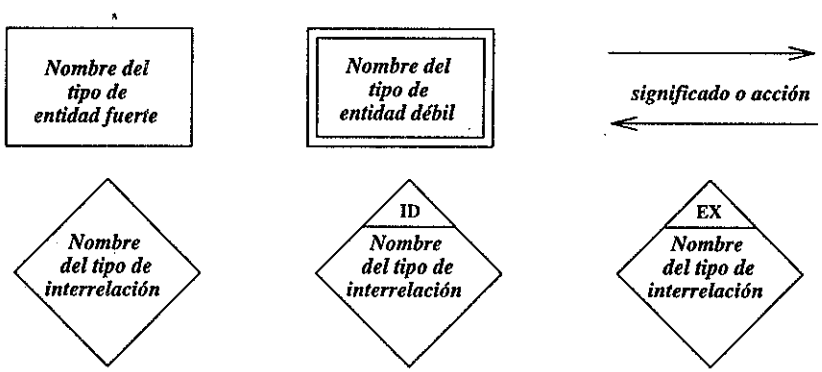


Figura 2.4 Esquemas simbólicos utilizados en el modelo E-R

Una debilidad de identificación implica una debilidad de existencia, pero no al contrario. Un tipo de entidad débil por existencia no necesariamente requiere para su identificación del tipo de entidad fuerte por la cual tiene existencia.

Al igual que los tipos de entidad pueden ser *fuertes* o *débiles*, los tipos de interrelación pueden ser *fuertes* o *débiles*. Un tipo de interrelación fuerte representa la relación entre dos tipos de entidad fuertes y, viceversa, un tipo de interrelación débil representa la relación entre un tipo de entidad fuerte y una débil, o bien, dos tipos de entidad débiles. En este último caso, alguno de los tipos de entidad tiene el *papel* de fuerte en lo referente a la semántica de la relación, aunque sea débil desde la óptica global del problema.

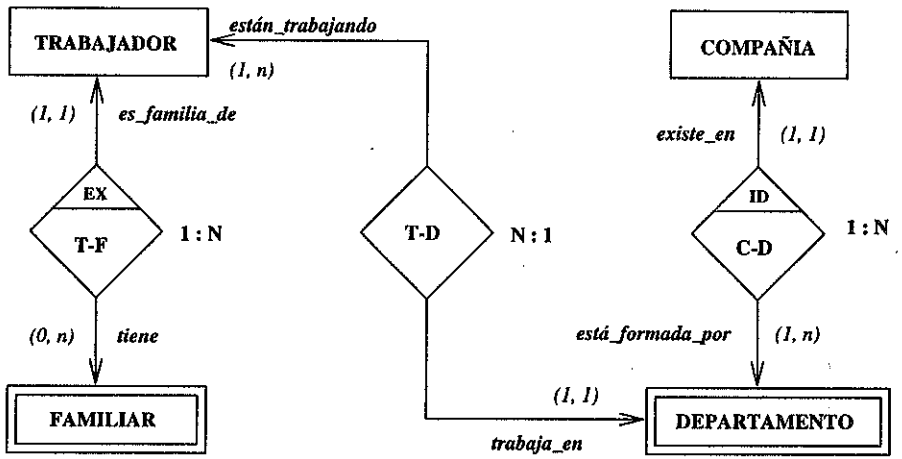


Figura 2.5 Ejemplo de un diagrama E-R

Como muestra la figura 2.4, los tipos de entidad son representados mediante un rectángulo etiquetado, y los tipos de interrelación mediante un rombo igualmente etiquetado. Los tipos de entidad débiles son representados con un doble rectángulo etiquetado, y los tipos de interrelación débiles mediante un rombo en el cual se indica el tipo de debilidad existente (**EX**istencia, **ID**entificación).

Gráficamente, las relaciones entre los tipos de entidad son representadas por arcos cuya punta de flecha señala a las entidades que intervienen en el tipo de interrelación. Cada arco es etiquetado con el significado, semántica o acción que las entidades realizan para la relación que se representa.

La figura 2.5 muestra un ejemplo de un diagrama *E-R* en el que se presentan los tipos de entidad e interrelación anteriormente descritos. En este diagrama se presentan cuatro tipos de entidad: *Trabajador, Familiar, Compañía* y *Departamento*. Los tipos de entidad *Trabajador* y *Compañía* son tipos de entidad fuertes, mientras que los tipos de entidad *Familiar* y *Departamento* son tipos de entidad débiles. La debilidad del

tipo de entidad *Familiar* es por existencia, de forma que una entidad *Familiar* no existirá si no existe una *Trabajador* con la cual esté relacionada a través del tipo de interrelación *(T-F)*. Por otro lado, la debilidad del tipo de entidad *Departamento* es tanto por existencia como por identificación. De esta forma, una entidad *Departamento* no existirá si no existe una entidad *Compañía* asociada con la cual esté relacionada a través del tipo de interrelación *(C-D)*, y además, una entidad *Departamento* no puede ser identificada por sí sola, sino que es necesario identificar a la entidad *Compañía* con la cual está relacionada para poder diferenciarla del resto de las entidades *Departamento*.

Se muestra también en la figura 2.5, cómo se representa en un diagrama *E-R* el significado o acción que cada tipo de entidad realiza en el tipo de interrelación que relaciona dos o más tipos de entidad. Así, en el tipo de interrelación *(T-D)*, se indica que una entidad *Trabajador trabaja_en* entidades *Departamento*, y que en una entidad *Departamento están_trabajando* entidades *Trabajador*. De esta forma, en los arcos que unen los tipos de interrelación con los tipos de entidad se describe el significado en el mundo real, en el mundo del problema en estudio, de las relaciones existentes entre los tipos de entidad.

En un diagrama *E-R* es necesario representar, para cada tipo de interrelación, las *cardinalidades* con las que cada tipo de entidad interviene en el tipo de interrelación. Si un tipo de interrelación representa una correspondencia entre conjuntos (tipos de entidad), para identificar completamente esta correspondencia es necesario explicitar qué número de entidades de cada tipo intervienen en cada una de las correspondencias (cada una de las interrelaciones).

En un diagrama *E-R*, las cardinalidades se representan mediante una pareja de datos (en minúsculas) en la forma:

(cardinalidad mínima, cardinalidad máxima)

asociada a cada uno de los tipos de entidad que intervienen en un tipo de interrelación dado.

En la figura 2.5 se muestra esta información. Así, por ejemplo, para el tipo de interrelación *(T-F)* existente entre los tipos de entidad *Trabajador* y *Familiar*, el tipo de entidad *Trabajador* participa en esta interrelación con las cardinalidades *(1,1)*, que informa que una entidad correspondiente al tipo *Familiar* está relacionada con 1 como mínimo y 1 entidad como máximo del tipo *Trabajador*. Y una entidad del tipo *Trabajador* puede estar relacionada, a través del tipo de interrelación *(T-F)*, con 0 como mínimo y *n* (muchas) entidades como máximo del tipo *Familiar*.

Las parejas de cardinalidades mínima y máxima con la que un tipo de entidad puede intervenir en un tipo de interrelación son: *(0,1), (1,1), (0,n), (1,n)* o *(m,n)*, que representan los diferentes tipos de correspondencias que pueden presentarse entre los elementos de los conjuntos relacionados.

Es conveniente también acompañar a la representación de los tipos de interrelación en un diagrama *E-R* de las cardinalidades máximas (en mayúsculas) con las que intervienen los tipos de entidad relacionados en el tipo de interrelación. Así, se puede observar en la misma figura 2.5 que, acompañando al tipo de interrelación *(T-F)*, aparecen indicadas las cardinalidades máximas con las que intervienen los tipos de entidad *Trabajador* y *Familiar* en el tipo de interrelación, *1:N* en este caso; que informa de una relación o correspondencia de un trabajador con muchos familiares.

2.3.2. Descripción de los tipos de entidad e interrelación

Los tipos de entidad, en el modelo *E-R*, son caracterizados en base a un conjunto de propiedades. A este conjunto de propiedades se les denominan *atributos*. Un atributo es una interpretación de un dominio para un determinado tipo de entidad, aportando un significado en el mundo del problema en el cual este tipo de entidad es considerado.

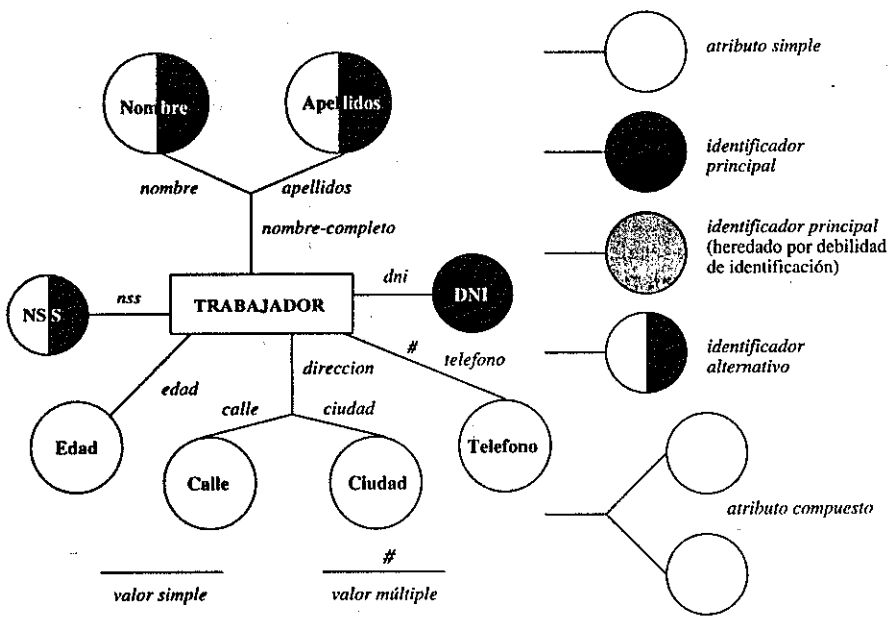


Figura 2.6 Representación de los atributos en los tipos de entidad

En el modelo *E-R*, los dominios son representados mediante elipses o círculos etiquetados. Cada tipo de entidad es relacionado con un determinado dominio a través de un atributo, y esta relación es representada mediante un arco, etiquetado con el nombre del atributo, que une el tipo de entidad con el dominio.

Como se muestra en la figura 2.6, los atributos pueden ser simples o compuestos: un atributo simple aporta un significado semántico a un único dominio, mientras que un atributo compuesto aporta una semántica a un conjunto de dominios.

Generalmente los atributos pueden tomar un único valor para cada una de las entidades con las que están relacionados, pero como se muestra en la figura 2.6, ciertos atributos pueden tomar un conjunto de valores para cada una de las entidades. En el modelo *E-R*, a los atributos que pueden tomar un conjunto de valores se les denomina *atributos múltiples* y son representados mediante el signo # acompañando al nombre del atributo, o bien indicando la cardinalidad o número de valores que pueden tomar (2, 3, ..., n) al lado del círculo que representa al dominio en el que el atributo se ha definido.

Para simplificar la representación de un modelo haciendo uso de estos diagramas, tradicionalmente se les asigna a los atributos el mismo nombre que al dominio en el cual están definidos. De esta forma, se puede eliminar la etiqueta del arco que une el dominio con el tipo de entidad, presuponiendo un nombre del atributo igual al del dominio representado por el círculo.

Como se ha comentado anteriormente, una de las propiedades que debe tener un tipo de entidad para ser considerada como tal en la representación de un problema es que no exista confusión ni en cuanto a los tipos de entidad (o interrelaciones) ni en cuanto a las instancias de estos tipos.

Para eliminar cualquier ambigüedad en la representación de estos objetos, tanto los tipos de entidad como los tipos de interrelación deben tener asignado un nombre único en el modelo. Por otra parte, es necesario establecer un mecanismo para distinguir sin ambigüedad a cada una de las instancias de la entidad e interrelación del resto del mismo tipo. Esto se consigue mediante la especificación, para los tipos de entidad, del conjunto de atributos identificadores.

Se denomina *identificador* de un tipo de entidad al conjunto de atributos (tal vez uno sólo) que no toma el mismo valor para dos entidades diferentes del mismo tipo. Estos conjuntos de atributos deben cumplir la condición de ser mínimos; es decir, el conjunto de atributos identificadores será tal que si se eliminase algún atributo de este conjunto dejaría de cumplir la propiedad de identificador.

Para un tipo de entidad puede haber más de un conjunto de atributos que satisfagan esta condición, siendo *candidatos* para desempeñar este papel de identificación. En estos casos, a uno de estos grupos se le asignará el papel de *identificador principal*, y al resto el de *identificador alternativo*. En el diagrama *E-R* estos tipos de atributos se representan con un círculo en negro y un círculo con la mitad relleno y la mitad vacío respectivamente (ver figura 2.6).

Si entre el conjunto de atributos que forman parte de un tipo de entidad no existe ningún conjunto de atributos que satisfagan la propiedad de identificación y no se trata de un tipo de entidad débil, es necesario introducir un nuevo atributo, ajeno a

la estructura del tipo de entidad, que satisfaga esta propiedad, puesto que si no, no sería posible la identificación de diferentes entidades del mismo tipo.

Se ha tomado el criterio de representar con un círculo sombreado aquellos atributos que forman parte del identificador de un tipo de entidad débil pero que pertenecen a otro tipo de entidad con el que participa en un tipo de interrelación débil por identificación, como así se muestra en la figura 2.6. Con ello se pretende que el lector pueda *seguir* más fácilmente los ejemplos presentados en este texto, al presentarle en cada figura todos y cada uno de los atributos que deben ser considerados para identificar a cada uno de los tipos de entidad, independientemente de que esos atributos sean propios o pertenecientes a otro tipo de entidad con el cual mantiene un tipo de interrelación débil por identificación.

Al igual que las instancias de los tipos de entidad deben poder identificarse sin ambigüedad, las instancias de los tipos de interrelación también. Una interrelación es identificada, generalmente, por la concatenación o agregación de los atributos que identifican las entidades relacionadas. Igualmente, los tipos de interrelación pueden tener asociados atributos al igual que los tipos de entidad. Un atributo asociado a un tipo de interrelación tiene la función de caracterizar la relación entre dos entidades, aportando información a la correspondencia entre los tipos de entidad.

Consideremos el ejemplo de la figura 2.7, en el que se muestra a dos tipos de entidad *Profesor* y *Alumno* relacionados mediante un tipo de interrelación que representa el hecho de que un *Profesor* enseña o imparte docencia a uno o varios alumnos mientras que a un *Alumno* sólo le imparte docencia uno y sólo un *Profesor*. Cada uno de estos tipos de entidad viene caracterizado por un conjunto de atributos, siendo los atributos *matricula* y *dni* los atributos identificadores para las entidades *Alumno* y *Profesor* respectivamente.

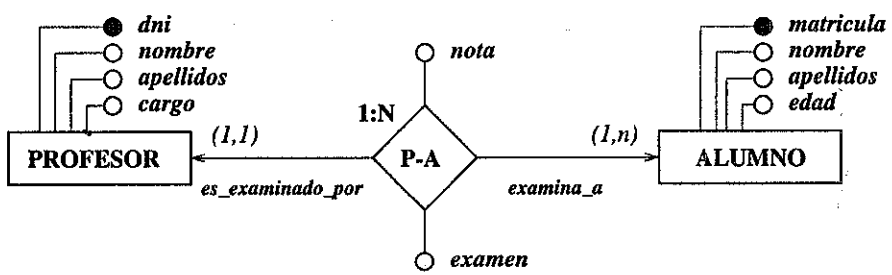


Figura 2.7 Diagrama E-R de la relación entre profesores y alumnos

Si consideramos que deseamos representar que cada profesor examina un número variable de veces a cada alumno al que le imparte docencia, y que en cada examen se le asigna una calificación a cada uno de los alumnos, el esquema conceptual del problema es el representado en la figura 2.7.

Se observa que el tipo de interrelación entre los tipos de entidad *Profesor* y *Alumno*, denotado por *(P-A)*, tiene asociados dos atributos *examen* y *nota*, atributos que aportan información a la relación entre las instancias de los dos tipos de entidad. El diagrama *E-R* de la figura 2.7 debería representar que aunque a un alumno sólo le imparte docencia un profesor, existe una serie de calificaciones o notas que el profesor imputa a cada alumno para los exámenes que realiza.

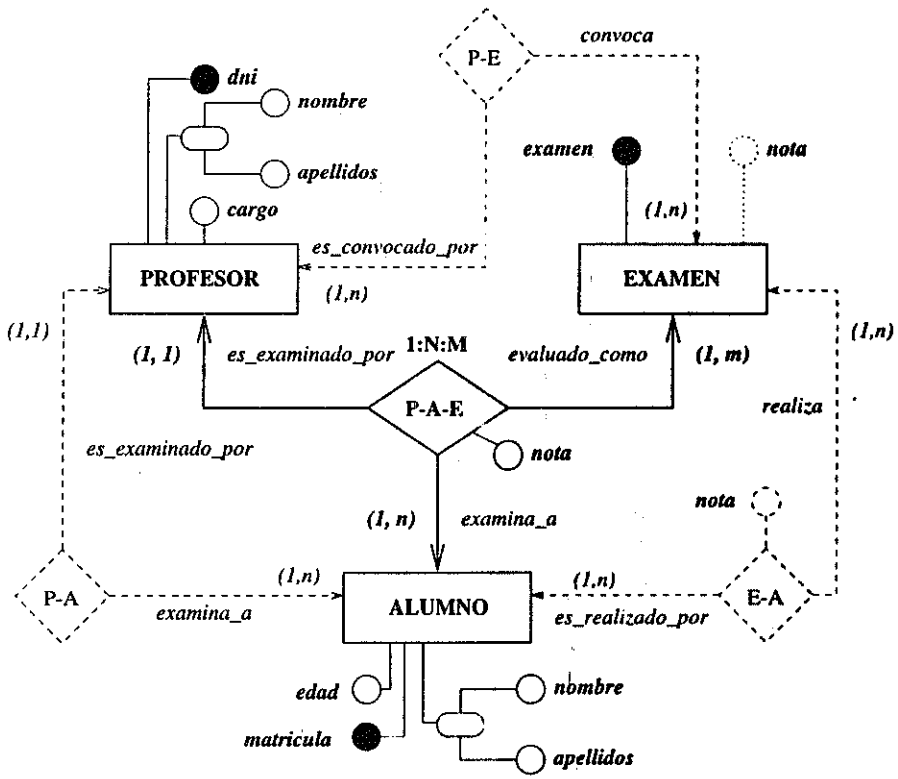


Figura 2.8 Refinamiento del diagrama E-R de los profesores y alumnos

En muchas ocasiones un tipo de interrelación en la que estén presentes atributos puede (y a veces debe, como en el ejemplo descrito) descomponerse en un tipo de entidad que mantiene una serie de interrelaciones con los tipos de entidad que participaban en el tipo de interrelación. Dependiendo de las características del problema a representar, será necesaria la consideración de una nueva interrelación *n-aria* o *n* interrelaciones binarias. En la figura 2.8 se muestra esta descomposición. El tipo de interrelación *(P-A)* se ha descompuesto en un nuevo tipo de entidad denominado *Examen*, y se ha considerado un nuevo tipo de interrelación *ternaria* entre los tipos de entidad *Profesor, Alumno* y *Examen*. Esta descomposición clarifica el

hecho de que un profesor podrá calificar a un mismo alumno en una serie de exámenes diferentes, teniendo la misma o distinta nota.

2.3.2.1. REFINANDO EL PROBLEMA DE PROFESORES Y ALUMNOS

En la figura 2.8 se ha representado el nuevo tipo de interrelación ternaria existente entre los tipos de entidad *Profesor, Alumno*, y el nuevo tipo considerado, *Examen*.

Se aprecia que el atributo *nota*, que en la figura 2.7 caracterizaba el tipo de interrelación *(P-A)*, sigue formando parte del nuevo tipo de interrelación *(P-A-E)*, mientras que el atributo *examen* pasa a formar parte, como atributo identificador, del tipo de entidad *Examen*.

De esta forma, se está representando que un alumno es calificado por un profesor con una nota en una serie de exámenes. El alumno se puede presentar a un conjunto de exámenes diferentes, obteniendo, en cada uno, la misma o diferente calificación, y siendo calificado por un único profesor.

Si el atributo *nota* se hubiera considerado como parte del tipo de entidad *Examen*, caracterizaría a las entidades de este tipo y, por tanto, sería independiente de las relaciones que mantuviera este tipo de entidad con las otras entidades presentes en el problema. Este hecho daría lugar a que, con independencia de los alumnos que se presentasen a los exámenes y de los profesores que los calificasen, todos obtendrían la misma nota.

En ocasiones, es conveniente la descomposición de los tipos de interrelación ternarias en una serie de tipos de interrelaciones binarias, como así se ha mostrado también en la figura 2.8 en líneas de trazos. Si se observa este diagrama (representado en líneas de trazos) se puede apreciar que para que *nota* sea un atributo del tipo de entidad *Examen* (representado en línea de puntos), se debe cumplir que *Examen* sea una entidad débil por identificación con respecto a *Alumno* y, por tanto, el identificador del tipo de entidad *Examen* sea, por ejemplo, una concatenación de los atributos *examen* y *matricula*. En caso contrario, todos los alumnos que realizaran un mismo examen tendrían el mismo valor del atributo *nota*, pues éste sólo es dependiente del valor del atributo *examen* y no del alumno que lo realiza, dado que varios alumnos pueden realizar el mismo examen (al igual que se describió anteriormente cuando se consideraba el tipo de interrelación ternaria *(P-A-E)*).

En este último caso, la relación entre los tipos de entidad *Examen* y *Alumno* sería 1 → muchos, dado que un examen correspondería a un único alumno, puesto que vendrá identificado por el par *(examen, matricula)*.

Como esta situación no es la que se desea representar, es necesario que el atributo *nota* se considere como un atributo del tipo de interrelación entre *Examen* y *Alumno*, como así se muestra en la misma figura.

2.3.3. Los tipos de interrelación en el modelo E-R

El modelo *E-R* permite la representación de cualquier tipo de relación existente entre los objetos del mundo real que forman parte del dominio del problema en estudio. Las últimas actualizaciones del modelo *E-R*, que han dado lugar a lo que se denomina *Modelo Entidad-Interrelación Extendido* (*EE-R*), permiten la representación de cualquier tipo de relaciones existentes entre clases de objetos que considera los principios de la abstracción.

2.3.3.1. INTERRELACIONES REFLEXIVAS

Las interrelaciones reflexivas son relaciones unarias y, por tanto, consideran que en el tipo de interrelación se ve involucrado un único tipo de entidad. Por ejemplo, consideraremos el tipo de interrelación presentado en la figura 2.9, que describe la relación existente entre el tipo de entidad *Trabajador* con ella misma, y que representa que un trabajador es jefe de 0 o varios trabajadores, mientras que un trabajador sólo es dirigido por 0 (el jefe de todos) o 1 trabajador.

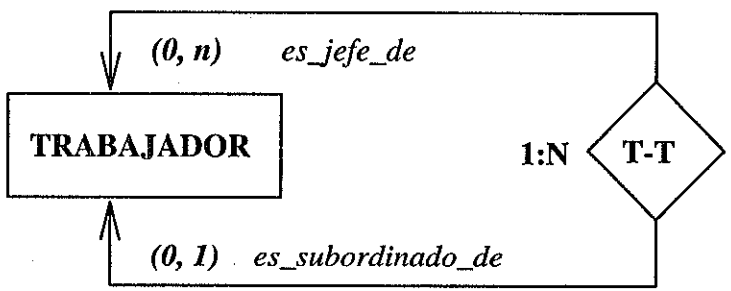


Figura 2.9 Tipo de interrelación reflexiva

Se trata de un tipo de interrelación reflexiva en la que interviene un único tipo de entidad que desempeña dos papeles diferentes en el mismo tipo de interrelación. No todos los modelos de datos permiten representar este tipo de interrelaciones, tal y como se presentan en el mundo real. El modelo *EE-R* permite este tipo de interrelaciones, aunque hay que tener en cuenta que no será siempre fácil la traslación de esta representación conceptual a una representación lógica.

2.3.3.2. INTERRELACIONES EXCLUSIVAS

En un problema del mundo real, un tipo de entidad puede mantener relaciones con un conjunto de otros tipos de entidad, pero no siempre estas relaciones son independientes. Consideremos el ejemplo que se muestra en la figura 2.10, en el que se presentan tres tipos de entidad *Articulo*, *Proveedor* y *Fabricante*, y el problema en el que los artículos son suministrados por los proveedores o por los fabricantes, pero un artículo nunca puede ser suministrado por un proveedor que no fabrica el artículo, de forma que si el fabricante puede suministrarlo, en ningún momento será solicitado ese artículo a ningún proveedor.

Para indicar la exclusividad entre dos tipos de interrelación que mantiene un mismo tipo de entidad se procede a representar un segmento que corta a los dos arcos que representan la relación del tipo de entidad con los tipos de interrelación exclusivas.

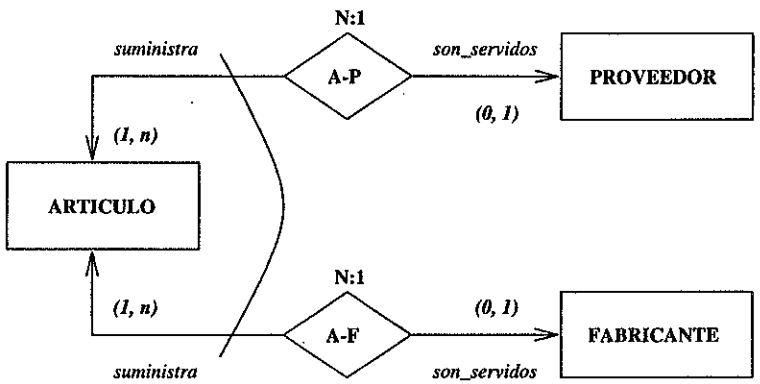


Figura 2.10 Tipo de interrelación exclusiva

2.3.4. Generalización y herencia en el modelo EE-R

El modelo *EE-R* permite representar las relaciones jerárquicas existentes entre los tipos de entidad de los problemas del mundo real. Como se ha descrito en las secciones anteriores, la generalización es una abstracción que identifica una relación jerárquica que representa un tipo de entidad *ES_UN* subtipo de otro tipo de entidad representada a un nivel de abstracción mayor. Este tipo de relaciones entre tipos de entidad implica la consideración de tipos de entidad (o supertipos) y de subtipos de entidad (clases, superclases y subclases de objetos).

Un subtipo de entidad es un tipo de entidad que mantiene un tipo de interrelación jerárquica con otro tipo de entidad, y que:

- Representa a un conjunto de entidades cuyas propiedades y comportamiento general es considerado por el tipo de entidad con la que mantiene el tipo de interrelación.
- La relación jerárquica puede ser *n-aria* entre un tipo de entidad y un conjunto de subtipos de ese tipo de entidad.
- Las propiedades y el comportamiento de los subtipos son heredados del tipo de entidad con el cual mantienen un tipo de interrelación jerárquica. La *herencia* es una abstracción incorporada al modelo *E-R* recientemente e implica la consideración de que con una única definición de las propiedades y comportamiento de un conjunto de entidades, esta definición es automáticamente considerada para todos aquellos conjuntos con los que exista una relación jerárquica (una especialización).

- Bien las propiedades y/o bien el comportamiento de un subtipo pueden y deben cambiar con respecto a otros subtipos que intervengan en la misma relación jerárquica *n-aria* entre todos estos subtipos y un mismo tipo de entidad. Puesto que los subtipos son tipos de entidad a un nivel de abstracción menor, éstos deben poder distinguirse sin ambigüedad del resto de los tipos de entidad existentes en la representación del problema y, por tanto, deben tener un conjunto de propiedades que permita esta discriminación.
- Para cada subtipo de entidad pueden ser redefinidas tanto las propiedades como el comportamiento del tipo de entidad con la que mantienen un tipo de interrelación jerárquica. Esta característica permite la existencia de mecanismos para diferenciar a los diferentes subtipos de una misma clase y, además, permite la cancelación condicionada de la herencia producida por la relación jerárquica.
- Un tipo de entidad puede ser un subtipo para más de un tipo de entidad con las que puede mantener diferentes relaciones jerárquicas. Esta característica, denominada *herencia múltiple*, permite que un tipo de entidad herede propiedades y comportamiento de más de otro tipo de entidad. La herencia múltiple puede dar lugar, en ocasiones, a inconsistencias en las propiedades y/o comportamiento que se hereda, lo que se debe solucionar mediante la redefinición de las herencias.

Un tipo de interrelación jerárquica representa una especialización de un tipo de entidad en otros tipos de entidad. Esta especialización puede ser debida a:

1. Una diferencia en cuanto al número de propiedades que definen los subtipos de entidad. De esta forma, diferentes subtipos que mantienen un mismo tipo de interrelación jerárquica son definidos sobre la base de la agregación de un conjunto diferente de propiedades, si bien entre el conjunto de subtipos puede existir un subconjunto de propiedades comunes.
2. Diferentes valores que pueden ser medidos para una propiedad que existe en el conjunto de subtipos que mantienen un mismo tipo de interrelación jerárquica.
3. Que se cumplan ambas condiciones.

La especialización de un tipo de entidad en un conjunto de subtipos puede ser exclusiva o inclusiva. Una especialización *exclusiva*, denominada ***especialización sin solapamiento***, representa el hecho de que una instancia del tipo de entidad más general sólo puede pertenecer o estar asociada a una y sólo una instancia de los subtipos de entidad especializados.

Una especialización *inclusiva*, denominada ***especialización con solapamiento***, representa el hecho de que una instancia del tipo de entidad más general puede tener asociadas instancias de cualquiera de los subtipos.

Por otro lado, la especialización de un tipo de entidad en un conjunto de subtipos puede ser total o parcial. Una *especialización total* representa el hecho de que las entidades que son reconocidas en el problema que se está representando son de alguno de los subtipos especializados, no existiendo entidades que no pertenezcan a alguno, varios o todos estos subtipos de entidad. Una *especialización parcial* representa el hecho de que pueden existir entidades que pertenezcan al tipo de entidad y no a ninguno de los subtipos en los cuales este tipo de entidad está especializado. Una especialización parcial describe un refinamiento incompleto del problema que se representa, debido a un conocimiento incompleto del mismo y/o a una simplificación de la representación del mismo.

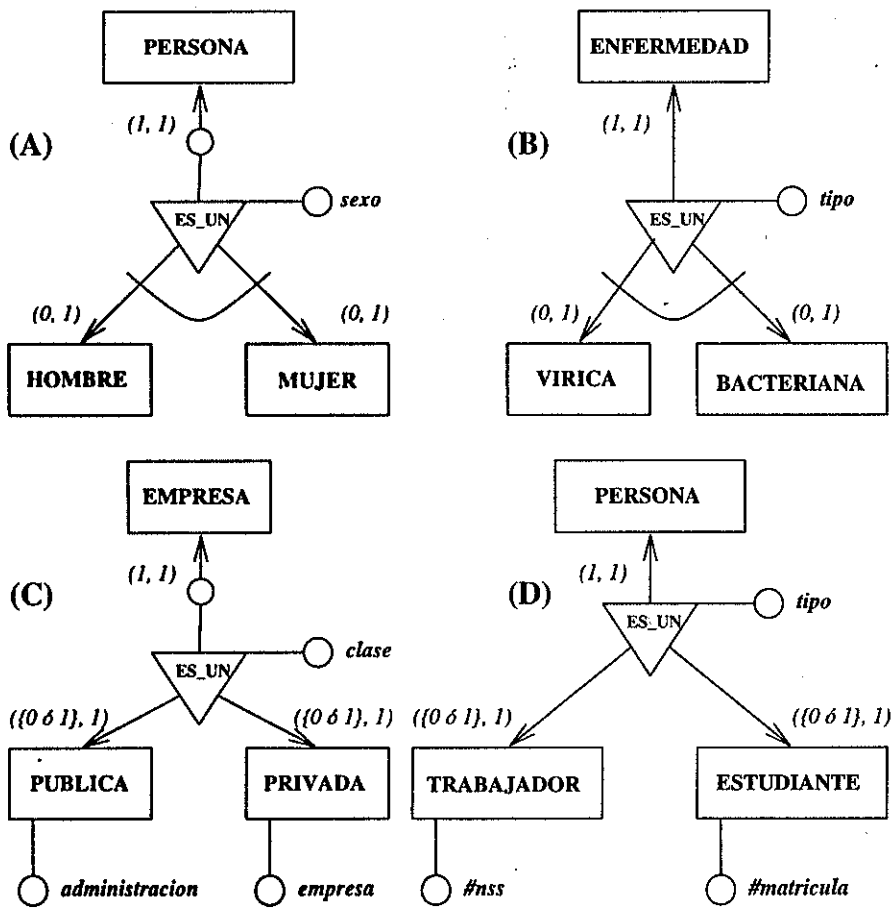


Figura 2.11 Tipos de interrelaciones jerárquicas

Por tanto, se pueden presentar cuatro tipos de interrelaciones jerárquicas que se representarían mediante el modelo *EE-R*: *total sin solapamiento, parcial sin solapamiento, total con solapamiento* y *parcial con solapamiento*, como se muestra en los ejemplos presentados en la figura 2.11-(A, B, C, D) respectivamente.

Consideremos el tipo de entidad *Persona* (figura 2.11-A), la cual puede ser especializada en dos subtipos de entidad *Hombre* y *Mujer* de forma total y sin solapamiento. Una entidad *Persona* podrá pertenecer al subtipo *Hombre* o al subtipo *Mujer* necesariamente; es decir, no existirá una entidad *Persona* que no sea de alguno de estos dos subtipos y además de forma exclusiva, por lo que una entidad pertenecerá a uno y sólo a uno de estos subtipos. Además, cada entidad de alguno de estos subtipos vendrá caracterizada por algún atributo o conjunto de atributos definidos para estos subtipos o heredados del tipo de entidad *Persona*, más el atributo *sexo*, el cual tiene la función de clasificador de las entidades *Persona* en alguno de estos subtipos.

En la figura 2.11-B se muestra un ejemplo de especialización parcial exclusiva. En este caso se ha considerado un tipo de entidad *Enfermedad* que puede ser especializada en dos subtipos *Virica* y *Bacteriana*. Este diagrama representa el hecho de que en el problema se consideran un conjunto de entidades *Enfermedad* las cuales pertenecerán bien a alguno de los subtipos considerados *Virica* o *Bacteriana*, pero que además existirán entidades *Enfermedad* las cuales no puedan ser clasificadas en ninguno de estos subtipos debido, posiblemente, al desconocimiento del valor del atributo *tipo* utilizado como discriminador.

Por otro lado, el ejemplo de la figura 2.11-C representa un refinamiento total con solapamiento en el que un tipo de entidad *Empresa* se ha refinado en dos subtipos *Pública* y *Privada*. Se está representando el hecho de que podrán existir en el dominio del problema entidades que puedan ser consideradas tanto del tipo *Publica* como *Privada*, o bien de ambos tipos al mismo tiempo y, además el hecho de que no podrán existir entidades que no puedan ser especializadas en alguno de estos dos subtipos.

El problema representado en la figura 2.11-D es diferente. En este caso se ha representado a un tipo de entidad *Persona* que puede ser refinado en dos subtipos *Trabajador* y *Estudiante* de forma parcial con solapamiento. Este ejemplo representa que una entidad *Persona* puede ser del tipo *Trabajador* y/o del tipo *Estudiante* y que además pueden existir entidades *Persona* que no puedan clasificarse en ninguno de estos dos subtipos.

En estos dos últimos ejemplos, los subtipos de entidad incorporan nuevos atributos mediante los cuales pueden diferenciarse entidades pertenecientes a los distintos subtipos (o del tipo de entidad general en el caso en que la especialización no sea total). Igualmente podrían existir atributos pertenecientes al tipo de interrelación jerárquica cuya función fuera esta diferenciación de las entidades pertenecientes a los subtipos.

2.3.4.1. LAS CARDINALIDADES EN LA JERARQUÍA

Dependiendo del tipo de interrelación jerárquica que se represente, y por tanto exista en el dominio del problema, los tipos de entidad que intervienen en el mismo

participan o pueden participar con un número determinado de ocurrencias. Así, se tienen que tener en cuenta las siguientes consideraciones en la representación de este tipo de interrelaciones:

- El tipo de entidad más general o el supertipo de entidad que es especializado participa siempre con la cardinalidad mínima 1 y con la cardinalidad máxima 1, puesto que se está representando como una entidad de este tipo puede especializarse en otros subtipos.
- Para cualquier clase de este tipo de interrelaciones jerárquicas, la cardinalidad máxima con la que participan los subtipos de entidad en el tipo de interrelación es 1, puesto que se está representando para cada entidad del supertipo una especialización o refinamiento de la misma.
- Si el tipo de interrelación es total o parcial sin solapamiento, los subtipos participan siempre con cardinalidad mínima 0, puesto que una entidad del tipo no puede pertenecer al mismo tiempo a más de un subtipo (figura 2.11-A, B).
- Si el tipo de interrelación es total o parcial con solapamiento, los subtipos pueden participar con la cardinalidad mínima 0 ó 1, puesto que una entidad del supertipo puede a su vez ser especializada en cualquiera de los subtipos simultáneamente (figura 2.11-C, D).

2.4. REPRESENTACIÓN DE LAS RESTRICCIONES EN EL MODELO EE-R

El modelo *EE-R* cuenta con algunos mecanismos para la representación de las restricciones que están presentes en los problemas del mundo real. Si consideramos que una *restricción* es una condición que está presente para un conjunto o subconjunto de objetos que están presentes en el dominio del problema, las restricciones pueden aparecer:

- En los valores que pueden ser medidos para un atributo. Si bien el atributo está definido en un determinado dominio, en el dominio del problema puede ser necesario considerar restricciones en cuanto a este dominio. Por ejemplo, un atributo *edad* definido en el dominio de los números enteros de dos dígitos y para el cual sólo se debe considerar que tome los valores comprendidos en el intervalo [18, 65].
- En los valores de las correspondencias entre conjuntos de objetos del sistema que representan los tipos de interrelación entre los tipos de entidad. Es decir, el valor de las cardinalidades máximas y mínimas.
- En la existencia de entidades pertenecientes a un determinado tipo de entidad, siempre y cuando no existan otras entidades pertenecientes a otro(s) tipo(s) de entidad (las entidades débiles o las que participan en un tipo de interrelaciones jerárquicas).

La representación de las restricciones existentes en un problema del mundo real está directamente ligada a la *semántica* del problema. La representación de la semántica es extremadamente compleja si no se utiliza un lenguaje natural, puesto que la semántica es dependiente del contexto en el que se encuentran los *ítems* de información acerca del problema. Pocos modelos de datos son capaces de representar de forma efectiva la semántica de un problema del mundo real, y menos aún mediante una representación gráfica. En el modelo *EE-R* es posible representar gráficamente parte de las restricciones antes señaladas, aunque no así todas ellas, como el límite o intervalo de valores que puede tomar un atributo, necesitándose para ello una representación textual del problema del mundo real.

DEFINICIÓN DEL PROBLEMA: *Nombre del problema*

DESCRIPCIÓN DE LOS TIPOS DE ENTIDAD

Se relacionan cada uno de los tipos de entidad con indicación de la siguiente información:
Tipo de entidad *Nombre*
Hereda de
Para cada uno de los atributos que caracterizan el tipo de entidad se indica:
Atributo *Nombre*
Si el atributo es compuesto, se declaran los atributos componentes
Lista de atributos componentes *Lista de Nombres*
Para cada atributo simple se especifica la siguiente información:
Dominio *Nombre del dominio*
Cardinalidad *Multiplicidad*
Restricciones
Se declara el rango de valores permitidos
Intervalo de valores *(valor mínimo, valor máximo)*
Lista de valores *(colección de valores permitidos)*
Identificador principal *Lista de atributos*
Se declara la lista de los identificadores alternativos para el tipo de entidad
Identificador alternativo *Lista de atributos*

DESCRIPCIÓN DE LOS TIPOS DE INTERRELACIÓN

Se relacionan cada uno de los tipos de interrelación con indicación de:
Tipo de interrelación *Nombre*
Se especifican cada uno de los tipos de entidad que participan en el tipo de interrelación
Tipo de entidad *Nombre*
Se especifica la cardinalidad con la que participa el tipo de entidad
Cardinalidad mínima *Cardinalidad mínima*
Cardinalidad máxima *Cardinalidad máxima*
/ Se especifican cada uno de los atributos que caracterizan el tipo de interrelación de la misma forma que se ha realizado para los tipos de entidad */*

Tabla 2.2 Sintaxis general en el modelo EE-R

2.5. SINTAXIS DEL MODELO EE-R

La representación textual de un problema del mundo real mediante el modelo *EE-R*, requiere la representación mediante una gramática preestablecida del conocimiento e información acerca de las características del problema; es decir, de cada uno de los tipos de entidad e interrelaciones reconocidos y de la estructura de cada uno de estos tipos.

Mediante una descripción simple se deben describir todos los elementos del problema y todas aquellas características de estos que permiten su identificación en el mundo real, así como dentro del conjunto de los objetos representados.

Como muestra la tabla 2.2, se describirán cada uno de los tipos de entidad, sus atributos, dominios y restricciones, así como los conjuntos de atributos que pueden ser considerados como identificadores de las entidades de este tipo. Y, además, se describirá la lista de tipos de interrelación existentes entre los tipos de entidad. Para cada tipo de interrelación se explicitarán los tipos de entidad que intervienen y las cardinalidades en que lo hacen, así como el conjunto de atributos que puedan estar asociados a los tipos de interrelación.

La sintaxis presentada en la tabla 2.2 no pretende ser una descripción formal del modelo *EE-R*, sino una guía a seguir que permita especificar la representación de un problema, además de gráficamente, sintácticamente favoreciendo la calidad y legibilidad de la documentación que se genere del proceso de análisis del problema en estudio.

CAPÍTULO 3

EL MODELO DE DATOS RELACIONAL

3.1. LA TEORÍA RELACIONAL

La representación de la información correspondiente a los problemas del mundo real no es una tarea fácil, y menos aún cuando aumenta el número de objetos a representar y las relaciones entre ellos. El número de objetos y relaciones que forman parte de un esquema va a depender de:

1. La información que debe formar parte del conocimiento necesario para el análisis y solución del mismo por métodos manuales o computerizados.
2. Los requisitos funcionales que determinan qué procesos se desean realizar a la información, y de qué forma, para que el tiempo de obtención de las soluciones parciales o total del problema a resolver sea mínimo.

Está claro que ambos factores están íntimamente relacionados. Un requisito funcional, una operación a realizar con la información, va a determinar la necesidad de una estructura de la información adecuada para un tratamiento efectivo de la misma y viceversa.

Si se tiene además en cuenta que la necesidad de conocimiento sobre los problemas es ilimitada y que la necesidad de obtención de información sobre el mismo es requerida cada día en un tiempo menor y que, sobre todo, los problemas no son estáticos sino que evolucionan con el tiempo influenciados tanto por el propio sistema como por el entorno del mismo, la representación de la información correspondiente a un problema está sujeta a continuas modificaciones, reestructuraciones y, sobre todo,

ampliaciones del número de objetos y relaciones entre ellos que intervienen en el problema.

Estos hechos dan lugar a que los esquemas (la representación bajo un modelo de datos de la información correspondiente a un problema de forma que pueda ser entendible por un sistema de gestión de bases de datos, por ejemplo) estén sujetos a continuas modificaciones que originan, si el modelo utilizado no es el adecuado, una complicación excepcional en la representación de la información. Una representación difícilmente entendible tanto por el usuario como por el administrador de la base de datos, que origina un difícil mantenimiento de la información y, por tanto, un bajo desempeño de la administración y manejo de la misma, además de otros problemas de integridad y seguridad.

No todos los modelos de datos dan lugar a esquemas complejos cuando el propio problema que representa es complejo o ha experimentado ampliaciones/ modificaciones, ni tampoco una representación compleja es directamente causada por el uso de un determinado modelo de datos. La complejidad del esquema va a depender tanto de:

- La complejidad del problema del mundo real. Es decir, de los objetos y de las relaciones existentes entre ellos.
- La calidad del análisis de ese problema, el cual originó la representación del mismo y, por tanto, sus revisiones.
- La evolución del problema y de los requisitos funcionales que originan las continuas ampliaciones y modificaciones del esquema.

Si bien es obvio que el uso de un modelo, que por su propia teoría en la que esté basado se vea menos influenciado por estos factores, dará lugar a una representación lógica de los problemas que origine esquemas que sean más entendibles, simples, claros y, por tanto, más fácilmente mantenibles, seguros y confiables que con el uso de otros modelos más sensibles a estos cambios. Y es ésta la base de la aparición del *Modelo de datos relacional.*

3.2. EL MODELO DE DATOS RELACIONAL

Fue *E.F. Codd* quien desarrolló en *IBM-San José (California)* el modelo de datos relacional. Este modelo está basado en conceptos muy sencillos teniendo asociada la teoría de normalización de relaciones que tiene por objeto la eliminación de los comportamientos anómalos de las relaciones durante los procesos de manejo de la información que representan y la eliminación de redundancias superfluas, facilitando así, la comprensión del esquema en cuanto a las relaciones semánticas existentes entre los objetos del dominio del problema.

El modelo relacional es un modelo lógico de datos, todos los principios que van a ser tratados en este capítulo hacen referencia a la descripción lógica de la información y, por tanto, no son directamente aplicables a la representación física de

la misma. Una base de datos relacional puede ser estructurada físicamente de múltiples formas, aunque como es lógico, la representación física deberá satisfacer y representar, de alguna forma, las relaciones y restricciones lógicas del esquema relacional.

El modelo relacional propone una representación de la información que:

- Origine esquemas que representen fielmente la información, los objetos y relaciones entre ellos existentes en el dominio del problema.
- Pueda ser entendida fácilmente por los usuarios que no tienen una preparación previa en esta área.
- Haga posible ampliar el esquema de la base de datos sin modificar la estructura lógica existente y, por tanto, sin modificar los programas de aplicación.
- Permita la máxima flexibilidad en la formulación de los interrogantes previstos, y no previstos, sobre la información mantenida en la base de datos.

3.2.1. Terminología del modelo relacional

Una de las formas más naturales, y por tanto fácilmente entendibles por el usuario, de representar la información es sobre la base del uso de una representación tabular plana de la misma. Una tabla bidimensional mediante la cual se representen tanto los objetos como las relaciones entre ellos existentes en el dominio del problema.

Una tabla es una matriz rectangular que puede ser descrita de forma simple matemáticamente y que posee las siguientes propiedades:

1. Cada entrada de la tabla, es decir, cada elemento de la matriz rectangular, representa a un ítem de datos elemental.
2. Una tabla es homogénea por columnas; es decir, todos los ítems de datos elementales de una columna (en todas las filas) son de la misma clase y, por tanto, están definidos en el mismo dominio de datos y representan una misma propiedad o característica en el dominio del problema.
3. Cada columna de la tabla tiene asignado un nombre único en el conjunto de columnas de esa tabla, aunque pueden existir tablas diferentes con columnas de igual nombre.
4. Para una tabla todas las filas son diferentes, no se admiten filas duplicadas.
5. Tanto las filas como las columnas pueden ser consideradas en cualquier secuencia sin afectar, por ello, ni al contenido de la información ni a la representación semántica de la misma.

Como cualquier modelo de datos, el modelo relacional, introduce su propia terminología para nominar los objetos y elementos utilizados por el modelo para representar el dominio de la información. Por ejemplo, a una tabla o matriz rectangular se le denomina *relación*, a las filas de la misma se les denomina *tuplas* y al conjunto

de sus columnas *dominio* de la relación. Así, una base de datos relacional estará formada por un conjunto de relaciones.

3.2.1.1. RELACIÓN

Dada una serie de conjuntos D_1, D_2, D_3, ..., D_n, no necesariamente distintos, se dice que R es una relación entre estos n conjuntos si es un conjunto de n tuplas no ordenadas $(d_1, d_2, d_3, ..., d_n)$ tales que $d_1 \in D_1$, $d_2 \in D_2$, $d_3 \in D_3$, ..., $d_n \in D_n$. A los conjuntos D_1, D_2, D_3, ..., D_n se les denomina *dominios* de R, y el valor de n es el *grado* de la relación R.

ALUMNO	*matricula#*	*nombre*	*apellidos*	*curso*	*nota*
	3456	José	Pérez de la Lastra	1	5.25
	0101	María	Antúnez Sastre	2	7.80
	8743	Lourdes	Sánchez Argote	1	4.50
	1234	Antonio	Soria Madrid	3	6.35
	5674	Luis	González Silos	1	3.20
	0678	Pilar	Alcántara Badajoz	2	5.50
	0345	Dolores	Almiz Márquez	3	7.30
	2985	Manuel	Rives Fuentes	3	3.50

Tabla 3.1 Ejemplo de relación

Veamos un ejemplo de relación que se muestra en la tabla 3.1. Se trata de una tabla denominada *Alumno*, en cuyo ejemplo están presentes ocho tuplas. La relación es de grado cinco. Los cinco dominios son conjuntos de valores que representan, respectivamente: el número de matrícula de los alumnos, el nombre, los apellidos, el curso en el que están matriculados, y la nota obtenida por los alumnos.

El dominio correspondiente a la *nota* es el conjunto de todas las notas posibles que pueden ser asignadas a los alumnos aunque, como se observa en el ejemplo, no todos los valores posibles tienen que estar presentes en un momento dado en una relación en la que exista ese dominio.

Al número de tuplas de una relación en un instante dado se le denomina *cardinalidad* de la relación. Así, la relación *Alumno* tienen una cardinalidad de ocho.

Al número de columnas de una relación se le denomina *grado* de la relación. Así, la relación *Alumno* tiene un grado de cinco.

Mientras que la cardinalidad de una relación depende del momento en que ésta sea considerada, el grado de una relación es independiente del tiempo. El grado de una relación hace referencia al número de dominios que define la relación y éstos son independientes del momento en que ésta se considere. Dependiendo del grado de la relación éstas se denominan: unarias, binarias, ternarias, etc.

Se puede introducir, en estos momentos, otra definición equivalente de relación como sigue:

> *Dados una serie de conjuntos* D_1, D_2, D_3, ..., D_n, *no necesariamente distintos, se define el producto cartesiano de estos conjuntos y se denota por* $D_1 \times D_2 \times D_3 \times ... \times D_n$, *como un nuevo conjunto formado por todas las tuplas no ordenadas posibles* $(d_1, d_2, d_3, ..., d_n)$, *tales que* $d_1 \in D_1$, $d_2 \in D_2$, $d_3 \in D_3$ ... $d_n \in D_n$.
>
> *Se dice que* R *es una relación sobre los conjuntos* D_1, D_2, D_3, ..., D_n, *si es un subconjunto del producto cartesiano de los mismos.*

Como se puede observar no se define ningún orden de los elementos que forman la relación (las tuplas), puesto que una relación es un conjunto y los conjuntos no son ordenados. Si bien, desde el punto de vista de la teoría de conjuntos, el orden de las columnas sí influye —el orden en que se realiza el producto cartesiano de los conjuntos determina la estructura de los elementos del nuevo conjunto; es decir, el conjunto $D_1 \times D_2$ es distinto del conjunto $D_2 \times D_1$—, no es así en las relaciones. La razón es simple, en una relación cada columna tiene asignado un nombre único en el contexto de la relación y, por tanto, es posible distinguir el significado de los elementos constituyentes del conjunto, con independencia del orden de consideración de los mismos.

3.2.1.2. DOMINIOS Y ATRIBUTOS

Es importante aclarar desde el principio la diferencia existente entre *dominio* y *atributo*. Un atributo representa el uso de un dominio para una determinada relación; es decir, un atributo aporta un significado semántico a un dominio. Mientras que un dominio es un conjunto homogéneo definido mediante el uso de la abstracción en base a otro conjunto (véase el Capítulo 2).

Para aclarar esta diferencia, para el ejemplo de la relación mostrada en la tabla 3.1, se van a asignar nombres diferentes a los dominios y a los atributos, de la forma siguiente:

```
Define Dominio expediente      entero (4)     Fin definición;
Define Dominio primer-nombre   carácter (15)  Fin definición;
Define Dominio final-nombre    carácter (40)  Fin definición;
Define Dominio estudios        entero (2)     Fin definición;
Define Dominio nota            real (4)       Fin definición;
Define Relación Alumno
                ( matricula#:  Dominio   expediente,
                  nombre:      Dominio   primer-nombre,
                  apellidos:   Dominio   final-nombre,
                  curso:       Dominio   estudios,
                  nota:        Dominio   nota);
```

Se han definido, en este ejemplo, cinco dominios, y basándose en ellos son definidos los cinco atributos de la relación. Se observa que el dominio puede, o no, tener asignado el mismo nombre que el atributo mediante el cual se define la relación.

De igual forma, puede definirse más de un atributo sobre un mismo dominio. Por ejemplo, puede definirse una nueva relación *Alumno* de la forma:

Define Relación *Alumno*

*(*matricula#:	**Dominio**	*expediente,*
nombre:	**Dominio**	*primer-nombre,*
apellidos:	**Dominio**	*final-nombre,*
curso:	**Dominio**	*estudios,*
nota:	**Dominio**	*nota,*
edad:	**Dominio**	*estudios)*;

en la que el atributo *edad* está definido, al igual que el atributo *curso*, en el dominio estudios.

3.2.1.3. INTENCIÓN Y EXTENSIÓN DE RELACIONES

Una relación en una base de datos relacional tiene dos componentes: la *intención* o *comprensión* y la *extensión*, aunque es usual referirse a los dos componentes al mismo tiempo cuando se utiliza el término relación.

La intención de una relación hace referencia a la estructura estática del objeto del mundo real, el cual es representado mediante la relación. Es decir, la intención va a ser siempre invariante con el tiempo y se trata de la definición de las propiedades o atributos del objeto del mundo real representado por la relación, de las cuales cada una está definida en su correspondiente dominio de datos.

La intención de una relación define todas las posibles extensiones admisibles para la misma. La intención de una relación define dos aspectos:

1. Una estructura de datos nominada en la que tanto la estructura —el objeto— como los ítems de datos que la componen —los atributos/propiedades— tienen asignado un nombre único y están definidos en un determinado dominio.
2. Un conjunto de restricciones de integridad, las cuales serán tratadas en la siguiente sección de este capítulo.

Por otro lado, la extensión de una relación depende del momento específico en el cual la relación es tenida en cuenta, y hace referencia al conjunto de tuplas que forman parte de la relación en un instante dado. La extensión de una relación representa a cada uno de los objetos (tuplas), pertenecientes a un mismo tipo (relación), existentes en el dominio del problema en un momento dado.

3.2.2. Consistencia de la representación lógica relacional

La representación lógica de un problema, independientemente del modelo utilizado, debe satisfacer la representación conceptual de ese problema y, por tanto, las especificaciones, restricciones y requisitos definidos en el proceso de análisis del problema del mundo real que se esté tratando.

El modelo relacional propone una serie de reglas de integridad, las cuales permiten representar de forma consistente la semántica del problema. Pero antes de presentar estas reglas es necesaria la definición de una serie de términos.

3.2.2.1. CLAVES DE LAS RELACIONES

Es usual que en el conjunto de atributos que forman parte de la intención de una relación específica, uno o un conjunto de ellos tengan la propiedad de tomar valores únicos en el dominio del problema para cualquier extensión de esa relación y, por tanto, tengan la facultad de identificar sin ambigüedad y de forma única a una, y sólo una, de las tuplas de esa relación —un único objeto del mundo real que pertenece al tipo o clase representado por la relación—.

Por ejemplo, en la relación *Alumno* utilizada en las secciones anteriores se puede considerar:

- Que el atributo *matricula#* tiene la propiedad de identificación única.
- Que sobre la base de la extensión de la relación *Alumno* presentada en la tabla 3.1 la agregación de los atributos *apellidos* y *nombre* también tiene esta propiedad.

De esta forma, tanto los valores de los atributos *matricula#* y la agregación de los atributos *(apellidos + nombre)* tienen la propiedad de identificar sin ambigüedad a cada una de las tuplas de la relación *Alumno*.

A los atributos, tal vez compuestos, que satisfacen la propiedad de identificación única de las tuplas de una relación se les denomina ***claves candidatas*** de la relación.

Toda relación, por definición, debe tener alguna clave candidata. Como se ha presentado en el ejemplo, una clave candidata puede estar formada por uno o un conjunto de atributos. Si por definición, en el modelo relacional, no pueden existir tuplas duplicadas en una relación, al menos la agregación de todos los atributos de una relación cumplirá la condición de clave candidata para esa relación.

De entre todas las claves candidatas de una relación, en la definición del esquema se deberá especificar cuál de ellas se considera como ***clave primaria*** o *principal*, denominándose al resto de las claves candidatas como ***claves alternas***. La distinción entre clave primaria y alterna va a tener influencia sobre la implementación

física de la relación y sobre los procesos de acceso a la información mantenida en la misma, pero no sobre la semántica de la relación.

La elección de los atributos (uno o varios) que forman parte de las claves candidatas no es un proceso trivial, siendo necesario tener en cuenta que no debe existir una información *superflua* para satisfacer la función de identificación que tiene por objeto una clave candidata. Es decir, el número de atributos de la relación que forman parte de cada clave candidata debe ser el mínimo que satisface la propiedad de identificación única de las tuplas de esa relación.

Por ejemplo, en la relación *Alumno*, se podría considerar la agregación de los atributos *(matricula# + nota)* como clave candidata de esa relación. Pero esta clave candidata tendría una estructura superflua, pues si se elimina de esta clave candidata el atributo *nota*, sigue satisfaciendo la propiedad de identificación única.

3.2.2.2. INTEGRIDAD DE LOS ESQUEMAS RELACIONALES

La intención de un esquema relacional, es decir, el conjunto de las intenciones de las relaciones consideradas en el esquema, debe satisfacer las siguientes reglas de integridad mediante las cuales se garantiza la consistencia de la información que pueda ser manejada sobre la base de ese esquema:

Integridad de la clave: *Ningún atributo que forme parte de la clave candidata de una relación podrá tomar valores nulos para ninguna tupla de esa relación.*

Es una regla de integridad directamente derivable de la propia definición de relación. Si una relación representa a un conjunto que representa a una clase de objeto del dominio del problema en el cual no se admite la existencia de elementos duplicados, y cada tupla representa a cada uno de los elementos de ese conjunto identificados por la clave, entonces cualquiera de los atributos que forme parte de esta clave tendrá siempre algún valor, el valor correspondiente a la propiedad de ese objeto del dominio del problema.

Integridad de referencia: *Sea* D *un dominio primario, y sea* R_1 *una relación con un atributo* R_1.a *definido sobre el dominio* D, *entonces, en cualquier instante dado, cada valor de* R_1.a *en* R_1 *debe ser nulo o bien igual a algún valor* V, *el cual existe, en ese instante, para un atributo* R_2.b *definido en el mismo dominio* D *sobre la relación* R_2 *y en la cual está definido como clave primaria.*

A aquellos atributos R_1.a que satisfacen esta regla de integridad se les denominan ***claves foráneas***, las cuales, junto con las claves primarias, proporcionan al modelo relacional los mecanismos adecuados para representar las relaciones existentes entre los objetos del dominio del problema.

Otras restricciones: En la definición del esquema relacional pueden imponerse, en teoría (dependerá del *SGBD.* relacional utilizado), otra serie de restricciones que garanticen la integridad del modelo y, por lo tanto, de la información almacenada en la base de datos; restricciones concernientes a:

- Los valores permitidos para los atributos que forman parte de las relaciones existentes en el esquema. Por ejemplo, valor máximo y mínimo, lista de valores, etc.
- Condiciones que determinan el valor que pueden tomar los atributos. Estas condiciones pueden definirse en base a diferentes predicados: en función del valor de otros atributos de la misma o diferente relación, o al estado de la base de datos, o en función del usuario, etc.

3.3. NORMALIZACIÓN DE RELACIONES

Como se ha descrito en las secciones anteriores, el modelo relacional está soportado sobre una teoría de igual nombre basada en los principios de la teoría general de conjuntos. Si bien una relación representa a un conjunto, y como tal puede y debe ser considerada, no todos los conjuntos pueden ser considerados en un esquema relacional para satisfacer en la base de datos los siguientes objetivos:

1. No-existencia de redundancias superfluas, aminorando el espacio requerido para el almacenamiento de la información y, por tanto, reduciendo posibles problemas de integridad en la información almacenada en la base de datos.
2. Aumentar el desempeño de las operaciones de actualización de la base de datos.
3. Representar de forma coherente los objetos y relaciones existentes en el dominio del problema y cuya información es almacenada en la base de datos.
4. Aumentar el desempeño y garantizar la fiabilidad de las interrogaciones sobre la información mantenida en la base de datos.

Para satisfacer estos objetivos, las relaciones que forman parte de un esquema relacional deben satisfacer una serie de reglas que restringen el universo de relaciones/ conjuntos que pueden ser considerados en un esquema relacional.

A este conjunto de reglas se les denomina *Reglas de normalización de relaciones* y a la teoría en la que se basan se le denomina *Teoría de normalización de relaciones*.

3.3.1. Dependencias funcionales

La normalización de relaciones está basada en otra teoría, la *teoría de las dependencias*, la cual se centra en el estudio de las dependencias que presenta cada atributo de una relación con respecto al resto de atributos de la misma relación.

Dada una relación R, *se dice que el atributo* R.y $\in$ R *es funcionalmente dependiente de otro atributo* R.x $\in$ R, *y se expresa de la forma* R.x $\rightarrow$ R.y *si, y sólo si, cada valor de* R.x *tiene asociado a él exactamente un valor de* R.y *para cualquier extensión de la relación* R.

Se puede apreciar lo siguiente en la definición anterior:

- El concepto de dependencia funcional tiene en cuenta a atributos de una misma relación y no a atributos de relaciones diferentes.
- El concepto de dependencia funcional hace referencia a la relación funcional que pueda existir entre parejas de atributos de una relación.
- El concepto de dependencia funcional es independiente del estado o extensión de una relación y, por lo tanto, es intrínseca a la intención de una relación.
- La definición anterior no hace una referencia explícita a la complejidad de los atributos dependientes, por lo que:
  - Los atributos funcionalmente dependientes de una relación pueden ser simples o estar formados por la agregación de varios atributos, *R.x* y/o *R.y* pueden ser de la forma:

    $$R.x \equiv \{R.a, R.b, ..., R.n\}$$

    para $R.a, R.b, ..., R.n \in R$.
  - Los atributos funcionalmente dependientes pueden ser, o no, atributos que formen parte de la clave primaria o de alguna clave candidata de la relación.
- En la definición anterior no se hace referencia alguna al número de veces (número de tuplas) en las que el atributo *R.x* tiene un mismo valor.

En la relación *Alumno* se puede observar que los atributos *curso* y *nota* son funcionalmente dependientes del atributo *matricula#*, puesto que para un valor dado de *matricula#* se tiene un valor asociado de los atributos *curso* y *nota*.

Igual razonamiento se puede realizar con los atributos *nombre* y *apellidos* con respecto al atributo *matricula#*. Ya que para un valor dado de *matricula#* se tiene un único valor asociado de *nombre* y *apellidos*.

Se puede apreciar que la definición anterior de dependencia funcional no impone la restricción de que no puedan repetirse los valores del atributo *R.y* para distintos valores del atributo *R.x*. Así, en la relación *Alumno*, se pueden presentar extensiones en las que para distintos valores del atributo *matricula#* (*R.x*) estén presentes los mismos valores para cualesquiera de los atributos dependientes funcionalmente (*R.y*), *curso, nota, nombre* y *apellidos*.

Si en la relación *Alumno* se considera que la agregación de los atributos *nombre* y *apellidos* puede ser una clave candidata de la relación, se observa que también existirá una dependencia funcional entre el atributo *matricula#* y el agregado *(nombre + apellidos)*[8]. Puesto que para un valor dado de *matricula#* siempre se tendrá asociado un único valor del agregado *(nombre + apellidos)*. Pero además, como el agregado *(nombre + apellidos)* es una clave candidata de la relación *Alumno*, también se cumple

[8] El signo + se ha utilizado para representar la agregación de atributos.

que para un valor dado del agregado *(nombre + apellidos)* se tendrá asociado siempre un único valor del atributo *matricula#*. Se dice en estos casos que existe una *dependencia funcional mutua* entre los atributos *R.x* y *R.y* de la relación *R*, y se representa de la forma $R.x \leftrightarrow R.y$.

Se puede introducir, en estos momentos, una definición algo más amplia de dependencia funcional, de la forma:

> *Dada una relación* R, *se dice que el atributo* R.y ∈ R *es funcionalmente dependiente de otro atributo* R.x ∈ R, *y se expresa en la forma* R.x → R.y *si, y sólo si, siempre que dos o más tuplas de* R *coincidan en sus valores de* R.x, *también, para esas tuplas, existirá una coincidencia en los valores del atributo* R.y.

En la definición de los esquemas relacionales es de vital importancia el estudio de las dependencias funcionales existentes en las relaciones, y sobre la base de este estudio se deben aplicar las reglas de normalización de las relaciones que forman parte del esquema.

Es conveniente la representación textual y gráfica de las dependencias funcionales existentes en las relaciones. Así, para el caso de la relación *Alumno* se aprecia que existen las siguientes dependencias que se representan de la forma:

Representación textual de las dependencias funcionales

Formato general

$$R.x \rightarrow (R.a, R.b, ... R.n)$$

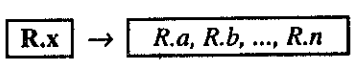
denotando que los atributos *R.a, R.b, ..., R.n* son funcionalmente dependientes del atributo *R.x*.

Relación *Alumno*

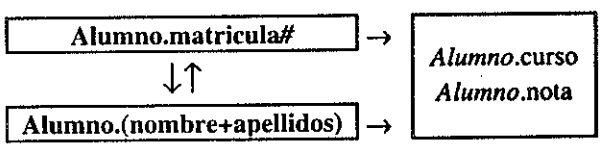
Alumno.*matricula#* → (Alumno.*nota*, Alumno.*curso*)
Alumno.*matricula#* → (Alumno.*nombre*, Alumno.*apellidos*)
Alumno.*matricula#* ↔ (Alumno.*(nombre + apellidos)*)

Representación gráfica de las dependencias funcionales

Formato general



Relación *Alumno*



Es conveniente introducir en estos momentos el concepto de *Dependencia funcional completa*, de la forma:

> *Se dice que el atributo* R.y ∈ R *es funcionalmente dependiente y de forma completa de otro atributo* R.x ∈ R *si, y sólo si, depende funcionalmente de* R.x *y no de ningún subconjunto de los atributos que formen parte del atributo* R.x.

Según esta definición se puede apreciar:

- Que si el atributo *R.x* es un agregado formado por la concatenación de varios atributos pertenecientes a la relación, entonces la dependencia funcional *R.x → R.y*, puede no ser completa, ya que puede existir una dependencia funcional de la forma *R.z → R.y, para R.z ⊂ R.x.*
- Que el atributo *R.y* sea simple o compuesto no tiene relevancia para que la dependencia funcional *R.x → R.y* sea completa o no.

En la relación *Alumno* se podría considerar que existe una dependencia entre el atributo *nota* y el agregado *(matricula#+curso)*, (Alumno.(*matricula#,curso*) → Alumno.*nota*); pero esta dependencia funcional no es completa puesto que el atributo *nota* depende también funcionalmente del atributo *matricula#*.

3.3.1.1. PROPIEDADES DE LAS DEPENDENCIAS FUNCIONALES

Las dependencias funcionales satisfacen una serie de propiedades que es conveniente conocer para el análisis de las relaciones del esquema relacional en las que se presentan:

Reflexiva: *Dados los atributos **a** y **b** de una relación* R, *para los que se cumple que* R.b ⊆ R.a *entonces, en la relación* R, *está presente una dependencia funcional de la forma* R.a → R.b.

Esta regla considera que todo conjunto de atributos de una relación es funcionalmente dependiente de sí mismo y de cualquiera de sus posibles agregados.

Aumento: *Dados los atributos **a** y **b** de una relación* R *en la que está presente la dependencia funcional* R.a → R.b, *entonces también estará presente la dependencia funcional* R.(a+c) → R.(b+c), *siendo **c** cualquier otro atributo que forme parte de la intención de la relación* R.

Transitiva: *Dados los atributos **a, b y c** de una relación* R *en la que están presentes las dependencias funcionales* R.a → R.b *y* R.b → R.c, *entonces también estará presente la dependencia funcional* R.a → R.c.

A estas tres reglas descritas se les denomina *Axiomas de Armstrong*, y de ellos se pueden deducir otro conjunto de propiedades que satisfacen las dependencias funcionales:

Unión: *Dados los atributos **a, b y c** de una relación* R *en la que están presentes las dependencias funcionales* R.a → R.b *y* R.a → R.c, *entonces también estará presente la dependencia funcional* R.a → R.(b+c).

Pseudo-transitiva: *Dados los atributos **a, b, c y d** de una relación* R *en la que están presentes las dependencias funcionales* R.a → R.b *y* R.(b+c) → R.d, *entonces también estará presente la dependencia funcional* R.(a+c) → R.d.

Descomposición: *Dados los atributos **a, b y c** de una relación* R *en la que está presente la dependencia funcional* R.a → R.b, *y se cumple que* R.c ⊆ R.b, *entonces también estará presente la dependencia funcional* R.a → R.c.

3.3.2. Reglas de normalización

La teoría de la normalización está basada en la aplicación de una serie de reglas a las que se les denomina *Reglas de normalización*. Se dice que una relación está en una determinada forma normal si satisface un cierto conjunto específico de restricciones impuestas por la regla de normalización correspondiente.

La aplicación de una regla de normalización es una operación que toma una relación como argumento de entrada y da como resultado dos o más relaciones, y:

- La relación objeto de la aplicación de la regla es desestimada en el nuevo esquema relacional considerado.
- No se introducen nuevos atributos en el esquema relacional resultante de la normalización.
- Los atributos de la relación objeto de la normalización pasan a formar parte de la intención de una o más de las relaciones resultantes.
- En la aplicación de la regla de normalización se ha debido eliminar, al menos, una dependencia existente entre los atributos de la relación objeto de la normalización.

De esta forma, la sucesiva aplicación de las reglas de normalización va a dar lugar a la generación de un número mayor de relaciones que formen parte del esquema relacional y, desde un punto de vista sólo lógico, una redundancia de los atributos considerados en el esquema.

La aplicación sucesiva de las reglas de normalización restringe, por tanto, el número de relaciones que las satisfacen. Por regla general, se dice que un esquema relacional es consistente si las relaciones satisfacen al menos la forma normal de *Boyce-Codd*.

3.3.2.1. LA PRIMERA FORMA NORMAL *FN1*

Una relación R *satisface la primera forma normal (FN1) si, y sólo si, todos los dominios subyacentes de la relación* R *contienen valores atómicos.*

La aplicación de esta regla es fácil y directa para cualquier relación, y este proceso se realiza de forma automática en el proceso de análisis del dominio del problema. Simplemente, consiste en descomponer aquellas tuplas en las que los atributos tengan más de un valor en tantas tuplas como valores estén presentes. De hecho, es una restricción innata al propio modelo relacional.

El que una relación se encuentre en *FN1* no es condición suficiente, aunque sí necesaria, para garantizar la consistencia del esquema relacional.

3.3.2.2. LA SEGUNDA FORMA NORMAL *FN2*

Una relación R *satisface la segunda forma normal (FN2) si, y sólo si, satisface la primera forma normal y cada atributo de la relación depende funcionalmente de forma completa de la clave primaria de esa relación.*

Para explicar la naturaleza de esta forma normal se va a analizar una nueva relación denominada *Matricula*, como se muestra a continuación:

Esquema-1
Matricula *(dni, asignatura#, apellidos, nombre, nota, curso, aula, lugar)*

en la que:

- El atributo *asignatura#* representa la identificación de las asignaturas en las que se encuentra matriculado cada uno de los alumnos.
- El atributo *aula* representa las aulas en las que se imparte la docencia de las asignaturas.
- El atributo *lugar* representa los lugares de estudio en los que se imparte la docencia correspondiente a las asignaturas.
- El atributo *curso* representa el curso en el que se imparte la docencia de una asignatura. Por tanto, se va a suponer que existe una dependencia funcional entre los atributos *asignatura#* y *curso*, de la forma: *Matricula.asignatura#* $\rightarrow$ *Matricula.curso*, que representa que si bien en un curso se puede impartir docencia para varias asignaturas, una asignatura está asignada a la docencia de un único curso.

Se puede apreciar que esta relación no se encuentra en *FN2*, puesto que existe una dependencia funcional no completa entre atributos de la relación que no forman parte de la clave (atributos no primos) y la clave de la relación[9].

La relación *Matricula* presenta una serie de inconvenientes debidos a que no satisface la *FN2*. Inconvenientes que se ponen de manifiesto en:

Inserción de tuplas: se aprecia que no se pueden conocer las asignaturas que se imparten en un curso hasta que no exista algún alumno matriculado en esas asignaturas. Por ejemplo, si la asignatura '4' se impartiera en el curso '3', esta información no podría ser conocida hasta que algún alumno se hubiera matriculado en esta asignatura. Debido, simplemente, a que en caso contrario se violaría la integridad de la clave al existir valores nulos en la misma.

Borrado de tuplas: de igual forma se aprecia que si se eliminan las tuplas correspondientes a los alumnos matriculados en una asignatura, se pierde la información que representa que una asignatura forma parte de la docencia de un curso.

Actualización de tuplas: si se realiza un cambio de asignación de curso para una asignatura, sería necesario actualizar todas las tuplas correspondientes a todos los alumnos matriculados en esa asignatura. Esto es debido a que existe una gran redundancia de información, puesto que en la relación para cada alumno de una asignatura se almacena el curso en que se imparte, mientras que esta información es independiente del alumno que está matriculado en esa asignatura, dependiendo únicamente de la asignatura.

Para eliminar todos estos inconvenientes es necesario realizar un proceso de descomposición de la relación *Matricula* en dos nuevas relaciones, las cuales satisfacen la segunda forma normal. Estas relaciones quedarían con la siguiente estructura[10]:

Esquema-2	
Imparte	*(asignatura#, curso)*
Matricula-2	*(dni, asignatura#, apellidos, nombre, nota, aula, lugar)*

Se aprecia que:

[9] A los atributos que forman parte de la clave de una relación se les denomina atributos *primos*, y a los que no forman parte de la clave, atributos *no primos*.

[10] En los esquemas se representarán: *los atributos que forman la clave primaria de las relaciones* subrayados, *los atributos que forman la(s) clave(s) alterna(s) de las relaciones* con doble subrayado, y *los atributos que son claves foráneas de otras relaciones* en negrita, con independencia de que sean, a su vez, claves primarias o alternas de la relación de la que forman parte.

- La relación *Imparte*, se encuentra en *FN2*. La clave de esta relación es el atributo *asignatura#*, y el único atributo no primo (*curso*) depende funcionalmente de forma completa de la clave de la relación.
- Al eliminarse el atributo *curso* en la relación *Matricula-2*, se ha eliminado la dependencia funcional no completa entre el atributo *curso* y la clave de la relación (la existente entre los atributos *asignatura#* y *curso*) y, por lo tanto, los problemas que causaba esta dependencia en los procesos de mantenimiento de la información para esta relación.
- No se ha producido pérdida de información, puesto que el atributo *asignatura#* debe definirse como clave foránea de la relación *Imparte*. De esta forma, el atributo *asignatura#* en la relación *Matricula-2* para cualquier tupla, sólo podrá tomar valores existentes en alguna tupla de la relación *Imparte*. Así, basándose en esta referencia podrá conocerse en cada momento en qué curso es impartida cada asignatura para cada uno de los alumnos matriculados.

El mismo razonamiento debe hacerse para la dependencia funcional no completa existente entre la clave de la relación *Matricula-2* y el agregado *(apellidos + nombre)*, puesto que este agregado sólo depende del atributo *dni* —el identificador del alumno— y no de las asignaturas —valores del atributo *asignatura#*— en las que se encuentra matriculado. Si se aplica, por tanto, a la relación *Matricula-2* la *FN2* para eliminar esta dependencia, se obtiene el siguiente esquema:

Esquema-3	
Imparte	*(asignatura#, curso)*
Alumno-2	*(dni, apellidos, nombre)*
Matricula-3	**(dni, asignatura#**, *nota, aula, lugar)*

3.3.2.3. LA TERCERA FORMA NORMAL *FN3*

> ***Una relación*** **R** ***satisface la tercera forma normal (FN3) si, y sólo si, satisface la segunda forma normal y cada atributo no primo de la relación no depende funcionalmente de forma transitiva de la clave primaria de esa relación. Es decir, no pueden existir dependencias entre los atributos que no forman parte de la clave primaria de la relación*** **R**.

Se puede observar que la relación *Matricula-3*, la cual se encuentra en *FN2*, sigue presentando problemas en los procesos de manipulación de la misma. Los problemas son debidos a que existe una dependencia entre los atributos *aula* y *lugar*.

Si se considera, como es lógico, que cada aula se encuentra ubicada físicamente en un único lugar, y que la docencia de una determinada asignatura —para un determinado conjunto de alumnos; es decir, diferentes alumnos matriculados en una asignatura pueden recibir docencia en aulas diferentes (grupos de clase)— se imparte en una única aula, se observa que existe una dependencia funcional entre los atributos no primos *lugar* y *aula*, además de las dependencias funcionales completas entre los atributos *lugar* y *aula* con la clave de la relación *Matricula-3*.

La existencia de estas dependencias entre los atributos no primos ocasiona problemas en la manipulación de la relación *Matricula-3*, problemas en:

Inserción de tuplas: se observa que no se pueden conocer las aulas de cada uno de los lugares utilizadas para la docencia de asignaturas hasta que no haya algún alumno matriculado en esa asignatura.

Borrado de tuplas: una vez borrados todos los alumnos matriculados en una asignatura, se pierde la información correspondiente a las aulas de cada lugar utilizadas para la docencia.

Modificación de tuplas: existe una gran redundancia de información puesto que se almacena el lugar en el que se encuentra ubicada un aula para cada uno de los alumnos que cursan una asignatura que se imparte en dicha aula.

Estos problemas se deben, no a la presencia de una dependencia funcional no completa, sino a la presencia de una dependencia funcional transitiva. La relación *Matricula-3* presenta las siguientes dependencias:

Matricula-3.(*dni, asignatura#*) → Matricula-3.*aula*
Matricula-3.(*dni, asignatura#*) → Matricula-3.*lugar*
Matricula-3.(*aula*) → Matricula-3.*lugar*

Y, como se observa, el atributo *lugar* es dependiente de la clave de la relación de forma transitiva, debido a que es dependiente funcionalmente de otro atributo no primo (*aula*), el cual depende también de la clave de la relación.

Para eliminar los problemas que ocasiona en la relación *Matricula-3* la existencia de esta dependencia funcional, esta relación debe descomponerse en dos relaciones, quedando el esquema de la forma:

Esquema-4	
Imparte	*(asignatura#, curso)*
Alumno-2	*(dni, apellidos, nombre)*
Ubicacion	*(aula, lugar)*
Matricula-4	**(dni, asignatura#**, *nota,* **aula)**

Donde se observa que la relación *Matricula-4* se encuentra en *FN3*, puesto que todas las dependencias funcionales existentes son completas y entre atributos no primos y la clave de la relación.

Se ha generado una nueva relación, la relación *Ubicacion*, para la cual el atributo *aula* es clave, y tiene un único atributo no primo, *lugar*, el cual depende funcionalmente de la clave, por lo que también se encuentra en *FN3*.

De nuevo, el atributo *aula* de la relación *Matricula-4* deberá definirse, en el esquema, como clave foránea de la relación *Ubicacion*, de forma que para cualquier

tupla de la relación *Matricula-4* este atributo sólo pueda tomar valores nulos o bien igual a valores existentes en alguna tupla de la relación *Ubicacion*.

3.3.2.4. LA FORMA NORMAL DE BOYCE-CODD *FNBC*

La forma normal *FNBC* es conceptualmente distinta, y mucho más sencilla, a la segunda y tercera forma normal. Las relaciones que satisfacen la restricción impuesta por la forma normal *FNBC* satisfacen la *FN2* y *FN3*. La *FNBC* se basa en el concepto de *Determinante funcional*, y está soportada en las características de las claves candidatas de las relaciones.

> *Se denomina determinante funcional a uno o un conjunto de atributos de una relación* R *del cual depende funcionalmente de forma completa algún otro atributo de la misma relación.*

Se puede expresar la forma normal de *Boyce-Codd* de la forma:

> *Una relación* R *satisface la forma normal de Boyce-Codd (FNBC) si, y sólo si, se encuentra en FN1, y cada determinante funcional es una clave candidata de la relación* R.

Del análisis del *Esquema-4* se puede deducir que todas las relaciones se encuentran en *FNBC*, puesto que:

- Los únicos determinantes funcionales son las claves para cada una de las relaciones, puesto que en ninguna relación existen claves alternativas.
- En todas las relaciones, las únicas dependencias funcionales son las existentes entre los atributos no primos de cada relación y la clave de la misma.

Pero se pueden dar casos en los que una relación se encuentre en *FN3* pero no satisfaga la *FNBC*, puesto que la *FNBC* es más restrictiva que la *FN3*.

Por ejemplo, consideremos de nuevo el *Esquema-1*, pero en este caso se va a realizar la consideración de que el agregado *apellido, nombre, asignatura#* es una clave candidata de la relación *Matricula*, quedando el esquema siguiente:

Esquema-5	
Matricula-5	*(dni, asignatura#, apellidos, nombre, nota, curso, aula, lugar)*

clave ← claves candidatas

En la cual existen dos determinantes funcionales, cada uno de ellos compuesto y formado por los agregados: *dni, asignatura#* y *apellidos, nombre, asignatura#*, para lo cual es necesario suponer que no existen dos alumnos con el mismo nombre completo matriculados en la misma asignatura.

Las dependencias existentes en la relación *Matricula-5* son:

Matricula-5.*asignatura#* → Matricula-5.*curso*
Matricula-5.*(dni, asignatura#)* → Matricula-5.*aula*

Matricula-5.*(dni, asignatura#)* → Matricula-5.*lugar*
Matricula-5.*(dni, asignatura#)* → Matricula-5.*nota*
Matricula-5.*(apellidos, nombre, asignatura#)* → Matricula-5.*aula*
Matricula-5.*(apellidos, nombre, asignatura#)* → Matricula-5.*lugar*
Matricula-5.*(apellidos, nombre, asignatura#)* → Matricula-5.*nota*
Matricula-5.*aula* → Matricula-5.*lugar*
Matricula-5.*(dni, asignatura#)* ↔ Matricula-5.*(apellidos, nombre, asignatura#)*

Como se puede observar existen dependencias entre atributos que no son determinantes funcionales y que es necesario eliminar. Por tanto, la relación *Matricula-5* debe ser descompuesta, quedando el nuevo esquema de la forma:

Esquema-6	
Imparte	*(asignatura#, curso)*
Ubicacion	*(aula, lugar)*
Matricula-6	*(dni, asignatura#, apellidos, nombre, nota,* **aula***)*

Se puede observar que, sobre la base de la definición de la *FNBC*, en el *Esquema-6* todas las relaciones se encuentran en *FN3*, simplemente por la consideración de los determinantes funcionales y las dependencias que otros atributos de la relación mantienen con ellos. Se han eliminado directamente las dependencias funcionales no completas y las dependencias transitivas.

Ahora bien, las relaciones *Imparte* y *Ubicacion* se encuentran además en *FNBC*, pues sólo existe un determinante funcional y un atributo dependiente del mismo de forma completa, pero la relación *Matricula-6*, aunque se encuentra en la *FN3*, no satisface la *FNBC*.

Se puede observar que en esta relación existe una dependencia funcional que no ha sido considerada hasta el momento. Se trata de la dependencia funcional mutua existente entre el atributo *dni* y el agregado *(apellidos + nombre)*.

Matricula-6.*dni* ↔ Matricula-6.*(apellidos, nombre)*

En las dos claves candidatas está presente un atributo común, el atributo *asignatura#*, el cual forma parte de los dos determinantes funcionales, por lo que ambas claves se encuentran traslapadas[11]. La presencia de claves traslapadas puede ocasionar serios problemas en el mantenimiento de la información que hay que resolver.

La relación *Matricula-6* sería conveniente descomponerla en dos relaciones que satisfagan la *FNBC*, quedando el nuevo esquema de la forma:

[11] Dos claves candidatas se dice que están traslapadas si cada una de ellas está formada por dos o más atributos y alguno de ellos es común a ambas.

Esquema-7	
Imparte	*(asignatura#, curso)*
Ubicacion	*(aula, lugar)*
Alumno-3	*(dni, apellidos, nombre)*
Matricula-7	**(dni, asignatura#**, *nota, aula)*

que, como se puede observar, es similar al *Esquema-4* obtenido previamente por aplicación de las formas normales *FN1* a *FN3*. Si bien en este caso al considerar el agregado formado por los atributos *(apellidos+nombre)* como identificador alternativo de los alumnos, será necesario definir en el esquema a este agregado como clave alterna de la relación *Alumno-3*, para así hacer cumplir la dependencia funcional mutua existente entre este agregado y el atributo *dni*.

3.3.2.5. EL PROCESO DE DESCOMPOSICIÓN

Se ha podido observar a lo largo de las secciones anteriores que para que una relación satisfaga una determinada forma normal es necesario descomponerla en al menos dos nuevas relaciones, las cuales satisfacen la regla de normalización para la cual se realiza la descomposición.

El proceso de descomponer una relación no es un proceso trivial y debe ser realizado siguiendo los siguientes principios:

Descomposición por aplicación de la *FN2*: Dada una relación R cuya clave es un agregado de la forma $(R.x, R.y)$, y en esta relación existe una dependencia funcional incompleta de la forma $R.y \rightarrow R.z$, donde $R.z$ es un atributo no primo de la relación R, el proceso de descomposición se realizará de la forma siguiente:

- De la relación R se elimina el atributo $R.z$, quedando igual el resto de su intención.
- Se construye una nueva relación R_1 cuya intención es:
  1. El atributo $R.z$ como atributo no primo de la relación R_1.
  2. El atributo $R.y$ como atributo primo (clave primaria) de la relación R_1.
- Se define a $R.y$ como clave foránea de la clave de la relación R_1 $(R_1.y)$.

Luego la descomposición que se debe realizar para la aplicación de la *FN2* se debe hacer en base a la dependencia funcional incompleta, y nunca por cualquier otra dependencia entre atributos de la relación. En este proceso tanto $R.x$, $R.y$, $R.z$ pueden ser a su vez atributos simples o agregados de datos.

Descomposición por aplicación de la *FN3*: Dada una relación R de clave $R.x$ y dos atributos no primos $R.y$ y $R.z$, y en esta relación están presentes las siguientes dependencias funcionales: $R.x \rightarrow R.y$, $R.y \rightarrow R.z$ y, por tanto la dependencia transitiva, $R.x \rightarrow R.z$, el proceso de descomposición se realizará de la forma siguiente:

- De la relación R se elimina el atributo que mantiene una dependencia funcional transitiva con la clave de la relación; es decir, el atributo $R.z$, quedando igual el resto de la intención de la relación R.
- Se construye una nueva relación R_1 cuya intención es:
  1. El atributo $R.z$ que mantenía una dependencia funcional transitiva.
  2. El atributo $R.y$ con el cual el atributo $R.z$ mantenía una dependencia funcional completa.
  3. La clave de la relación R_1 será el atributo $R_1.y$ y, en esta relación sólo existirá una dependencia funcional completa de la forma: $R_1.y \rightarrow R_1.z$.
- En la relación R se define $R.y$ como clave foránea de la relación R_1 $(R_1.y)$.

Luego la descomposición que se debe realizar por aplicación de la *FN3* se debe hacer basándose en la dependencia funcional transitiva y nunca por cualquier otra dependencia entre los atributos de la relación. En este proceso tanto $R.x$, $R.y$, $R.z$ pueden ser a su vez atributos simples o agregados de datos.

Para aclarar este proceso vamos a ver de nuevo el *Esquema-3* en el cual está presente una dependencia funcional transitiva. La relación *Matricula-3* puede ser sometida a tres descomposiciones diferentes, como se muestra en la tabla 3.2, y analicemos cada una de las descomposiciones posibles:

Descomposición-1	
R-1	*(aula, lugar)*
M-1	*(dni, asignatura#, nota,* **aula***)*
Descomposición-2	
R-2	*(dni, asignatura#, nota, aula)*
M-2	**(dni, asignatura#**, *lugar)*
Descomposición-3	
R-3	*(aula, lugar)*
M-3	*(dni, asignatura#, nota, lugar)*

Tabla 3.2 Ejemplos de descomposición de relaciones

1. En primer lugar, la *Descomposición-3* es inviable pues da lugar a una pérdida de información. Se puede observar que si se realiza esta descomposición se pierde la información correspondiente al aula en la que se imparte la docencia de una asignatura para un grupo de alumnos. Se conocerá el lugar (véase la relación *M-3*) pero no el aula de ese lugar, aunque en la relación *R-3* se mantenga la información correspondiente a la ubicación de cada una de las aulas.
2. Aunque la *Descomposición-2* no ocasiona una pérdida de información, tiene los siguientes problemas:

- Siguen estando presentes los problemas existentes inicialmente para el mantenimiento de la información debido a que para conocer la ubicación de cada una de las aulas es necesario que existan alumnos matriculados en las asignaturas que se impartirán en esas aulas.
- Además, la información correspondiente entre las aulas y los lugares se representa de forma independiente, perdiéndose el significado semántico de la dependencia funcional entre aula y lugar *(Matricula-3.aula → Matricula-3.lugar)*. Este hecho hace que se tengan que poner en práctica procedimientos especiales en los procesos de mantenimiento de las tablas *R-2* y *M-2* para representar esta dependencia.
- Y, también, introduce una redundancia excesiva al duplicarse la información correspondiente al *dni* y *asignatura#* para cada una de las tablas. Esta redundancia podrá ocasionar inconsistencias graves. Por ejemplo, la existencia de información correspondiente a un alumno para una asignatura con su nota y con aula asignada (una tupla de *R-2*) y, sin embargo, que no tenga un lugar para recibir la docencia (ninguna tupla en *M-2*).

Debido a ello esta descomposición es también incorrecta.

3. Por último, se aprecia que la *Descomposición-1* es la acertada, pues no da lugar ni a pérdida de información ni a pérdida de la semántica del problema. Esto es debido a que la *Descomposición-1* se ha realizado en base a la dependencia funcional transitiva, mientras las otras se han realizado en base a dependencias funcionales no transitivas.

Se ha podido observar, con este ejemplo, que para que una descomposición sea correcta, las nuevas relaciones que se generen deben de ser independientes, no se debe producir una pérdida de información y, para mantener la integridad de la información, se deberán definir nuevas claves foráneas entre los atributos de la relación original (en su nueva intención) y las claves de las nuevas relaciones construidas.

Descomposición por aplicación de la *FNBC*: Dada una relación R en la cual están presentes uno o más determinantes funcionales, *R.x, R.y, ..., R.z*, posiblemente formados por un agregado de datos de la forma: $R.[] \equiv \{R.a, R.b, ..., R.j\}$, y en esta relación existen dependencias funcionales distintas a dependencias completas de la forma $R.[] \rightarrow R.n$, donde $R.n$ es cualquier atributo de la relación R, el proceso de descomposición se realizará de la forma siguiente:

- De la relación R se elimina el atributo $R.n$, quedando igual el resto de la intención de la relación R.
- Se construye una nueva relación R_i cuya intención es:
  1. El atributo $R.n$ como atributo no primo de la nueva relación R_i.
  2. El atributo $R.m$ como atributo primo (clave primaria), donde $R.m \in R.[]$, $R.m \subset R.[]$ está formado por un subconjunto de los agregados de

datos que forman parte del determinante funcional $R.[]$, y por el que existe una dependencia funcional de la forma $R.m \rightarrow R.n$.

- Se define a $R.m$ como clave foránea de la clave de la relación R_i $(R_i.m)$.

Este proceso se debe realizar para cada uno de los casos de dependencias funcionales no completas entre los atributos de la relación R y los determinantes funcionales de la misma, incluidos, por supuesto, los atributos que forman parte de los determinantes funcionales cuando éstos están formados por agregados de datos (traslapación de claves).

Se puede observar que el procedimiento es análogo a la aplicación de las descomposiciones que se realizan en la *FN2* y *FN3*, si bien en este caso se tiene:

- Las descomposiciones correspondientes a la *FN2* y *FN3* se aplican en un solo paso sin hacer referencia al tipo de dependencia funcional.
- Se tienen en cuenta las dependencias entre los atributos que forman parte de las claves candidatas (determinantes funcionales), eliminándose las mismas por el proceso de descomposición.

3.4. OTROS TIPOS DE DEPENDENCIAS

Se han considerado hasta este momento las dependencias que pueden estar presentes en una relación R y por las cuales los valores que toma un atributo $R.x$ de esta relación determinan, para cada valor, el valor que puede tomar otro atributo R.y de la misma relación R; es decir, dependencias funcionales de la forma $R.x \rightarrow R.y$.

Pero pueden considerarse otros tipos de dependencias funcionales en las que el valor de $R.x$ no tenga que determinar un único valor de $R.y$, sino un conjunto bien definido de posibles valores que, a su vez, pueda depender o no de los valores que tome otro atributo $R.z$ de la misma relación R.

La existencia de este tipo de dependencias ocasiona problemas de redundancia y de manipulación de la información que mantiene la relación R y, por tanto, es necesario aplicar alguna regla que elimine estas dependencias. Éste es el fin de las reglas de normalización cuarta y quinta.

Si bien estos nuevos tipos de dependencias pueden existir entre cualesquiera de los atributos de una relación R, es más usual que estén presentes cuando los atributos implicados forman parte de las claves candidatas de la relación (determinantes funcionales). La razón es bien sencilla, si son otros atributos los implicados, por aplicación de las reglas de normalización estudiadas hasta el momento, estas dependencias han debido ser eliminadas y, sin embargo, como se ha podido observar, las reglas *FN1* a *FNBC* no tienen en cuenta las dependencias existentes entre los atributos que forman parte de las claves candidatas de una relación cuando estas claves están formadas por un agregado de datos.

3.4.1. La cuarta forma normal *FN4*

La regla de normalización *FN4* está basada en la eliminación de las relaciones de un tipo especial de dependencias denominadas *Dependencias multivaluadas* (*DMV*), las cuales fueron consideradas por primera vez por *Fagin* y *Zamiolo* en 1977, y que se pueden definir de la forma siguiente:

> *Dada una relación* R, *se dice que el atributo* R.y ∈ R *depende de forma multivaluada de otro atributo* R.x ∈ R, *o que* R.x *multidetermina a* R.y, *y se expresa de la forma* R.x →→ R.y *si, y sólo si, cada valor de* R.x *tiene asignado un conjunto bien definido de valores de* R.y *y este conjunto es independiente de cualquier valor que tome otro atributo* R.z ∈ R, *el cual depende del valor de* R.x.

Puede apreciarse en la definición de dependencia multivaluada que para que este tipo de dependencia esté presente en una relación, ésta, al menos, debe estar formada por tres atributos.

Las dependencias multivaluadas representan la independencia existente entre dos conjuntos *R.y* y *R.z*, la cual está correlacionada por la dependencia que tiene cada uno de estos conjuntos con el conjunto *R.x* del cual dependen ambos de forma multivaluada.

Así, es fácil demostrar que para que en una relación *R* esté presente la dependencia multivaluada *R.x* →→ *R.y* es condición necesaria que además esté presente la dependencia multivaluada *R.x* →→ *R.z*. Las dependencias multivaluadas nunca están presentes solas en una relación, sino que siempre están presentes en parejas, por lo que es usual representarlas de la forma: *R.x* →→ *R.y/R.z*, significando que: para cada valor de *R.x* no existe un único valor de *R.y* asociado (se trataría de una dependencia funcional), sino un conjunto bien definido de valores independientes del valor de *R.z*, el cual sólo depende del valor de *R.x*.

Para aclarar el concepto de dependencia multivaluada vamos a considerar la siguiente relación:

Docencia (*dni, asignatura#, aula*)

La relación *Docencia* está formada por tres atributos: *dni*, *asignatura#* y *aula*, y los tres forman parte de la clave de la relación (la única clave candidata, pues no hay más atributos presentes en la relación). Y se van a suponer, aunque no de forma muy realista, los siguientes supuestos en el problema representado por la relación *Docencia*:

- Cada alumno, representado por el valor del atributo *dni*, puede estar matriculado en un conjunto de asignaturas.
- Sobre la base del valor del atributo *dni* (podría considerarse también el nombre y apellidos de los alumnos para hacer los grupos), al alumno se le asigna un conjunto de aulas en las cuales recibirá la docencia con independencia de las asignaturas en las que esté matriculado o no.

- Cada asignatura puede ser impartida en cualquiera de las aulas disponibles, con independencia de que haya alumnos matriculados en las mismas.

Se observa, en la relación *Docencia*, que existe una independencia entre:

- Las aulas en las que se les imparte docencia a los alumnos y las asignaturas, puesto que (aunque no es un hecho muy real) los alumnos tienen asignado un conjunto de aulas para la impartición de la docencia con independencia de las asignaturas que cursan, pues tan sólo depende del alumno (valor del dni).
- Las asignaturas que cursa cada alumno y el aula en la que se les imparte la docencia, puesto que los alumnos se pueden matricular en cualquier conjunto de asignaturas con independencia del aula en la que se impartan.

Estas independencias existen porque en la relación *Docencia* se ha considerado que se puede impartir cualquier asignatura en cualquier aula y que cualquier alumno puede acudir a cualquier aula de las asignadas para recibir la docencia de cualquier asignatura de las que está matriculado.

Así, puede observarse que en la relación *Docencia* están presentes las siguientes dependencias multivaluadas:

Docencia.*dni* →→ Docencia.*asignatura#*
Docencia.*dni* →→ Docencia.*aula*

La existencia de estas dependencias multivaluadas ocasiona problemas en el mantenimiento de la información representada por la relación *Docencia*, problemas debidos a:

- La relación *Docencia* presenta mucha redundancia, puesto que, por ejemplo, como un alumno puede recibir la docencia en cualquier aula independientemente de la asignatura no es necesario considerar la información de la asignatura para cada pareja de valores (*dni, aula*). El mismo razonamiento se puede hacer para las parejas de valores (*dni, asignatura*) con respecto a las aulas.
- Otros problemas en los procesos de inserción, borrado y modificación, al formar los tres atributos parte de la clave primaria, eliminándose las independencias propuestas en el problema. Por ejemplo, aunque el aula en la que un alumno recibe la docencia de las asignaturas que cursa sea independiente de las asignaturas, no se puede conocer hasta que no se matricule en esas asignaturas.

Todos estos inconvenientes se resuelven descomponiendo la relación *Docencia* en dos nuevas relaciones, según se muestra en el *Esquema-8*:

Esquema-8	
Docencia-1	(*dni, asignatura#*)
Docencia-2	(*dni, aula*)

Ambas relaciones *Docencia-1* y *Docencia-2* se encuentran ahora en *FN4*, pues no están presentes dependencias multivaluadas. Cada relación está formada por dos atributos y ambos forman parte de la clave primaria de la misma. Se puede ahora definir la *FN4* de la forma siguiente:

> *Una relación* R *está en FN4 si, y sólo si, siempre que exista una dependencia multivaluada en* R *de la forma* R.x →→ R.y, *todos los demás atributos de R son funcionalmente dependientes de* R.x (R.x → R.z, ∀R.z ∈ R).

Es decir, para que una relación satisfaga la *FN4*, las únicas dependencias admitidas son las dependencias funcionales de la forma *R.x → R.z* donde *R.x* es un determinante funcional de *R*, y *R.z* es cualquier otro atributo de *R*.

Obsérvese que las dependencias multivaluadas son una generalización de las dependencias funcionales o, lo que es lo mismo, una dependencia funcional es un caso particular de una dependencia multivaluada en la que se considera en lugar de un conjunto bien definido de valores a un solo valor.

3.4.1.1. PROPIEDADES DE LAS DEPENDENCIAS MULTIVALUADAS

Las dependencias multivaluadas, además de satisfacer los *Axiomas de Armstrong* (una dependencia funcional es un caso particular de una dependencia multivaluada), satisfacen la siguiente serie de propiedades:

Repetición: *Dada una relación* R *en la que está presente la dependencia funcional* R.a → R.b, *entonces también estará presente la dependencia multivaluada* R.a →→ R.b.

Complementación: *Dada una relación* R *en la que está presente la dependencia multivaluada* Ra. →→ R.b, *entonces se cumple* R.a →→ R-R.b-R.a, *siendo b cualquier otro atributo de la relación* R.

Aumento multivaluado: *Dada una relación* R *en la que está presente la dependencia multivaluada* R.a →→ R.b, *y en la que están presentes dos atributos c, d ∈ R, para los que se cumple que* R.c ⊆ R.d, *entonces también está presente en* R *la dependencia multivaluada* R.(a+d) →→ R.(b+c).

Transitiva multivaluada: *Dada una relación* R *en la que están presentes las dependencias multivaluadas* R.a →→ R.b *y* R.b →→ R.c, *entonces también estará presente la dependencia multivaluada* R.a →→ R.(c-b).

Condensación: *Dada una relación* R *en la que está presente la dependencia multivaluada* R.a →→ R.b, *y en la que se cumple que* R.c ⊆ R.b, *y existe la dependencia funcional* R.d → R.c, *para un atributo d, tal* que d ⊆ R, *y* R.d∩R.b = ∅ *entonces, en la relación* R, *está presente la dependencia funcional* R.a → R.c.

En base a estas reglas se pueden deducir las siguientes tres reglas:

Unión: *Si en una relación* R *están presentes las dependencias multivaluadas* R.a →→ R.b y R.a →→ R.c, *entonces también está presente la dependencia multivaluada* R.a →→ R.(b+c).

Intersección: *Si en una relación* R *están presentes las dependencias multivaluadas* R.a →→ R.b *y* R.a →→ R.c, *entonces también está presente la dependencia multivaluada* R.a →→ R.(b∩c).

Diferencia: *Si en una relación* R *están presentes las dependencias multivaluadas* R.a →→ R.b *y* R.a →→ R.c, *entonces también están presentes las dependencias multivaluadas* R.a →→ R.(b-c) *y* R.a →→ R.(c-b).

3.4.2. La quinta forma normal *FN5*

A lo largo de estas secciones se ha podido comprobar que para que una relación satisfaga una determinada forma normal ha sido necesario, en todos los casos, descomponerla en dos nuevas relaciones, cada una de las cuales satisface la forma normal por la cual se aplicó la descomposición.

Hay ocasiones en las que la descomposición de una relación en dos nuevas relaciones no puede realizarse sin pérdida de información y, sin embargo, es necesario realizar un proceso de normalización de una relación dada, pues están presentes una serie de dependencias que ocasionan, al menos, problemas de redundancias superfluas en esa relación.

En una relación *R* puede estar presente un caso especial de dependencias llamadas *Dependencias de Reunión*, las cuales son la base de la aplicación de la *FN5*. Las dependencias de reunión son difíciles de detectar y, por tanto, la aplicación de esta regla de normalización no es muy usual, además de cuestionarse la mejora que puede introducir la aplicación de la misma si se tiene en cuenta la complejidad que añade al nuevo esquema relacional por el número de relaciones nuevas que introduce.

> *Dada una relación* R *de esquema* R(a_1, a_2, ..., a_n) *se dice que existe una dependencia de reunión si, y sólo si, la relación* R *puede ser construida a partir de la reunión natural de las relaciones* R_1, R_2, ..., R_n *obtenidas por la proyección de* R *sobre los atributos* a_1, a_2, ..., a_n, *respectivamente.*

Es decir, dada una relación *R*, de esquema: $R(a_1, a_2, ..., a_n)$, y dado un conjunto de relaciones $R_1, R_2, ..., R_n$ obtenidas de la forma:

$$R_1 = PROJECT(R/a_1) = \bullet\ (R/a_1)$$
$$R_2 = PROJECT(R/a_2) = \bullet\ (R/a_2)$$
$$... = ... = ...$$
$$R_n = PROJECT(R/a_n) = \bullet\ (R/a_n)$$

se cumple que la relación *R* puede ser obtenida en base a la reunión de las relaciones $R_1, R_2, ..., R_n$ de la forma siguiente:

$$R=R_1|x|R_2|x|.....|x|R_n$$

o, lo que es lo mismo:

$$R=\Pi\ (R/a_1)|x|\ \Pi\ (R/a_2)|x|.....|x|\ \Pi(R/a_n)$$

Puede ser definida en estos momentos la regla de normalización *FN5* de la forma siguiente:

> *Una relación* R *satisface la FN5, también denominada forma normal de proyección (FNPR) si, y sólo si, toda dependencia de reunión en* R *está implicada por las claves candidatas entre sí y no por cualquier otro atributo de* R, *forme o no parte de las claves candidatas.*

Para explicar más claramente este tipo de dependencia vamos a detenernos en la relación *Docencia*, presentada en la sección anterior

Docencia *(dni, asignatura#, aula)*

Pero, en este caso se va suponer que el problema que representa esta relación es el siguiente:

- Que un alumno puede estar matriculado en un conjunto de asignaturas.
- Que para cada asignatura existen una serie de aulas en las que se puede impartir la docencia.
- Que para cada asignatura el alumno recibe la docencia en todas las aulas asignadas a esa asignatura.

Se está suponiendo, en este caso, que a cada asignatura se le asigna una serie de aulas para impartir la docencia y que, por tanto, cuando el alumno se matricula en una asignatura va a recibir la docencia para esa asignatura en todas las aulas habilitadas para la misma y que se puede matricular en cualquier conjunto de asignaturas.

En este caso, aunque la relación *Docencia* se encuentra en *FNBC* y además en *FN4* (no existen dependencias que violen estas reglas, pues ahora se ha considerado que las aulas no son independientes de las asignaturas en las que se matriculan los alumnos), se observa que si existe dependencia de reunión entre los atributos *dni*, *asignatura#* y *aula*, esta dependencia da lugar a que exista una gran redundancia en esta relación.

Esta relación, *Docencia*, puede ser descompuesta, sin pérdida de información, en tres relaciones dando lugar al siguiente esquema:

Esquema-9	
Docencia-1	*(dni, asignatura#)*
Docencia-2	*(asignatura#, aula)*
Docencia-3	*(dni, aula)*

Las tres relaciones se encuentran ahora en *FN5* sin que exista ninguna dependencia de reunión trivial. Además, se puede comprobar fácilmente que si se realiza la reunión de las relaciones *Docencia-1*, *Docencia-2* y *Docencia-3*, se obtiene de nuevo la relación *Docencia*:

Docencia = Docencia-1 |x| *Docencia-2* |x| *Docencia-3*

El nuevo *Esquema-9* obtenido representa mucho más claramente el problema real en el que:

- Un alumno está matriculado en una serie de asignaturas, y en una asignatura pueden estar matriculados varios alumnos. Representado por la relación *Docencia-1*.
- A cada asignatura se le asigna un conjunto de aulas para la impartición de la docencia, y en un aula se puede impartir la docencia de varias asignaturas. Representado por la relación *Docencia-2*.
- Cada alumno recibe la docencia en un conjunto de aulas, y en cada aula se imparte la docencia para un conjunto de alumnos. Representado por la relación *Docencia-3*.

Ahora bien, el problema de la dependencia de reunión sigue estando presente aunque se haya normalizado la relación *Docencia* y va a ser necesario el establecimiento de una serie de procedimientos paralelos en el mantenimiento de las nuevas relaciones obtenidas para representar fielmente las restricciones establecidas en el ejemplo.

Por ejemplo, si se establece (y, por lo tanto, se insertan las tuplas correspondientes en la relación *Docencia-2*) que la asignatura *001* va a ser impartida en las aulas *A01* y *A02*, y se matricula un alumno en esta asignatura (se inserta la tupla correspondiente en la relación *Docencia-1*), será necesario insertar dos tuplas en la relación *Docencia-3*, las dos correspondientes a ese alumno y cada una de ellas haciendo referencia a las dos aulas en las que se imparte la docencia para la asignatura *001*. Esta operación será necesaria, pues la restricción impuesta al ejemplo —por la que existe la dependencia de reunión— es que los alumnos matriculados en una asignatura reciben docencia en todas las aulas asignadas a esa asignatura.

Vamos a ver esta dependencia de reunión con una extensión de la relación *Docencia*. Supongamos la siguiente extensión de esta relación:

Docencia		
dni	***asignatura#***	***aula***
30.111.222	001	A02
30.111.222	002	A01
30.666.777	001	A01

Y vamos a suponer que se ha insertado la tupla *<30.666.777, 001, A01>*. Si existe una dependencia de reunión entre los atributos *dni*, *asignatura#* y *aula*, la reunión de la proyección de la relación *Docencia* sobre estos atributos debe generar la relación *Docencia*. Vamos a comprobarlo:

Docencia-1	
dni	***asignatura#***
30.111.222	001
30.111.222	002
30.666.777	001

Docencia-2	
asignatura#	***aula***
001	A02
002	A01
001	A01

Docencia-3	
dni	***aula***
30.111.222	A02
30.111.222	A01
30.666.777	A01

Si se realiza la reunión natural de las relaciones *Docencia-1* y *Docencia-2*, se obtiene:

Docencia-12 = *Docencia-1* |x| *Docencia-2*

Docencia-12		
dni	***asignatura#***	***aula***
30.111.222	001	A02
30.111.222	002	A01
30.666.777	*001*	*A02*
30.666.777	001	A01
30.111.222	001	A01

y si ahora se reúne la relación *Docencia-12* con la relación *Docencia-3*, se obtiene:

Docencia-123 = *Docencia-12* |x| *Docencia-3*

Docencia-123		
dni	***asignatura#***	***aula***
30.111.222	001	A02
30.111.222	002	A01
30.666.777	001	A01
30.111.222	*001*	*A01*

Del análisis de este proceso se puede apreciar:

- Que en la relación *Docencia-12* está presente la tupla *<30.666.777, 001, A02>* que es espuria, la cual desaparece cuando esta relación se reúne con la relación *Docencia-3*.

- Que la relación *Docencia-123* no tiene la misma extensión que la relación *Docencia*, violando la dependencia de reunión que, por definición, existe en esta relación. Puede observarse que en la relación *Docencia-123* está presente la tupla *<30.111.222, 001, A01>*, que no existe en la relación *Docencia*.

La razón de esta inconsistencia es bien simple. Si se parte de la base de que en el ejemplo propuesto existe una dependencia de reunión, se tiene que un alumno va a recibir la docencia de una asignatura en la que esté matriculado en todas y cada una de las aulas asignadas a esa asignatura.

Al insertar la tupla *<30.666.777, 001, A01>*, se está insertando la información de que la asignatura *001* se imparte en el aula *A01*, y como existe una tupla en la relación *Docencia* que informa de que en la asignatura *001* está matriculado el alumno *30.111.222* es obligatorio *insertar una nueva tupla* (la tupla ***<30.111.222, 001, A01>***) en la que se represente que este alumno también recibe la docencia en el aula *A01* por el mero hecho de estar matriculado en la asignatura *001*.

Sin embargo, es fácil comprobar (basta realizar la proyección y posterior reunión de las relaciones) que si en lugar de la tupla *<30.666.777, 001, A01>* se inserta la tupla *<30.111.222, 001, A01>*, no se habría producido ninguna inconsistencia.

El problema presentado de la dependencia de reunión para el proceso de inserción se puede presentar para la operación de borrado de una tupla. Por ejemplo, si en la relación *Docencia-123* se eliminara la tupla *<30.666.777, 001, A01>*, la nueva relación obtenida sería consistente manteniendo la dependencia de reunión. Pero, sin embargo, si se eliminara la tupla *<30.111.222, 001, A01>*, sería necesario eliminar también la tupla *<30.111.222, 001, A02>* para mantener la consistencia de la nueva relación obtenida.

3.4.3. Diferencias entre la *FN4* y *FN5*

Se ha podido observar que en el desarrollo de las secciones anteriores, en las que se han explicado las reglas de normalización *FN4* y *FN5*, se ha utilizado una misma relación, la relación *Docencia*, como ejemplo para la aplicación de las mismas.

Sin embargo, en la sección correspondiente a la *FN5* se ha argumentado de partida que en la relación *Docencia* no existían dependencias multivaluadas, mientras que en la sección correspondiente a la *FN4*, el punto de partida era la existencia de este tipo de dependencias en la citada relación.

Si se analizan bien los argumentos de partida en cada uno de los casos, aunque se ha utilizado la misma relación, los requisitos han sido completamente diferentes. En cada uno de los casos la relación *Docencia* estaba representando problemas del mundo real (ejemplo de trabajo) distintos y, aunque deba ser utilizada la misma relación bajo una óptica relacional (naturalmente, sin normalizar en las formas *FN4* o *FN5*), las dependencias existentes entre los atributos de la misma son distintas.

Las diferencias, antes citadas, están basadas en:

Para el caso de la *FN4*: se ha considerado que:

- Las asignaturas y las aulas son independientes.
- Tanto las asignaturas como las aulas dependen del alumno; es decir, un alumno se puede matricular en un conjunto de asignaturas, y a un alumno se le asigna un conjunto de aulas en base a sus propiedades (con independencia de las asignaturas en que se matricule).
- Es decir, el caso real que se representa es que, aunque completamente irreal, cada alumno va a cualquiera de las aulas que se le asigne, por ejemplo, por su valor del *dni*, para recibir la docencia de cualquiera de las asignaturas en que se matricule.

Para el caso de la *FN5*: se ha considerado que:

- Las aulas y las asignaturas son dependientes. A cada asignatura se le asigna un conjunto de aulas para que en ellas se imparta su docencia.
- La docencia para una asignatura se imparte en cada una de las aulas asignadas a esa asignatura.
- Por tanto, un alumno recibirá la docencia de una asignatura en cada una de las aulas asignadas a esa asignatura.
- Es decir, el caso real que representa es que para la impartición de la docencia correspondiente a una asignatura se utilizan varias aulas (por ejemplo, una para teoría, otra para problemas, otra para laboratorio, etc.) y que, por tanto, todos los alumnos matriculados en esa asignatura utilizan todas las aulas asignadas a la misma.

Como se ha podido observar, la diferencia entre una dependencia multivaluada y de reunión es función directa de la semántica de la relación, de ahí la dificultad en la observación de este tipo de dependencias y, para el caso de las de reunión, la dificultad en el mantenimiento de las relaciones (normalizadas o no) en las que existen atributos con estas dependencias.

CAPÍTULO 4

EL ÁLGEBRA RELACIONAL. SQL

Al mismo tiempo que E. F. Codd propuso el modelo relacional como una nueva forma de representar la información para su tratamiento, se propusieron dos planteamientos o visiones diferentes, desde el punto de vista matemático, para el acceso y manipulación de la información representada bajo este modelo: La visión conjuntivista y la visión predicativa de los datos.

Sobre la base de cada una de estas visiones se han ido desarrollando lenguajes que están basados, respectivamente, en el álgebra y en el cálculo de predicados.

Codd, cuando propuso el modelo relacional, sentó las bases de un lenguaje algebraico denominado *SQUARE* para la manipulación de los datos. Este lenguaje fue mejorándose y ampliándose, hasta que se denominó posteriormente *SEQUEL* y, por último, en 1976, *SQL* —debido al sonido de la pronunciación del nombre primitivo—.

Durante la década de los setenta tanto el modelo relacional como el lenguaje propuesto por Codd estuvieron siendo validados, período en el cual una serie de firmas empezaron a comercializar productos basados en los mismos. No fue hasta los primeros años de la década de los ochenta cuando *IBM* —padre de la idea— comercializó su primer producto relacional, el *SQL/DS* y el *DB2*, más adelante. Y fue en esta década cuando *ANSI* publicó los primeros estándares relacionales.

Hoy en día existe en el mercado un gran número de *SGBD* relacionales cuyo lenguaje de manipulación de datos es el *SQL* con más o menos mejoras introducidas particularmente por los fabricantes. *SQL* se ha convertido en un estándar gracias a las mejoras que ha experimentado en los últimos años que le han conferido una gran potencia y sencillez de uso y aprendizaje.

4.1. EL ÁLGEBRA RELACIONAL

SQL es un lenguaje de manipulación de datos de los *SGBD* relacionales basado en el lenguaje algebraico al cual se le ha añadido una semántica que lo hace más próximo al lenguaje natural.

SQL utiliza una serie de operadores —operadores algebraicos— que operan sobre las relaciones o tablas de un esquema relacional. En una operación *SQL* pueden intervenir una o varias tablas y uno o varios operadores algebraicos. Cada operador puede operar sobre una o dos tablas —operadores unarios y binarios— pero siempre sobre la totalidad de las tuplas que forman la extensión de la tabla.

El resultado de una operación *SQL* es una tabla, la cual, a su vez, es susceptible de ser sometida a nuevas operaciones *SQL*. Es decir, al igual que en las operaciones algebraicas matemáticas, las operaciones *SQL* se pueden concatenar.

Si bien *SQL* utiliza los operadores algebraicos matemáticos, éstos operan de forma sensiblemente diferente a como lo hacen en la teoría de conjuntos ya que se aplican sobre tablas relacionales en lugar de sobre conjuntos o relaciones. A estos operadores algebraicos se les denomina operadores básicos. Además, *SQL* incorpora una serie de operadores avanzados —en realidad son composiciones de los operadores básicos— que le confiere una gran potencia al lenguaje.

4.1.1. Los Operadores Básicos

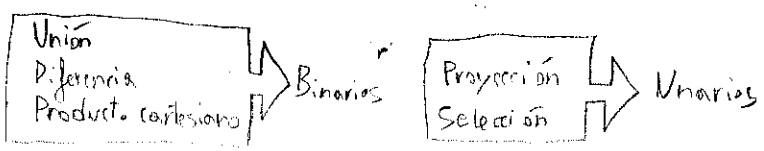
Los operadores algebraicos básicos son: *unión, diferencia, selección, proyección* y *producto cartesiano*. Los operadores unión, diferencia y producto cartesiano son operadores binarios, mientras que los operadores selección y proyección son unarios. Además, para los operadores binarios, unión y diferencia, es necesario que las dos tablas que intervienen en la operación sean compatibles.

> *Dos relaciones* R_1 *y* R_2 *se dice que son compatibles si ambas relaciones tienen el mismo grado y el atributo enésimo de* R_1 *está definido en el mismo dominio que el atributo enésimo de la relación* R_2*, si bien el nombre de los atributos puede ser diferente.*

4.1.1.1. OPERADOR UNIÓN (*UNION*)

> *La unión de dos relaciones compatibles* R_1 *y* R_2 *es una nueva relación* R_3*, también compatible, cuyo esquema es igual al esquema de* R_1 *y* R_2*, y cuya extensión está formada por la agrupación, sin repetición, de las extensiones de* R_1 *y* R_2*.*

Ejemplo: **Tabaco-3 = Tabaco-1** *UNION* **Tabaco-2**



TABACO-1				
Nombre	**Licencia**	**Hoja**	**Nic.**	**Alq.**
Camel sin filtro	R.J Reynolds Tobacco Co.	Turca	1.1	15
Marlboro	Philips Morris	Sin especificar	0.9	12
Coronas	Tabacalera S.A.	Canaria	0.9	15
Rex	Tabacalera S.A.	Canaria	0.9	15

TABACO-2				
Nombre	**Licencia**	**Hoja**	**Nic.**	**Alq.**
Ducados	Tabacalera S.A.	Sin especificar	1.1	15
Fortuna	Tabacalera S.A.	Sin especificar	1.0	14
Camel sin filtro	R.J Reynolds Tobacco Co.	Turca	1.1	15
Lucky sin filtro	Sin especificar	Sin especificar	1.0	15
Habanos	Tabacalera S.A.	Sin especificar	1.3	15

TABACO-3				
Nombre	**Licencia**	**Hoja**	**Nic.**	**Alq.**
Camel sin filtro	R.J Reynolds Tobacco Co.	Turca	1.1	15
Marlboro	Philips Morris	Sin especificar	0.9	12
Coronas	Tabacalera S.A.	Canaria	0.9	15
Rex	Tabacalera S.A.	Canaria	0.9	15
Ducados	Tabacalera S.A.	Sin especificar	1.1	15
Fortuna	Tabacalera S.A.	Sin especificar	1.0	14
Lucky sin filtro	Sin especificar	Sin especificar	1.0	15
Habanos	Tabacalera S.A.	Sin especificar	1.3	15

4.1.1.2. OPERADOR DIFERENCIA (*MINUS*)

> *La diferencia de dos relaciones compatibles* R_1 *y* R_2 *es una nueva relación* R_3*, también compatible, cuyo esquema es igual al esquema de* R_1 *y* R_2*, y cuya extensión está formada por aquellas tuplas de la relación* R_1 *que no se encuentran en la relación* R_2*.*

Ejemplo: **Tabaco-4 = Tabaco-1** *MINUS* **Tabaco-2**

TABACO-4				
Nombre	**Licencia**	**Hoja**	**Nicotina**	**Alquitran**
Marlboro	Philips Morris	Sin especificar	0.9	12
Coronas	Tabacalera S.A.	Canaria	0.9	15
Rex	Tabacalera S.A.	Canaria	0.9	15

4.1.1.3. OPERADOR SELECCIÓN (*SELECT*)

La selección sobre una relación R_1 *mediante una cualificación* Q *es una nueva relación* R_2*, cuyo esquema es igual a* R_1*, y cuya extensión está formada por todas aquellas tuplas de* R_1 *que satisfacen la cualificación Q.*

La cualificación Q es una expresión lógica cuyo formato general es:

R_1.atributo ⊗ valor [(AND, OR) R_1.atributo ⊗ valor ...]

donde: ⊗ representa a los operadores de comparación: ≤ , < , = , > , ≥ , ≠ .

Ejemplo: **Tabaco-5** = *SELECT*(**Tabaco-1**/Licencia = Tabacalera S.A.)

TABACO-5

Nombre	Licencia	Hoja	Nicotina	Alquitran
Coronas	Tabacalera S.A.	Canaria	0.9	15
Rex	Tabacalera S.A.	Canaria	0.9	15

4.1.1.4. OPERADOR PROYECCIÓN (*PROJECT*)

La proyección sobre una relación R_1 *con esquema* $R_1.a_i$, $R_1.a_j$, ..., $R_1.a_z$ *mediante un subesquema* $S(R_1) \equiv R_1.a_m$, $R_1.a_n$, ..., $R_1.a_p$, *donde* $a_m \geq a_i$ *y* $a_p \leq a_z$*, es una nueva relación* R_2*, cuyo esquema es igual al subesquema* $S(R_1)$*, y cuya extensión es igual a todas las tuplas de* R_1 *sin repetición sobre el subesquema* $S(R_1)$.

Ejemplo: **Tabaco-6** = *PROJECT*(**Tabaco-1**/Nombre, Nicotina, Alquitran)

TABACO-6

Nombre	Nicotina	Alquitran
Camel sin filtro	1.1	15
Marlboro	0.9	12
Coronas	0.9	15
Rex	0.9	15

4.1.1.5. OPERADOR PRODUCTO CARTESIANO (*PRODUCT*)

El producto cartesiano de dos relaciones R_1 *y* R_2 *no necesariamente compatibles es una nueva relación* R_3*, cuyo esquema es igual a la concatenación de los esquemas de* R_1 *y* R_2*, y cuya extensión está formada por el conjunto de las tuplas que se obtiene de concatenar cada una de las tuplas de* R_1 *con todas y cada una de las tuplas de* R_2.

Ejemplo: **Tabaco-7** = **Tabaco-1** *PRODUCT* **Estancos**

ESTANCOS

Propietario	Calle	Telefono
La Pajarita	El Nido, 5	276-5578
El Clavel	El Jardín, 23	444-8765

TABACO-7

Nombre	Licencia	Hoja	Nic.	Alq.	Propietario	Calle	Telefono
Camel s.f.	R.J Reynolds	Turca	1.1	15	La Pajarita	El Nido, 5	276-5578
Camel s.f.	R.J Reynolds	Turca	1.1	15	El Clavel	El Jardín, 23	444-8765
Marlboro	Philips Morris	s.e.	0.9	12	La Pajarita	El Nido, 5	276-5578
Marlboro	Philips Morris	s.e.	0.9	12	El Clavel	El Jardín, 23	444-8765
Coronas	Tabacalera S.A.	Canaria	0.9	15	La Pajarita	El Nido, 5	276-5578
Coronas	Tabacalera S.A.	Canaria	0.9	15	El Clavel	El Jardín, 23	444-8765
Rex	Tabacalera S.A.	Canaria	0.9	15	La Pajarita	El Nido, 5	276-5578
Rex	Tabacalera S.A.	Canaria	0.9	15	El Clavel	El Jardín, 23	444-8765

4.1.2. Operadores algebraicos avanzados

Las operaciones relacionales realizadas con estos operadores pueden descomponerse en un conjunto de operaciones básicas en las que sólo intervienen los operadores primitivos o básicos estudiados en la sección anterior. Se trata de operadores muy potentes que realizan operaciones complejas con las tablas que forman parte de un esquema relacional. Estos operadores son: la intersección, la reunión y la división.

4.1.2.1. OPERADOR INTERSECCIÓN (*INTERSECT*)

La intersección de dos relaciones compatibles R_1 *y* R_2 *es una nueva relación* R_3*, también compatible, cuyo esquema es igual al esquema de* R_1 *y* R_2*, y cuya extensión está formada por el conjunto de tuplas que son comunes a* R_1 *y* R_2.

La operación de intersección es una operación formada por un conjunto de operaciones básicas, en particular, por una operación unión y dos operaciones diferencia. Así, la intersección de dos relaciones R_1 y R_2 se puede expresar de la forma siguiente:

$$R_3 = R_1 \textit{ INTERSECT } R_2 =$$
$$= R_2 \textit{ MINUS } ((R_1 \textit{ UNION } R_2) \textit{ MINUS } R_1) =$$
$$= R_1 \cap R_2 = R_2 - ((R_1 \cup R_2) - R_1) = R_1 - (R_1 - R_2)$$

Ejemplo: **Tabaco-8** = **Tabaco-1** *INTERSECT* **Tabaco-2**

TABACO-8				
Nombre	**Licencia**	**Hoja**	**Nic.**	**Alq.**
Camel sin filtro	R.J. Reynolds Tobacco Co.	Turca	1.1	15

4.1.2.2. OPERADOR REUNIÓN (*JOIN*)

La reunión de dos relaciones R_1 *y* R_2 *no necesariamente compatibles, pero en las que existe al menos un atributo con el dominio común, sobre una cualificación* Q, *es una nueva relación* R_3 *cuya intención está formada por la concatenación de las intenciones de* R_1 *y* R_2, *y cuya extensión está formada por las tuplas que resultan del producto cartesiano de* $R_1 \times R_2$ *que satisfacen la cualificación* Q.

Dependiendo del tipo de cualificación *Q* la operación reunión recibe distintos nombres. Así:

Equi*reunión*: en la que la cualificación *Q* es una expresión que contiene el operador de comparación = (igual).

Theta*reunión*: ó Θ-reunión en la que la cualificación *Q* es una expresión que contiene un operador de comparación distinto a la igualdad.

reunión **Natural:** se representa por el signo |x|, y es una equireunión sobre todos los atributos comunes existentes entre las relaciones R_1 y R_2.

Auto*reunión*: es una reunión natural sobre una misma relación; es decir, $R_1 = R_2$.

Semi*reunión*: esta operación se realiza entre dos relaciones R_1 y R_2 sobre una cualificación multiatributo *Q* en la que puede aparecer cualquier operador de comparación, y el resultado es una relación R_3 de esquema igual a R_1, y cuya extensión está formada por todas aquellas tuplas de R_1 que intervienen en la reunión de R_1 y R_2.

Se puede observar que se trata de un tipo especial de reunión, puesto que se genera una relación con un esquema muy diferente al que se produce con el resto de los tipos de reuniones. Esta operación se representa por el signo |x.

Dada la relación *Descripcion* que representa, para los distintos tipos de hojas, las diferentes picaduras de tabaco que se preparan, podemos realizar una reunión de la forma siguiente:

Ejemplo: **Tabaco-9 = Tabaco-1** *JOIN* **Descripcion**

DESCRIPCION		
Hoja	**Clase**	**Color**
Turca y Americana	Normal	Rubio
Turca y Americana	Light	Rubio
Holandesa	Normal	Rubio
Canaria	UltraLight	Negro

TABACO-9						
Nombre	**Licencia**	**Hoja**	**Nic.**	**Alq.**	**Clase**	**Color**
Camel sin filtro	R.J. Reynolds ...	Turca ...	1.1	15	Normal	Rubio
Camel sin filtro	R.J. Reynolds ...	Turca ...	1.1	15	Light	Rubio
Coronas	Tabacalera S.A.	Canaria	0.9	15	UltraLight	Negro
Rex	Tabacalera S.A.	Canaria	0.9	15	UltraLight	Negro

4.1.2.3. OPERADOR DIVISIÓN (*DIVISION*)

La división de una relación R_1 *con esquema* $R_1.a_1, R_1.a_j, \ldots, R_1.a_z$ *entre una relación* R_2 *de esquema* $R_1.a_m, R_1.a_n, \ldots, R_1.a_p$, *donde* $a_m \geq a_i$ *y* $a_p \leq a_z$, *es una nueva relación* R_3 *cuyo esquema es igual a la diferencia del esquema de* R_1 *menos el esquema de* R_2, *y cuya extensión es igual a todas las tuplas de* R_1 *sin repetición para las cuales está presente toda la extensión de la relación* R_2.

Se trata de una operación compleja y a la vez muy potente que permite expresar de forma muy sencilla el cuantificador universal (∀). Esta operación tiene las siguientes características:

1. El esquema de la relación R_2 debe ser un subesquema del esquema de la relación R_1 y los atributos deben tener el mismo nombre.
2. El esquema de la relación resultado de la división está formado por todos aquellos atributos del esquema de R_1 que no están presentes en el esquema de R_2.
3. Las tuplas de la relación resultado R_3 son todas aquellas tuplas de la relación R_1 para las cuales, y para cada valor de los atributos del esquema de R_3, están presentes en R_1 todas las tuplas de la extensión de la relación de R_2 (para todos los valores de R_2).
4. Esta operación se representa también con el signo ÷.

Dada la relación *Distribucion* que representa los distintos fabricantes de tabacos que distribuyen los estancos, podemos realizar una división de la forma siguiente:

Ejemplo: **Tabaco-10 = Tabaco-7** *DIVISION* **Distribucion**

DISTRIBUCION			
Propietario	**Calle**	**Telefono**	**Licencia**
La Pajarita	El Nido, 5	276-5578	Tabacalera S.A.
El Clavel	El Jardín, 23	444-8765	Tabacalera S.A.

TABACO-10			
Nombre	**Hoja**	**Nicotina**	**Alquitran**
Coronas	Canaria	0.9	15
Rex	Canaria	0.9	15

4.2. EL LENGUAJE RELACIONAL *SQL*

SQL es el lenguaje estandarizado y de uso universal utilizado por los *SGBD* basados en el álgebra relacional. No es objeto de este manuscrito ofrecer una explicación detallada de este lenguaje, por lo que sólo nos detendremos con detenimiento en los aspectos relacionados con la manipulación de los datos.

Como cualquier lenguaje de un *SGBD*, *SQL* cubre tres aspectos bien diferenciados: la descripción, la manipulación y el control y seguridad de los datos. Para cada uno de estos aspectos *SQL* cuenta con un conjunto de verbos que realizan funciones específicas con la información.

Si bien *SQL* es un lenguaje interactivo, las sentencias *SQL* pueden ser incorporadas a las sentencias de programas realizadas con lenguajes *huésped* como: *COBOL, C, JAVA, PL/SQL*, etc.

Un esquema de una base de datos relacional está formado, básicamente, por la definición de un conjunto de tablas (relaciones).

Cada tabla debe tener un nombre único en el esquema y estar definida sobre la base de la especificación de un conjunto de atributos, las columnas de la tabla, que representan a propiedades del objeto del mundo real representado por la tabla.

Cada atributo debe tener un nombre único para una tabla y estará definido en un dominio de datos preestablecido. De entre todos los atributos que forman la definición de una tabla, uno o un conjunto de ellos se elegirán como clave principal de la misma.

Opcionalmente, en la definición de una tabla se especificarán las claves foráneas; es decir, aquellos atributos de la tabla que están definidos en el mismo dominio y representan la misma propiedad que la clave principal de alguna otra tabla que forma parte del esquema de la base de datos.

SQL utiliza el verbo **CREATE TABLE** para la definición de las tablas del esquema relacional. El formato general es el siguiente:

```
CREATE TABLE NombreTabla    (
  nombre-atributo-1:        DOMINIO (opción),
  nombre-atributo-2:        DOMINIO (opción),
. . . . . .                 . . . . . .
  nombre-atributo-n:        DOMINIO (opción),
  PRIMARY KEY               (lista-atributos),
  FOREIGN KEY               (lista-atributos),
      REFERENCES            NombreTabla(clave-principal)
      [ON DELETE CASCADE]   );
```

Para cada atributo es necesario especificar un dominio en el cual pueda tomar valores esa propiedad del objeto. La definición de un dominio comprende:

1. La definición de un tipo de datos. *Oracle* reconoce una serie de tipos de datos predefinidos, que se especializan en subtipos, entre los cuales se encuentran:

CHAR*(longitud)*, representando a cadenas de caracteres de longitud fija de hasta un máximo de 2.000 bytes[12]. El valor por defecto, si no se especifica la longitud de la cadena, es de 1 byte.

VARCHAR2*(longitud)*, representando a cadenas de caracteres de longitud variable de hasta un máximo de 4.000 caracteres, si bien es necesario indicar la longitud máxima esperada de la cadena. Los subtipos existentes son *VARCHAR* y *STRING*[13].

NUMBER*(precisión)*, representando a información numérica de longitud fija o coma flotante en un rango de 1^{-130} a 10^{125}. Existe un amplio número de subtipos dependiendo de la clase de número que se desee almacenar: *DECIMAL, DOUBLE PRECISION, FLOAT, INTEGER, NUMERIC, REAL, SMALLINT.*

BOOLEAN, representando a información lógica. Puede tomar los valores *TRUE, FALSE* o *NULL.*

DATE, representando información cronológica: fechas válidas (incluyendo horario) desde el 1 de enero del 4712 a.C hasta el 31 de diciembre del 4712 d.C. El uso de este tipo de dato es relativamente complejo, y se analizará con mayor amplitud en los ejemplos de manipulación de la información que se introducirán en este y sucesivos capítulos.

RAW*(longitud)*, representando datos binarios de tamaño o longitud máxima de 2.000 bytes.

---

[12] En un procedimiento *PL/SQL*, una variable definida en este tipo puede almacenar hasta 32767 bytes.

[13] En un procedimiento *PL/SQL*, una variable definida en este tipo puede almacenar hasta 32767 bytes.

LONG RAW, similar al anterior tipo. El tamaño máximo es de 2 Gb.

ROWID, un tipo de dato binario que representa la dirección de una tupla de una tabla.

CLOB, representando a objetos de caracteres de hasta 4 Gb.

BLOB, representando a objetos binarios de hasta 4 Gb. En las nuevas versiones se utiliza en lugar de *LONG RAW*.

BFILE, almacena punteros de archivos a *LOB* administrados por sistemas de archivos externos a la base de datos.

Oracle proporciona una serie de funciones que permite la conversión de datos de un tipo a otro cuando ésta tiene sentido, como por ejemplo:

TO_DATE, para convertir datos numéricos o cadenas a *DATE*.

TO_CHAR, para convertir datos numéricos o *DATE* a datos *CHAR* o *VARCHAR2*.

TO_NUMBER, para convertir datos alfanuméricos a numéricos.

Por otra parte, *Oracle* convierte automáticamente un valor de un tipo de dato a otro cuando esta operación tiene sentido.

2. En la definición de los atributos se pueden especificar algunas restricciones u opciones para ellos. Estas restricciones se pueden especificar directamente en la definición de la columna de la tabla, o más fácilmente haciendo uso de las cláusulas *CONSTRAINT*.

   Un *CONSTRAINT* es un cuerpo de definición en el esquema que permite definir las restricciones de integridad de la información definida en el mismo. La sintaxis general es la siguiente:

```
CONSTRAINT nombre_constraint
           cuerpo-constraint,
```

   donde el *nombre_constraint* debe ser único en el esquema, y en el *cuerpo_constraint* pueden aparecer las siguientes cláusulas:

NOT NULL: para indicar que el atributo no podrá tomar valores nulos para ninguna tupla de esa tabla. Esta opción va a permitir el control de la integridad de la clave principal de las tablas, y la representación de la debilidad por existencia en las claves foráneas.

PRIMARY KEY: para indicar el atributo o conjunto de atributos que deben ser considerados como la clave principal de la tabla. Estos atributos no pueden haber sido definidos en los tipos *LONG RAW, CLOB, BLOB*, entre otros.

UNIQUE: para indicar que ese atributo o lista de atributos no podrá tomar valores duplicados en la tabla. Esta opción permite la especificación de unicidad de la clave principal o secundaria de una tabla.

FOREIGN KEY: para indicar al atributo o conjunto de atributos que actúan como clave foránea de la clave principal de otra tabla del esquema, y así garantizar la integridad de referencia. Esta cláusula viene acompañada siempre de la siguiente.

REFERENCES: para indicar a qué clave principal de qué tabla se realiza una referencia por una clave foránea definida mediante la cláusula anterior.

ON DELETE CASCADE: acompaña siempre a las cláusulas anteriores e indica si en el proceso de borrado de la tupla referenciada por tuplas de otra tabla deben ser borradas éstas también.

CHECK: para indicar cualquier restricción en cuanto al conjunto de valores que puede tomar un atributo de una tabla.

DISABLE: para indicar la desactivación temporal de un *CONSTRAINT*.

ENABLE: para indicar la activación de un *CONSTRAINT* desactivado temporalmente.

La definición de una tabla del esquema relacional puede ser modificada haciendo uso del verbo *ALTER TABLE*, cuya sintaxis es la siguiente:

```
ALTER TABLE nombre-tabla
            cuerpo-modificación;
```

pudiendo realizarse modificaciones sobre los siguientes objetos del esquema, entre otros:

- Añadir una columna o atributo

```
ADD (nombre_atributo DOMINIO (opción))
```

- Añadir un *CONSTRAINT*

```
ADD CONSTRAINT (definición del constraint)
```

- Activar, desactivar o borrar un *CONSTRAINT*

```
ENABLE, DISABLE, DROP CONSTRAINT
```

En la definición de los esquemas relacionales son muy importantes los índices. Un índice es un objeto de la base de datos que contiene una entrada para cada valor que aparece en una columna (o conjunto de ellas) de una tabla, proporcionando accesos rápidos y directos a las tuplas de la tabla indexada.

Sin embargo, un índice puede aminorar los desempeños de las operaciones de inserción, borrado y modificación que afectan a las columnas para las cuales se ha

generado el índice debido a que *Oracle* debe gestionar tanto los índices como la información de la tabla indexada.

Así, cuando se insertan inicialmente valores de una tabla, es generalmente más rápido crear una tabla, insertar las tuplas y, posteriormente, crear el índice. Si se crea el índice antes de la inserción de las tuplas, *Oracle* debe actualizar el mismo tras cada inserción. Para crear un índice se utiliza la siguiente sintaxis:

```
CREATE INDEX nombre_indice ON
             nombre_tabla(lista_campos)
             [OPCIONES];
```

Otro elemento de interés y necesario para la definición de la base de datos es la *VISTA*. Una vista *VIEW* es una tabla lógica que permite acceder a la información de otras tablas o vistas[14].

Una vista permite además proporcionar un nivel adicional de seguridad a las tablas del esquema, restringiendo el acceso a un conjunto predeterminado de tuplas y/o columnas de las tablas base en las que se apoya.

Por otra parte, una vista permite ocultar el nivel de complejidad de los datos, proporcionando una visión lógica externa de los mismos más simple y próxima al usuario.

El lector puede observar que una vista es una visión externa parcial del esquema lógico general de la base de datos, como así ha sido descrito en los capítulos anteriores. Para crear una vista se utiliza la siguiente sintaxis:

```
CREATE VIEW nombre_vista AS
            sentencia SELECT;
```

donde *sentencia SELECT* es una expresión bien formada del *DML*, cuya sintaxis se describirá a continuación.

4.2.1. Manipulación de la información

La manipulación de la información de una base de datos comprende operaciones de inserción, modificación, borrado y consulta de los datos almacenados en la misma. *SQL* incorpora verbos para cada una de estas operaciones, con una sintaxis clara y sencilla, a la vez que potente, puesto que permite la anidación de sentencias en las que aparecen más de un verbo *SQL* que realizan la misma o distinta operación.

[14] El término *lógica* quiere decir que una vista no contiene información por sí misma, sino que su información está basada en la que contienen otras tablas a las que se les denomina *tablas base*.

Para presentar algunos ejemplos de sentencias *SQL* nos vamos a apoyar en la base de datos descrita en el Capítulo 10, y que hace referencia al problema de la *Venta y Consumo de Cigarrillos*. Por lo tanto, recomendamos al lector que visite ese capítulo para conocer el conjunto de tablas sobre las cuales se van a proponer los ejemplos que se presentan a continuación.

4.2.1.1. CONSULTA DE LA INFORMACIÓN

Consultar información de una base de datos supone el acceso seleccionado de los datos existentes para un subconjunto de los atributos que forman parte del esquema de la misma, obteniendo como resultado una nueva tabla, un valor o conjunto de valores. *SQL* aporta una gran potencia, a la vez que sencillez, en los procesos de acceso a la información existente en la base de datos. El formato general de esta operación es el que se muestra y que se interpreta de la forma siguiente:

```
SELECT [DISTINCT | ALL] (lista-atributos-1 | *)
  [INTO cursor]
  FROM lista-tablas
    [WHERE condición-1]
  [GROUP BY lista-atributos-2
    [HAVING condición-2]]
  [ORDER BY lista-atributos-3 [ASC | DESC]]
```

La *lista-atributos-1* representa a aquel conjunto de atributos sobre los cuales se desea realizar una operación de proyección. Si no se quiere realizar una proyección, este parámetro se sustituye por el modificador *, que representa a todos aquellos atributos de todas las tablas que intervienen en la operación de consulta.

La palabra *DISTINCT*, acompañando a algún atributo, indica que no podrán existir valores duplicados para ese atributo en las tuplas de la relación resultante. Por defecto, en el resultado de una consulta pueden existir valores duplicados para los atributos seleccionados *(ALL)*.

El verbo *INTO* permite asignar una variable definida por el usuario para el control y acceso de las tuplas resultado de la operación de consulta.

Las tablas que van a ser manejadas en la operación de consulta son especificadas mediante la cláusula *FROM*, detallando cada una de las tablas separadas por comas.

Si se especifica la cláusula *WHERE* la operación de consulta realiza una selección de las tuplas resultantes sobre la base de la expresión especificada en la *condición-1*. Esta expresión puede estar formada por otra instrucción *SELECT* de forma que se pueden anidar las consultas, teniendo en cuenta que son las consultas más internas las que se realizan en primer lugar.

Mediante la cláusula *GROUP BY* pueden construirse grupos de tuplas sobre la base de la *lista-atributos-2*. La agrupación permite la recuperación de una sola tupla por cada uno de los grupos construidos sobre la base de cada uno de los valores de los atributos especificados en la *lista-atributos-2*.

La cláusula *HAVING* permite restringir el conjunto de grupos que pueden ser formados por la cláusula *GROUP BY*. Como se puede apreciar tiene la misma función que la cláusula *WHERE*. Por último, la cláusula *ORDER BY* tiene como función proporcionar un orden de las tuplas obtenidas como resultado de la consulta. En esta cláusula se indican los atributos que van a determinar el orden de las filas y si el orden será ascendente *(ASC)* o descendente *(DESC)*.

OPERADORES Y FUNCIONES

Oracle pone a disposición del usuario una serie de operadores y funciones que le ayudan en la consulta, manipulación y operación sobre los datos almacenados en la base de datos. Vamos, a continuación, a describir únicamente aquéllos más generales y que serán utilizados en los capítulos próximos. Algunos de los operadores más utilizados son:

OPERADORES DE COMPARACIÓN (x Θ y)	
=, !=, <, <=, >, >=, IS NULL	Comparan dos operandos simples.
OPERADORES DE COMPARACIÓN (x Θ lista-1)	
=, !=, <, <=, >, >=, ANY, SOME, ALL, IN, NOT IN	Comparan el operando *x* con una lista de valores especificados o bien obtenidos a través de una subconsulta.
EXISTS, NOT EXISTS	Compara el operando *x* con una lista de valores obtenidos a través de una subconsulta.
OPERADORES ARITMÉTICOS (x Θ y)	
=, +, -, *, /	Los operadores clásicos de asignación (=), y de las operaciones aritméticas de suma, resta, multiplicación y división.
OTROS LÓGICOS	
NOT, AND	Operadores lógicos clásicos.
OTROS OPERADORES	
x BETWEEN y AND z	Calcula si *x* se encuentra en el intervalo marcado por los valores [*y*, *z*].
x LIKE y [ESCAPE z]	Calcula si el patrón de *x* se encuentra en y. Se pueden utilizar caracteres comodín (% y _), para representar cualquier cadena excepto el carácter nulo y un carácter sencillo respectivamente. Se utiliza la opción ESCAPE para indicar que se desea que el carácter z se interprete de forma literal (por ejemplo para los caracteres comodín).

Y algunas de las funciones más utilizadas son:

OPERADORES DE COMPARACIÓN (x Θ y)	
UPPER(cadena)	Convierte una cadena a mayúsculas.
INITCAP(cadena)	Convierte a mayúscula sólo el primer carácter de la cadena.
MONTHS_BETWEEN(d1, d2)	Devuelve el número de meses transcurridos entre las dos fechas señaladas.
SYSDATE	Devuelve la fecha y hora del sistema.
TO_DATE(cadena [,formato])	Convierte una variable de tipo cadena a tipo DATE, indicando el formato en el cual se desea representar la fecha (y hora).
AVG(lista_valores)	Obtiene el valor medio de la lista de valores numéricos.
SUM(lista_valores)	Obtiene el sumatorio de la lista de valores numéricos.
MAX(lista_valores)	Obtiene el valor máximo de la lista de valores numéricos.
MIN(lista_valores)	Obtiene el valor mínimo de la lista de valores numéricos.
STDEV(lista_valores):	Obtiene la desviación típica de la lista de valores numéricos.
COUNT(expresión):	Obtiene el número de tuplas que satisfacen la expresión. Si la expresión es (*) obtiene el número de tuplas que satisface la consulta.

A continuación, se presentan algunos ejemplos de consultas *SQL*:

Cap04ej01.sql

Obtener el total de compras realizadas de cigarrillos de color negro.

Se realiza una inspección de la tabla Compras, seleccionando por el atributo color y calculando, mediante el operador SUM, el acumulado de las compras para cada tipo de cigarrillos. Por último, se ordena la salida en función del total de las compras.

```
SELECT SUM(c_comprada), marca, filtro, color, clase, mentol
   FROM Compras
   WHERE Compras.color = 'N'
   GROUP BY marca, filtro, color, clase, mentol
   ORDER BY 1;
```

Cap04ej02.sql

Obtener aquellos cigarrillos para los cuales existen pedidos de cantidad superior a la media de los pedidos realizados.

Se realiza, en primer lugar, el cálculo de la media de los pedidos para cada tipo de cigarrillo y, en base al valor obtenido, se obtienen los cigarrillos con pedidos superiores a la misma.

```
SELECT marca, filtro, color, clase, mentol,
       fecha_compra, c_comprada
   FROM Compras
```

```
   WHERE c_comprada >
      (SELECT AVG(c_comprada)
       FROM Compras CopiaCompras
       WHERE CopiaCompras.marca  = Compras.marca AND
             CopiaCompras.filtro = Compras.filtro AND
             CopiaCompras.color  = Compras.color AND
             CopiaCompras.clase  = Compras.clase AND
             CopiaCompras.mentol = Compras.mentol
       GROUP BY marca, filtro,
                color, clase, mentol);
```

Cap04ej03.sql

Obtener una relación completa de todas las compras y ventas de cigarrillos.

La respuesta a esta solicitud consiste en obtener todas las tuplas de las tablas Compras y Ventas, y unirlas con el uso del operador UNION.

```
SELECT *
   FROM Compras
 UNION
SELECT *
   FROM Ventas;
```

Cap04ej04.sql

Obtener la relación de cigarrillos que han sido abastecidos a los estancos para los cuales no se ha realizado ninguna venta.

Para obtener esta solicitud se obtendrán todas las tuplas de la tabla Compras y se las negará con el conjunto de las tuplas de la tabla Ventas sólo teniendo en cuenta los atributos que identifican a cada uno de los cigarrillos.

```
SELECT DISTINCT marca, filtro, color, clase, mentol
   FROM Compras
   WHERE (marca, filtro, color, clase, mentol) NOT IN
      (SELECT DISTINCT marca, filtro, color, clase, mentol
         FROM Ventas);
```

Cap04ej05.sql

Obtener la relación de cigarrillos que han sido abastecidos a los estancos y de los cuales se ha realizado alguna venta.

Procede de obtener los cigarrillos que existen tanto en la tabla Compras como en la tabla Ventas.

```
SELECT DISTINCT marca, filtro, color, clase, mentol
   FROM Compras
      INTERSECT
SELECT DISTINCT marca, filtro, color, clase, mentol
   FROM Ventas;
   /* Esta misma consulta se puede realizar sin hacer uso del
   verbo INTERSECT, de la forma siguiente */
SELECT DISTINCT marca, filtro, color, clase, mentol
   FROM Compras
   WHERE (marca, filtro, color, clase, mentol) IN
      (SELECT DISTINCT marca, filtro, color, clase, mentol
       FROM Ventas);
```


Cap04ej06.sql

Obtener la relación de estancos que no han vendido el cigarrillo de marca 'Ducados', con filtro, de color negro, sin mentol y clase normal.

Se realiza una inspección de la tabla Ventas cualificada por las condiciones impuestas en el enunciado, y se proyecta sobre los estancos.

```
SELECT *
   FROM Estancos
   WHERE nif_estanco NOT IN
      (SELECT DISTINCT nif_estanco
       FROM Ventas
       WHERE marca = 'Ducados'
          AND filtro = 'S' AND color = 'N'
          AND clase = 'Normal' AND mentol = 'N');
```

Cap04ej07.sql

Obtener la relación de fabricantes de los cuales se ha vendido algún cigarrillo.

Se realiza una inspección de la tabla Ventas y se relaciona con la tabla Cigarrillos, para obtener las marcas que satisfacen los requisitos del enunciado. En base a estas marcas se obtienen los fabricantes de las mismas a partir de la tabla Manufactura y, a continuación, se obtiene la información de la tabla Fabricantes.

```
SELECT *
   FROM Fabricantes
   WHERE nombre_fabricante IN
     (SELECT DISTINCT nombre_fabricante
      FROM Manufactura
      WHERE marca IN
        (SELECT DISTINCT marca
           FROM Cigarrillos
           WHERE  EXISTS
              (SELECT *
                  FROM Ventas
                  WHERE Cigarrillos.marca = Ventas.marca AND
                        Cigarrillos.filtro = Ventas.filtro AND
                        Cigarrillos.color = Ventas.color AND
                        Cigarrillos.clase = Ventas.clase AND
                        Cigarrillos.filtro = Ventas.filtro AND
                        Cigarrillos.mentol = Ventas.mentol)));
```

4.2.1.2. INSERCIÓN DE LA INFORMACIÓN

La inserción de la información consiste en la incorporación de nuevas tuplas a tablas de la base de datos. El formato general de esta operación en *SQL* es:

```
INSERT [INTO] (lista-atributos)
  NombreTabla [lista-tablas]
    VALUES (lista-valores)
```

La lista de valores puede introducirse directamente en la expresión o bien puede obtenerse a partir de la información existente en la base de datos mediante la inclusión de una consulta haciendo uso del verbo *SELECT*. En lugar de sobre una tabla, puede realizarse la inserción sobre una vista de la base de datos.

Si los atributos no se especifican en la cláusula *INSERT* se consideran, por defecto, todos los atributos de la tabla. Y si no se asigna un valor a un determinado atributo, por defecto, éste toma el valor nulo.

Cap04ej08.sql

Insertar una nueva marca de cigarrillos producida por un fabricante no existente en la base de datos.

Debido a la normalización existente en el esquema de la base de datos, es necesario realizar los siguientes procesos:

1. *Insertar la tupla correspondiente en la tabla Fabricantes.*
2. *Insertar la tupla correspondiente en la tabla Manufactura.*
3. *Insertar la tupla correspondiente en la tabla Cigarrillos.*

```
INSERT INTO Fabricantes
   VALUES ('Cohiba S. A.','España');
INSERT INTO Manufactura
   VALUES ('Cohiba', 'Cohiba S. A.', 10, 20);
INSERT INTO Cigarrillos
   VALUES ('Cohiba', 'N', 'R', 'Normal',
           'N', 1.1, 15, 2);
```

Cap04ej09.sql

Dar de alta un nuevo estanco y abastecerlo con 10 unidades de cada uno de los cigarrillos existentes en la base de datos.

Se realiza la inserción del nuevo estanco y se insertan tantas tuplas en las tablas Almacenes y Compras como resulte de la consulta sobre la tabla Cigarrillos de todas las tuplas existentes en la misma (en el ejemplo se introducen directamente las 10 unidades en las tuplas de la tabla Almacenes, si bien el proceso debería consistir en dar de alta cada cigarrillo en el almacén con cero unidades y, posteriormente, actualizar su stock en el almacén).

```
INSERT INTO Estancos
   VALUES ('3030404000-K', 3444, 28012, 'El Fumador Loco',
   'C/ La Hoja, 17', 'Madrid', 'Madrid');
INSERT INTO Almacenes
   (SELECT '3030404000-K', marca, filtro, color, clase,
           mentol, 10
    FROM Cigarrillos);
INSERT INTO Compras
   (SELECT '3030404000-K', marca, filtro, color, clase,
           mentol, SYSDATE, 10, precio_costo
    FROM Cigarrillos);
```

Cap04ej10.sql

Para todos los estancos de Sevilla realizar una venta automática de dos unidades para todos aquellos cigarrillos sin filtro y rubios.

Se insertarán en la tabla Ventas tantas tuplas como resulte de una subconsulta que se realiza sobre la reunión de las tablas Estancos, Cigarrillos y Almacenes en la que se seleccionan las tuplas que satisfacen los requisitos del enunciado (naturalmente, posteriormente a esta operación sería necesario actualizar la tabla Almacenes con las ventas realizadas).

```
INSERT INTO Ventas
   (SELECT DISTINCT Estancos.nif_estanco, Cigarrillos.marca,
           Cigarrillos.filtro, Cigarrillos.color,
           Cigarrillos.clase, Cigarrillos.mentol, SYSDATE,
           2, precio_venta
      FROM Estancos, Cigarrillos
      WHERE provincia_estanco = 'Sevilla'
           AND Cigarrillos,filtro = 'N'
           AND Cigarrillos.color  = 'R');
```

Cap04ej11.sql

Para todos aquellos tabacos de los que se hayan vendido más de cincuenta unidades en la fecha actual, realizar un pedido de compras del 50% de la cantidad vendida.

Se realiza una subconsulta sobre la tabla Ventas de la que se obtienen los cigarrillos que en la fecha actual se han vendido más de 50 unidades. El resultado de esta subconsulta dará las tuplas que deberán ser insertadas en la tabla Compras (de nuevo será necesario realizar una actualización posterior de la tabla Almacenes).

```
INSERT INTO Compras
   (SELECT DISTINCT nif_estanco, Ventas.marca, Ventas.filtro,
           Ventas.color, Ventas.clase, Ventas.mentol, SYSDATE,
           c_vendida*0.5, Cigarrillos.precio_costo
      FROM Ventas,Cigarrillos
      WHERE Ventas.c_vendida > 50 AND
           Ventas.fecha_venta <= SYSDATE
           AND Cigarrillos.marca  = Ventas.marca
           AND Cigarrillos.filtro = Ventas.filtro
           AND Cigarrillos.color  = Ventas.color
           AND Cigarrillos.clase  = Ventas.clase
           AND Cigarrillos.mentol = Ventas.mentol);
```

Cap04ej12.sql

Se ha fabricado un nuevo cigarrillo de marca 'Negritos', sin filtro, negro, 'SuperLight' y no mentolado, fabricado por un fabricante ya existente. Abastecer con 75 unidades a todos los estancos que no lo tengan.

En primer lugar, como ya existe el fabricante, se inserta la tupla correspondiente en la tabla Manufactura, para posteriormente insertar el cigarrillo (orden de inserción necesario debido a la referencia existente entre ambas tablas). A continuación se realizan las inserciones en las tablas Almacenes y Compras de todas aquellas tuplas obtenidas de una subconsulta de la que se extraen las tuplas que satisfacen los requisitos del enunciado.

```
INSERT INTO Manufactura
   VALUES ('Negritos', 'Tabacalera S.A.' ,10, 20);
INSERT INTO Cigarrillos
   VALUES('Negritos', 'S', 'N', 'SuperLight', 'N',
```

```
        0.8, 10, 100, 130);
INSERT INTO Almacenes
   (SELECT nif_estanco, 'Negritos', 'S', 'N',
           'SuperLight', 'N', 75
      FROM Estancos
      WHERE nif_estanco NOT IN
        (SELECT DISTINCT nif_estanco
           FROM Compras
           WHERE marca = 'Negritos' AND filtro = 'S'
                 AND color = 'N' AND clase = 'SuperLight'
                 AND mentol = 'N'));
INSERT INTO Compras
   (SELECT nif_estanco, 'Negritos', 'S', 'N',
           'SuperLight', 'N', SYSDATE, 75, 100
      FROM Estancos
      WHERE nif_estanco NOT IN
        (SELECT DISTINCT nif_estanco
           FROM Compras
           WHERE marca = 'Negritos' AND filtro = 'S'
                 AND color = 'N' AND clase = 'SuperLight'
                 AND mentol = 'N'));
```

4.2.1.3. MODIFICACIÓN DE LA INFORMACIÓN

La modificación de la información consiste en la actualización de los valores de los atributos para una o varias tuplas de una tabla. El formato general de esta operación en *SQL* es:

```
UPDATE NombreTabla
  SET atributo-i = valor
    [WHERE Condición];
```

Mediante esta instrucción son modificados los valores de los atributos especificados de la tabla al valor indicado en la cláusula. Este valor puede ser obtenido, al igual que con el verbo *INSERT*, de los valores existentes en la base de datos mediante la inclusión de una consulta a la misma. Opcionalmente, puede incluirse la cláusula *WHERE*, la cual permite que sólo se modifiquen los valores de los atributos cuando se cumpla la condición especificada en esta cláusula.

Cap04ej13.sql

Poner el código postal de todos los estancos de Madrid a 28004.

Se modifican aquellas tuplas de la tabla Estancos que satisfacen el requisito de que la localidad del mismo sea 'Madrid'.

```
UPDATE Estancos
  SET cp_estanco = 28004
    WHERE localidad_estanco = 'Madrid';
```


Cap04ej14.sql

Cambiar aquellos cigarrillos de clase 'Normal', con filtro fabricados por 'Tabacalera S.A.' de modo que ahora no tengan filtro.

Se realiza una consulta sobre la tabla Manufactura para obtener todas las marcas de cigarrillos fabricados por 'Tabacalera S.A', con filtro y de la clase 'Normal', lo que permitirá actualizar sólo aquellas tuplas de la tabla Cigarrillos cuya marca se encuentre en la lista obtenida por esa subconsulta.

```
UPDATE Cigarrillos
  SET filtro = 'N'
   WHERE  marca IN
     (SELECT marca
      FROM Manufactura
      WHERE nombre_fabricante = 'Tabacalera S.A.'
            AND filtro = 'S' AND clase = 'Normal');
```

Cap04ej15.sql

Para todos aquellos cigarrillos con valor de nicotina menor de 1.0 y clase 'Light' disminuir la última compra del mismo en 2 unidades.

De la tabla Compras se obtiene la última fecha en la que se realizó una compra, subconsulta que se utiliza para limitar el número de tuplas que satisfacen la reunión de la tabla Compras y Cigarrillos en la que se seleccionan también aquellas tuplas que satisfacen el enunciado. Por último, se modificarán aquellas tuplas de la tabla Compras cuyos cigarrillos se encuentran en la lista de tuplas obtenidas en la subconsulta anterior.

```
UPDATE Compras
  SET c_comprada = c_comprada - 2
   WHERE (marca, filtro, color, clase, mentol) IN
     (SELECT Compras.marca, Compras.filtro, Compras.color,
             Compras.clase, Compras.mentol
      FROM Compras, Cigarrillos
      WHERE Cigarrillos.clase = 'Light'
            AND Cigarrillos.nicotina < 1.0
            AND Compras.fecha_compra =
       (SELECT MAX(fecha_compra)
          FROM Compras
          WHERE Cigarrillos.marca = Compras.marca
             AND Cigarrillos.filtro = Compras.filtro
             AND Cigarrillos.color = Compras.color
             AND Cigarrillos.clase = Compras.clase
             AND Cigarrillos.mentol = Compras.mentol));
```

Cap04ej16.sql

Para todos aquellos cigarrillos de los que en el último mes se hayan realizado más de 20 ventas, modificar el pedido realizado en el día de hoy incrementándolo en 5 unidades.

Se realiza una subconsulta sobre la tabla Ventas de la que se obtienen aquellos cigarrillos que tuvieron más de 20 ventas en el último mes (para ello se hace uso del verbo GROUP BY). Por último, se actualizarán aquellas tuplas de la tabla Compras correspondientes a compras del día de hoy que corresponden a cigarrillos que se encuentran dentro del conjunto de tuplas obtenidas en la anterior subconsulta.

```
UPDATE Compras
   SET c_comprada = c_comprada + 5
    WHERE fecha_compra = SYSDATE AND
          (marca, filtro, color, clase) IN
      (SELECT DISTINCT marca, filtro, color, clase
          FROM Ventas
          WHERE MONTHS_BETWEEN(SYSDATE, fecha_venta) = 1
          GROUP BY marca, filtro, color, clase
             HAVING COUNT(*) > 20);
```

Cap04ej17.sql

Cambiar el país al valor 'USA' para todos aquellos fabricantes de cigarrillos de la clase 'SuperLight' y con filtro.

Se obtiene de la tabla Cigarrillos todos aquellos fabricantes que manufacturan cigarrillos del tipo indicado en el enunciado. De esta forma se modificarán aquellas tuplas de la tabla Fabricantes que satisfacen esta subconsulta.

```
UPDATE Fabricantes
   SET pais = 'USA'
    WHERE nombre_fabricante IN
      (SELECT DISTINCT nombre_fabricante
       FROM Manufactura
       WHERE marca IN
          (SELECT DISTINCT marca
             FROM Cigarrillos
             WHERE clase = 'SuperLight' AND filtro = 'S'));
```

4.2.1.4. BORRADO DE LA INFORMACIÓN

El borrado de la información consiste en la eliminación de una o varias tuplas de una tabla que satisfagan o no una condición. La tabla no es borrada del esquema de la base de datos, únicamente son borradas las tuplas de la misma. Para la eliminación de la tabla es necesario utilizar un verbo del *DDL* de *SQL*, éste es el verbo *DROP*[15]. El formato general de esta operación en *SQL* es:

```
DELETE [FROM] NombreTabla
   [WHERE Condición];
```

Cap04ej18.sql

Borrar todos los estancos de la localidad y provincia de Madrid.

Se realiza el borrado de aquellas tuplas de la tabla Estancos que satisfacen los requisitos del enunciado.

```
DELETE
   FROM Estancos
   WHERE localidad_estanco = 'Madrid'
         AND provincia_estanco = 'Madrid';
```

[15] Este verbo, aplicado con sus diferentes operandos, es utilizado por *Oracle* para la eliminación de cualquier objeto del esquema.

Cap04ej19.sql

Borrar todos los estancos que hayan realizado menos de 500 operaciones de venta.

Se obtienen de la tabla Ventas aquellos estancos que han realizado menos de 500 ventas, y se borrarán todas aquellas tuplas de la tabla Estancos cuyo identificador del estanco se encuentre en la lista de tuplas obtenidas en la subconsulta anterior.

```
DELETE
   FROM Estancos
   WHERE nif_estanco IN
     (SELECT DISTINCT nif_estanco
      FROM Ventas
      GROUP BY nif_estanco
        HAVING COUNT(*) < 500);
```

Cap04ej20.sql

Borrar todos los fabricantes que no manufacturen cigarrillos de clase 'UltraLight'.

De la tabla Cigarrillos se obtienen todas las marcas de los mismos para las cuales existen cigarrillos de la clase indicada en el enunciado y, de la tabla Manufactura, se obtienen los fabricantes que manufacturan los mismos. Por último, se borran aquellas tuplas de la tabla Fabricantes que satisfacen la subconsulta anterior.

```
DELETE
   FROM Fabricantes
   WHERE nombre_fabricante NOT IN
     (SELECT DISTINCT nombre_fabricante
      FROM Manufactura
      WHERE marca IN
        (SELECT marca
           FROM Cigarrillos
           WHERE clase = 'UltraLight'));
```

Cap04ej21.sql

Borrar todas las compras que se hayan hecho en el día actual para aquellos cigarrillos que no se hayan vendido en el mismo día.

Se obtienen de la tabla Ventas todos aquellos tipos de cigarrillos que han sido vendidos en el día de hoy para, con esta información, borrar todas aquellas tuplas de la tabla Compras correspondientes a cigarrillos cuyo tipo no se encuentra entre la solución de la subconsulta anterior.

```
DELETE
   FROM Compras
   WHERE fecha_compra = SYSDATE
         AND (nif_estanco, marca, filtro, color, clase,
              mentol) NOT IN
     (SELECT DISTINCT (nif_estanco, marca, filtro, color,
                       clase, mentol)
      FROM Ventas
      WHERE fecha_venta = SYSDATE);
```

Cap04ej22.sql

Borrar de los almacenes correspondientes todos los cigarrillos para los que no se han realizado ventas.

Se borran de la tabla Almacenes todas aquellas tuplas correspondientes a tipos de cigarrillos que no se encuentran en la solución de una subconsulta total realizada sobre la tabla Ventas.

```
DELETE
   FROM Almacenes
   WHERE (nif_estanco, marca, filtro, color, clase,
          mentol) NOT IN
     (SELECT DISTINCT (nif_estanco, marca, filtro, color,
                       clase, mentol)
       FROM Ventas);
   /* También puede realizarse esta operación de la forma
   siguiente */
DELETE
   FROM Almacenes A
   WHERE NOT EXISTS
     (SELECT *
      FROM Ventas V
      WHERE A.nif_estanco = V.nif_estanco AND
            A.marca = V.marca AND A.filtro = V.filtro AND
            A.color = V.color AND A.clase = V.clase  AND
            A.mentol = V.mentol);
```

4.3. NOCIONES SOBRE PROGRAMACIÓN CON PL/SQL

Habrá observado el lector que *SQL* no es un lenguaje de programación propiamente dicho, sino simplemente un lenguaje de acceso a bases de datos puesto que no incorpora construcciones de control que permitan representar el flujo del control y de la información a través de un procedimiento software. *Oracle* cuenta con un lenguaje de programación que engloba al *SQL* para interactuar con la base de datos denominado *PL/SQL*, mediante el cual se pueden construir poderosas aplicaciones software.

En esta sección realizaremos una breve descripción del lenguaje y de los componentes del mismo, describiendo la estructura clásica de los programas *PL/SQL*. Se trata de unas nociones básicas que permitirán al lector comprender los ejemplos propuestos a lo largo de la obra e iniciarse en la programación de aplicaciones con *PL/SQL*.

4.3.1. Estructura de los programas

Los programas *PL/SQL* se estructuran agrupando las declaraciones o cláusulas del lenguaje en distintos bloques o secciones encargadas de tareas específicas, como son:

1. **Declaración de las variables** de la interfaz del programa y de las variables internas de ejecución. Se pueden utilizar todos los tipos de datos definidos por *Oracle* y tipos de datos definidos por el usuario para definir las variables y asignar valores a las mismas.
2. **Flujo de control y operación** de las tareas para las cuales se construye el programa. Se hace uso de los verbos de control del flujo y de sentencias *SQL*.
3. **Control de las excepciones** o errores que se puedan producir durante la ejecución del programa. Se hace uso de las sentencias de control del flujo y de control de errores que permiten la captura de los mismos cuando éstos se producen, y la toma de decisión para su recuperación.

Cada una de estas secciones viene identificada por una cláusula de cabecera de forma que la estructura general de un procedimiento es la siguiente:

```
DECLARE
   Declaración de las variables utilizadas
BEGIN
   Cuerpo del procedimiento
EXCEPTION
   Cuerpo del control de las excepciones
END
```

4.3.2. Procedimientos y Funciones

Como en cualquier lenguaje de programación *PL/SQL* permite la declaración de subprogramas (subrutinas) denominados procedimientos y funciones. Tanto los procedimientos como las funciones pueden ser invocados desde un programa *PL/SQL* u otro procedimiento o función.

Una función *PL/SQL* tiene como objetivo realizar un proceso y devolver un valor al programa o subprograma (procedimiento o función) que la invocó, mientras que un procedimiento tiene como objetivo el realizar un proceso no devolviendo ningún valor. Tanto los procedimientos como las funciones permiten la declaración de una interfaz mediante la cual se les puede transferir una lista de valores. La estructura general de un procedimiento y una función *PL/SQL* son las siguientes:

```
PROCEDURE NombreProcedimiento
   [Variable [IN|OUT|IN OUT] TipoDeDatos [,...]]
   Declaraciones de asignación
IS [|AS]
 BEGIN
   Cuerpo del procedimiento
END;
```

```
FUNCTION NombreFunción
   [Variable [IN|OUT|IN OUT] TipoDeDatos [,...]]
 RETURN TipoDeDatos
   Declaraciones de variables internas
IS [|AS]
 BEGIN
   Cuerpo de la función
END;
```

Oracle reconoce dos formas diferentes de construir programas *PL/SQL*: bloques anónimos y bloques almacenados.

Un bloque anónimo es un programa que tiene existencia para la base de datos únicamente mientras éste se está ejecutando. Los bloques anónimos pueden ser almacenados o no en ficheros de tipo texto, o formar parte de otros programas, y son reconocidos por el gestor de la base de datos cuando se instancia su ejecución.

Los bloques almacenados son programas *PL/SQL* que se almacenan en el núcleo de la base de datos y que pueden ser consultados, accedidos, modificados y ejecutados cuantas veces se desee. En este caso estos programas forman parte de la base de datos al igual que cualquier otro objeto (tabla, índice, atributo, etc.).

La sintaxis descrita anteriormente es la utilizada para la definición de bloques anónimos. En el caso en que se desee construir un bloque almacenado basta con añadir la cláusula **CREATE OR REPLACE** antes de la definición del procedimiento o de la función.

4.3.3. Declaración de Variables

La declaración de las variables internas del programa *PL/SQL*, así como las variables utilizadas en la interfaz conforma la primera sección o bloque de todo programa *PL/SQL*. La declaración de variables se realiza asignando un tipo de datos a una variable y opcionalmente un valor por defecto, en la forma:

```
Variable TipoDeDatos [DEFAULT := valor]
```

PL/SQL introduce dos tipos de datos especiales para la definición de las variables con el fin de garantizar la correcta definición de las mismas. Éstos son:

%TYPE: cuya función es la de asignar a la variable el mismo dominio en el cual se encuentra definido un atributo existente en la base de datos. Una definición de la forma:

```
Var-1 NombreTabla.nombreatributo%TYPE
```

define una variable *Var-1* en el mismo dominio en que fue definido el atributo *nombreatributo* en la tabla *NombreTabla*.

%ROWTYPE: cuya función es la de asignar a la variable un dominio singular de *Oracle* que permite representar variables de tipo registro mediante las cuales, por ejemplo, se puede representar a las tuplas de una tabla.

4.3.3.1. TIPOS DE REGISTRO Y CURSORES

En los programas *PL/SQL* es preciso en muchas ocasiones hacer uso de unos tipos especiales de variables denominadas *tipo de registro* y *cursor*. Un registro es una estructura de datos abstracta definida por el usuario, consistente en un conjunto de campos o ítems relacionados que tienen un nombre único y están definidos en un dominio permitido por *Oracle*, y cuya finalidad es representar una abstracción de datos que es utilizada durante la ejecución del programa *PL/SQL*. La sintaxis general para la definición de un tipo de registro es:

```
TYPE NombreTipoRegistro IS RECORD (
   campo-1 TipoDeDatos [NOT NULL]
           [DEFAULT := expresión],
   [, otra definición de campos]
);
```

Una vez definido un tipo de registro, éste es reconocido por *Oracle* como un nuevo tipo de datos, lo cual permite que en la definición de variables se pueda utilizar el dominio definido por un tipo de registro generado por el usuario, en la forma:

```
Var-1 NombreTipoRegistro
```

definiéndose una variable *Var-1* con una estructura igual a la definida para el tipo de registro *NombreTipoRegistro*.

Cuando se ejecuta una sentencia *SQL* en un programa, *Oracle* genera automáticamente un cursor. Un cursor es un área de memoria de trabajo en la que se almacena la información de salida de la sentencia *SQL* y que es necesario declarar para posteriormente poder acceder a las tuplas que componen la salida de la sentencia. La sintaxis general para la declaración de un cursor es la siguiente:

```
CURSOR NombreCursor [(lista-parámetros)] IS
  Sentencia SELECT;
```

La *lista-parámetros* es una lista de declaración de variables que pueden ser utilizadas posteriormente en el cuerpo de la sentencia *SELECT* que define el cursor.

4.3.4. Construcciones de Control

PL/SQL incorpora las construcciones de control características de cualquier lenguaje de programación, las cuales simplemente señalaremos. Éstas son:

CONSTRUCCIONES DE CONTROL	
IF, ELSE, ELSEIF, END IF	Instrucciones de bifurcación condicionada
GOTO	Instrucció de bifurcación no condicionada
LOOP, END LOOP, EXIT WHEN	Instrucción de repetición no condicionada
WHILE, END LOOP	Instrucciones de repetición condicionada
FOR LOOP, END LOOP	

4.3.5. Control de las Excepciones

Durante la ejecución de un programa pueden producirse errores o excepciones previstas, o no, que deben ser controlados bien para su reparación, o bien para una terminación controlada del programa.

PL/SQL incorpora un bloque para el manejo de excepciones constituido por una rutina de manejo de excepciones que es invocada cuando se produce una excepción. Una excepción es una condición de error con nombre, que es disparada cuando se detecta un error en la ejecución del programa y produce que el flujo de control del mismo se transfiera al cuerpo o sección de manejo de excepciones. La sintaxis general de la sección de excepciones es la siguiente:

```
EXCEPTION
   WHEN NombreExcepción THEN
      CuerpoDeRecuperación;
   WHEN OTHERS THEN
      CuerpoDeRecuperación;
END
```

Las excepciones pueden ser definidas por el usuario en la declaración de variables mediante la sentencia:

```
NombreExcepción EXCEPTION
```

y pueden ser capturadas en la sección de control o en la propia sección de excepciones mediante la sentencia siguiente:

```
RAISE NombreExcepción
```

4.3.6. Un ejemplo de programa PL/SQL

Vamos a introducir un ejemplo de un procedimiento anónimo *PL/SQL* sobre el problema de los cigarrillos para una mejor comprensión del lenguaje.

No se preocupe el lector por aquellas sentencias que no comprenda inicialmente. Preste su atención en la estructura del procedimiento y en el flujo de control; conforme avance en el estudio de los demás capítulos se familiarizará con el lenguaje a través de los ejemplos introducidos en los mismos.

Cap04ejP1.sql

Modificar el precio de venta de los cigarrillos, según la siguiente regla. El precio de venta debe ser un 20% mayor que el precio de coste para los cigarrillos nacionales y un 30% mayor para los extranjeros.

Se abre un cursor (nuevo_precio) que contiene los atributos marca, filtro, color, clase, mentol, precio_costo y el país de sede del fabricante de la marca. Para cada cigarrillo se actualiza el atributo precio_venta con un incremento del 20% del precio_costo si la marca es nacional y con un 30% si la marca es extranjera.

```
DECLARE
   /* Se define la variable de trabajo y el cursor */
   incremento_costo NUMBER;
   CURSOR nuevo_precio IS
     SELECT Cigarrillos.marca, Cigarrillos.precio_costo,
            Fabricantes.pais, Cigarrillos.filtro,
            Cigarrillos.color, Cigarrillos.clase,
            Cigarrillos.mentol
        FROM Cigarrillos , Fabricantes, Manufactura
        WHERE Manufactura.nombre_fabricante =
              Fabricantes.nombre_fabricante AND
              Cigarrillos.marca=manufactura.marca;
BEGIN
   /* Se abre un bucle para recorrer el cursor */
   FOR precio_record IN nuevo_precio LOOP
   /* Se asigna el valor del incremento en función del país */
     IF precio_record.pais = 'España' THEN
       incremento_costo := 0.2;
     ELSE
       incremento_costo := 0.3;
     END IF;
   /* Se actualiza la tupla correspondiente en la tabla */
     UPDATE Cigarrillos
     SET precio_venta=precio_record.precio_costo +
        (incremento_costo * precio_record.precio_costo)
      WHERE Cigarrillos.marca = precio_record.marca AND
            Cigarrillos.filtro = precio_record.filtro AND
            Cigarrillos.color = precio_record.color AND
            Cigarrillos.clase = precio_record.clase AND
            Cigarrillos.mentol = precio_record.mentol;
   END LOOP;
   /* Se actualiza la base de datos */
COMMIT;
   /* Se controlan los errores que pudieran ocurrir */
EXCEPTION
   WHEN OTHERS THEN
   /* Se hace uso de la función PUT_LINE del paquete
   DBMS_OUTPUT proporcionado por Oracle para presentar un
   mensaje en pantalla */
     DBMS_OUTPUT.PUT_LINE(SQLERRM);
   /* En caso de error se cancela la actualización */
   ROLLBACK;
END;
/
```

CAPÍTULO 5

TRADUCCIÓN DE ESQUEMAS E-R A ESQUEMAS RELACIONALES

En este capítulo se van a presentar una serie de reglas fáciles de entender y de aplicar para el proceso de traducción de los esquemas conceptuales realizados mediante el modelo entidad-interrelación a esquemas lógicos basados en el modelo relacional.

El proceso de traducción de esquemas conceptuales a lógicos consiste en la aplicación, por pasos, de una serie de reglas que, aplicadas a los esquemas conceptuales, transforman los objetos de estos esquemas en objetos pertenecientes a los esquemas lógicos. En el caso que nos ocupa, la aplicación de las reglas va a dar lugar a la transformación de los tipos de entidad y los tipos de interrelación que forman parte de los esquemas conceptuales, en tablas o relaciones, los únicos objetos que intervienen en los esquemas lógicos relacionales.

5.1. PREPARACIÓN DE LOS ESQUEMAS CONCEPTUALES

Como paso previo a la aplicación de las reglas de transformación de esquemas conceptuales a esquemas relacionales, las cuales denominaremos abreviadamente **RTECAR**, es conveniente la preparación de los esquemas conceptuales mediante la aplicación de unas reglas preparatorias (a las que nos referiremos por **PRTECAR**) que faciliten y garanticen la fiabilidad del proceso de transformación.

Las *PRTECAR* se basan en la aplicación de la primera forma normal a los objetos que forman parte de los esquemas conceptuales. Si consideramos que tanto los

tipos de entidad como de interrelación pueden ser representados por tablas bidimensionales (al igual que en el modelo relacional), el principio en el que se basan estas reglas es el de eliminar de estos tipos de objetos las siguientes anomalías:

1. Aquellos atributos correspondientes a los tipos de entidad e interrelación que presenten valores múltiples.
2. Aquellos atributos correspondientes a los tipos de entidad e interrelación que sean compuestos.

El primer punto consiste en considerar que los atributos existentes en el esquema conceptual sólo pueden tomar valores atómicos (al igual que propone la *FN1*). Mientras que el segundo punto hace referencia a la eliminación de agregados de datos o atributos formados por otros atributos más simples.

5.1.1. Eliminación de atributos múltiples

El proceso de eliminación de los atributos múltiples es un proceso muy simple y consistente en la aplicación de la siguiente regla:

> **PRTECAR-1:** *Todos los atributos múltiples; es decir, los atributos que pueden tomar más de un valor en el dominio en el cual están definidos, se transformarán en un tipo de entidad débil por existencia el cual mantendrá una relación:*
>
> - uno a muchos si el atributo es un identificador alternativo en el tipo de entidad en el que estaba presente, o
> - muchos a muchos en caso contrario,
>
> *con el tipo de entidad sobre el cual estaba definido o los tipos de entidad que mantenían un tipo de interrelación si el atributo múltiple estaba definido sobre el tipo de interrelación.*
>
> *Este tipo de entidad débil, creado por la aplicación de esta regla, tendrá como propiedades el atributo por el cual la regla se ha aplicado, y cualquier otra que se considere necesario añadir. Además, se deberá tener en cuenta que si el atributo del tipo de entidad débil creado no pudiera identificar sin ambigüedad a las entidades de este tipo, entonces se procederá de alguna de las dos formas siguientes:*
>
> 1. *El tipo de entidad débil creado se considera que es débil por identificación con respecto al tipo de entidad con el que mantiene relación (siempre en relaciones uno a muchos, evidentemente), heredando, por tanto, sus atributos identificadores.*
> 2. *Se añadirá un nuevo atributo (externo, o no, al dominio del problema) que permita identificar sin ambigüedad a las entidades de este tipo de entidad débil.*

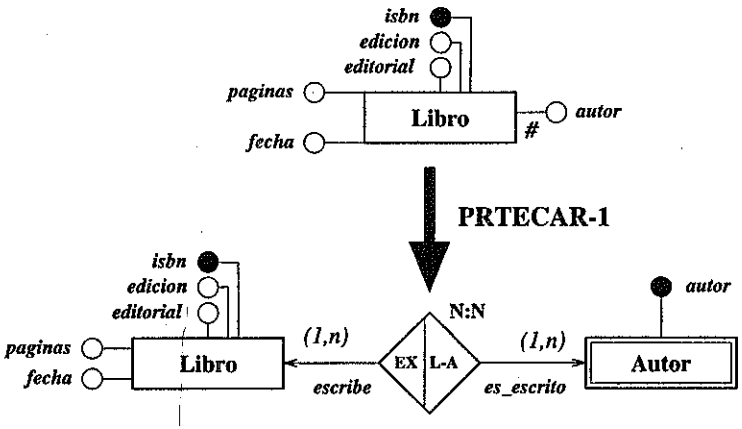


Figura 5.1 Eliminación de los atributos múltiples

En el diagrama de la figura 5.1 se muestra el tipo de entidad *Libro*, en el cual existe un atributo múltiple, el atributo *autor*, representando que un libro puede ser escrito por más de un autor.

Por aplicación de la regla *PRTECAR-1*, el atributo *autor* se ha convertido en un tipo de entidad débil por existencia, con respecto al tipo de entidad *Libro*, denominado *Autor*. El tipo de entidad *Autor* tiene como único atributo *autor*, el cual pertenecía al tipo de entidad *Libro* antes de la aplicación de la regla.

En este ejemplo, se ha considerado que este atributo puede ser utilizado como atributo identificador de las entidades del tipo de entidad *Autor*. En caso contrario, se podría:

1. Considerar el tipo de entidad *Autor* débil por identificación con respecto al tipo de entidad *Libro* (manteniendo un tipo de interrelación uno a muchos), en cuyo caso las entidades de este tipo vendrían identificadas por la concatenación de los atributos *isbn* (heredado del tipo de entidad *Libro*), y *autor* (correspondiente al tipo de entidad *Autor*).
2. Introducir un nuevo atributo, por ejemplo, un código que identifique a cada autor y que permita la identificación sin ambigüedad de las entidades de este tipo.

5.1.2. Eliminación de atributos compuestos

El segundo proceso preparatorio de los esquemas conceptuales para su transformación en esquemas lógicos relacionales consiste en la descomposición de todos los atributos compuestos en atributos simples, por aplicación de la siguiente regla:

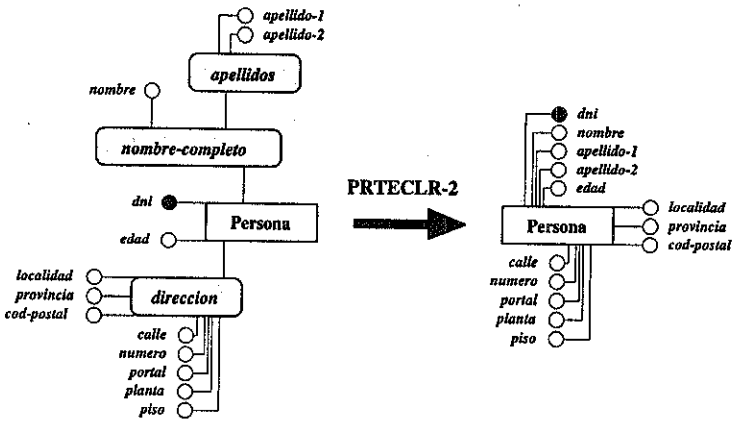


Figura 5.2 Eliminación de los atributos compuestos

PRTECAR-2: *Todos los atributos compuestos asociados con los tipos de entidad y los tipos de interrelación deben ser descompuestos en los atributos simples que formen parte o intervengan en los atributos compuestos correspondientes. En este proceso de descomposición, se eliminará el atributo compuesto, quedando los atributos simples definidos en el mismo dominio, e interviniendo de la misma forma en el tipo de entidad o interrelación.*

Se puede apreciar que se trata de una regla muy simple de aplicar, consistente simplemente en la no-consideración de agregados de datos como atributos en los esquemas conceptuales.

En la figura 5.2 se muestra un ejemplo de aplicación de la regla *PRTECAR-2*, aplicada al tipo de entidad *Persona*. Este tipo de entidad viene definido por un conjunto de atributos, dos de los cuales, el atributo *direccion* y el atributo *nombre-completo*, son atributos compuestos.

El atributo *direccion* está compuesto por los atributos *calle, numero, portal, planta, piso, cod-postal, localidad* y *provincia*. Un conjunto de atributos que son referenciados por un único atributo o agregado, y el atributo *nombre-completo* está compuesto por los atributos *nombre* y un nuevo atributo compuesto denominado *apellidos*, el cual, a su vez, está compuesto por otros dos atributos, *apellido-1* y *apellido-2*. Después de aplicar esta regla (como se muestra en la figura 5.2), se han eliminado los atributos *direccion, nombre-completo* y *apellidos*, quedando el tipo de entidad *Persona* definido únicamente en función de atributos simples.

5.2. TRANSFORMACIÓN DE LOS ESQUEMAS CONCEPTUALES

A continuación se van a presentar una serie de reglas **(RTECAR)** para la transformación de los objetos presentes en los esquemas conceptuales a objetos válidos en los esquemas lógicos relacionales. Estas reglas permiten la transformación de los esquemas de uno a otro tipo sin pérdida de información y, por tanto, conservando el nivel de representación del problema.

La aplicación de las reglas va a depender de:

1. El tipo de objeto del esquema conceptual que se debe transformar.
2. La cardinalidad de las relaciones que los objetos mantienen con otros objetos en el esquema conceptual.

5.2.1. Transformación de tipos de entidad

Se trata de un proceso muy simple y consistente en aplicar la siguiente regla:

RTECAR-1: *Todos los tipos de entidad presentes en el esquema conceptual se transformarán en tablas o relaciones en el esquema relacional manteniendo el número y tipo de atributos, así como la característica de identificador de estos atributos.*

Por ejemplo, en el caso del esquema representado en la figura 5.1, una vez aplicada esta regla se obtendrá el siguiente esquema relacional[16].

Libro *(isbn, edicion, editorial, paginas, fecha)*
Autor *(autor)*

en el que no se ha considerado la transformación que es necesario realizar para el tipo de interrelación, la cual se describirá en las secciones siguientes.

5.2.2. Transformación de tipos de interrelación uno a uno

El proceso de transformación de los tipos de interrelación binarias en los cuales los tipos de entidad participan con cardinalidad máxima uno va a depender del valor de la cardinalidad mínima con la cual participa cada tipo de entidad en el tipo de interrelación. En base a este valor se pueden presentar los siguientes casos:

1. Los dos tipos de entidad participan de forma completa en el tipo de interrelación. Es decir, los dos tipos de entidad participan con cardinalidad mínima uno.
2. Uno de los dos tipos de entidad participa de forma parcial en el tipo de interrelación; es decir, con cardinalidad mínima cero.

[16] Como norma general, los atributos claves de las tablas se representarán subrayados, las claves alternas con doble subrayado, y las claves foráneas en negrita.

3. Los dos tipos de entidad participan de forma parcial en el tipo de interrelación.

Por tanto, dependiendo de la forma en que participa cada tipo de entidad en el tipo de interrelación será necesario aplicar un criterio de transformación diferente.

RTECAR-2.1: *Si en un tipo de interrelación binaria los dos tipos de entidad participan de forma completa; es decir, ambos tipos de entidad participan con las cardinalidades mínima y máxima igual a uno, entonces:*

- *Si los dos tipos de entidad tienen el mismo identificador:*
  1. *Los dos tipos de entidad se transforman en una única tabla formada por la agregación de los atributos de los dos tipos de entidad.*
  2. *La clave de la tabla es el identificador de los tipos de entidad (es el mismo en ambas).*
- *Si los tipos de entidad tienen diferente identificador, cada tipo de entidad se transforma en una tabla y,*
  1. *Cada tabla tendrá como clave principal el identificador de cada uno de los tipos de entidad de los cuales se deriva.*
  2. *Cada tabla tendrá como clave foránea el identificador del otro tipo de entidad con el cual está relacionado.*

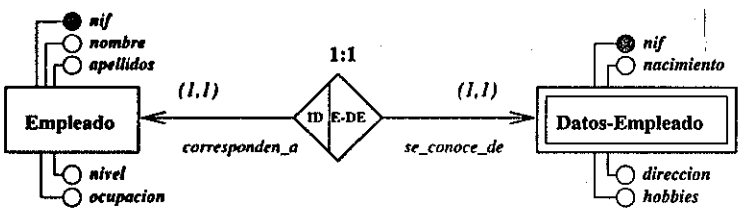


Figura 5.3 Relaciones binarias uno a uno. ***Regla:*** RTECAR-2.1

- *Si los dos tipos de entidad tienen el mismo identificador, pero uno de ellos es un tipo de entidad débil, entonces se procede de alguna de las dos formas expuesta anteriormente en función de los requisitos funcionales.*

Las relaciones uno a uno pueden representar situaciones muy diferentes dependiendo del problema que se esté tratando. En ocasiones representan relaciones de debilidad, de existencia generalmente (la consideración en tipos de entidades diferentes de un conjunto de características de una clase de objeto del problema) y, en otros casos, simplemente una relación fuerte entre diferentes tipos de entidad. En función del caso del que se trate se tomará una acción en la derivación del esquema relacional u otra.

Tomaremos, para la aplicación de esta regla el ejemplo propuesto en la figura 5.3, en este caso se puede derivar el siguiente esquema relacional:

Empleado *(nif, nombre, apellidos, nivel, ocupacion, direccion, nacimiento, hobbies)*

En este caso, ambos tipos de entidad representan a los empleados de una determinada empresa, pero ha sido conveniente descomponer las propiedades o características de este tipo de objeto en dos tipos de entidad. El tipo de entidad *Empleado* tiene aquellos atributos que son requeridos y utilizados para el conocimiento de los empleados desde el punto de vista operacional de la empresa, mientras el tipo de entidad *Datos-Empleado* tiene aquellos atributos que simplemente informan de algunos detalles particulares de las entidades de este tipo.

Se puede observar que se trata de un tipo de interrelación débil por identificación. El tipo de entidad *Datos-Empleado* es un tipo de entidad débil por identificación con respecto al tipo de entidad *Empleado*, puesto que no existirán datos personales de los empleados, ni se podrán identificar, si no existe un empleado.

Puede ser también necesario considerar, por motivos de procesamiento, más conveniente la existencia de tablas independientes para cada tipo de entidad. Entonces, el esquema relacional resultante quedaría de la forma:

Empleado *(nif, nombre, apellidos, nivel, ocupacion)*
Datos-Empleado *(nif, direccion, nacimiento, hobbies)*

Los tipos de interrelaciones uno a uno completas no suelen tener atributos cualificadores asociados, puesto que éstos pueden considerarse pertenecientes a alguno de los tipos de entidad que participan en el tipo de interrelación.

RTECAR-2.2: *Si en un tipo de interrelación binaria alguno de los tipos de entidad participa de forma parcial, entonces, cada tipo de entidad se transforma en una tabla por aplicación de la regla RTECAR-1 y se procede de alguna de las dos formas siguientes:*

1. *El identificador del tipo de entidad que participa de forma total pasa como atributo de la tabla correspondiente a la transformación del otro tipo de entidad. Este atributo será definido como clave alterna y foránea, no pudiendo tomar valores nulos para las diferentes tuplas de esta tabla, y no se genera ninguna tabla para el tipo de interrelación.*

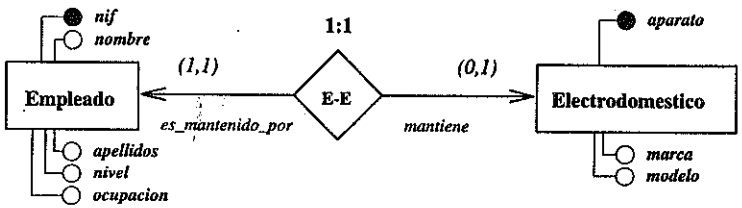


Figura 5.4 Relaciones binarias uno a uno. ***Regla:*** RTECAR-2.2

2. *Se construye una nueva tabla correspondiente al tipo de interrelación formada por los atributos identificadores de los dos tipos de entidad. Los atributos serán claves foráneas de las tablas correspondientes a la transformación de los tipos de entidad, y serán definidos como claves candidatas.*

La figura 5.4 muestra un ejemplo de aplicación de esta regla. Se ha considerado el tipo de entidad *Empleado* que mantiene una relación con el tipo de entidad *Electrodomestico*. El tipo de entidad *Empleado* participa de forma completa en el tipo de interrelación, de forma que un empleado puede mantener a ninguno o un electrodoméstico mientras que un electrodoméstico debe ser mantenido siempre por un solo empleado.

Por aplicación de esta regla el esquema relacional puede quedar de alguna de las dos formas siguientes:

1. No se construye una tabla para el tipo de interrelación, entonces:

   Empleado *(nif, nombre, apellidos, nivel, ocupacion)*
   Electrodomestico *(aparato, marca, modelo,* **nif***)*

   el identificador del tipo de entidad *Empleado* pasa a formar parte de la tabla correspondiente a la transformación del tipo de entidad *Electrodomestico*.

2. Se construye una tabla correspondiente a la transformación del tipo de interrelación:

   Empleado *(nif, nombre, apellidos, nivel, ocupacion)*
   Electrodomestico *(aparato, marca, modelo)*
   Emp-Elec *(***aparato, nif***)*

   en la que, en este ejemplo, la clave será el identificador del tipo de entidad *Electrodomestico* que participa de forma parcial en el tipo de interrelación.

Si analizamos las dos transformaciones posibles propuestas se puede observar:

- Que la primera transformación es la más favorable debido a que en ningún momento existirán atributos con valores nulos. El atributo *nif* que forma parte de la tabla *Electrodomestico* siempre tendrá algún valor, debido a que todos los

electrodomésticos son mantenidos por al menos uno y como máximo un empleado.

Si se hubiera realizado el proceso inverso, es decir, el identificador del tipo de entidad *Electrodomestico* hubiera pasado a formar parte de la tabla correspondiente al tipo de entidad *Empleado*, se hubiera obtenido un esquema relacional de la forma:

Empleado *(nif, nombre, apellidos, nivel, ocupacion,* **aparato***)*
Electrodomestico *(aparato, marca, modelo)*

en el que se puede observar que en la tabla *Empleado*, el atributo *aparato* tomará valores nulos para todas aquellas tuplas de empleados que no mantienen ningún aparato, no pudiéndose definir como clave alterna, lo cual viola los requisitos del problema.

Por tanto, en el proceso de transformación de relaciones *uno a uno semi-parciales*, como es el caso de aplicación de la regla *RTECAR-2.2*, cada tipo de entidad se transforma en una tabla y el identificador del tipo de entidad que participa de forma total entra a formar parte, como un atributo más, de la tabla correspondiente al tipo de entidad que participa de forma parcial. Además, este atributo será definido como clave alterna y clave foránea de la tabla correspondiente al tipo de entidad que participa de forma total.

- La segunda transformación, en la cual se crea una tabla que mantiene la relación entre los dos tipos de entidad, tiene el inconveniente de un esquema más grande en el que los procedimientos tendrán que manejar una tabla que no aporta más información o una mejor representación del problema.

En el caso en que la relación estuviera cualificada por atributos, para la primera transformación propuesta éstos pasarían a formar parte de la tabla correspondiente al tipo de entidad que participa de forma parcial (junto con la clave de la tabla correspondiente al tipo de entidad que participa de forma total). Y para la segunda transformación propuesta, pasarían a formar parte de la tabla correspondiente a la transformación del tipo de interrelación.

Por ejemplo, si la relación entre el tipo de entidad *Empleado* y *Electrodomestico* estuviera cualificada por el atributo *ultima-fecha*, que representa la última fecha en que el empleado trabajó con el electrodoméstico, el esquema relacional obtenido después de la aplicación de esta regla quedaría:

Empleado *(nif, nombre, apellidos, nivel, ocupacion)*
Electrodomestico *(aparato, marca, modelo,* **nif***, ultima-fecha)*

Por otra parte, si los atributos identificadores son los mismos para los dos tipos de entidad que participan en este tipo de interrelación se podría aplicar la regla *RTECAR-2.1*, construyéndose una única tabla formada por la agregación de los atributos de los dos tipos de entidad. El inconveniente que se presenta, en este caso, es

que podrían existir muchos valores nulos en los atributos de las tuplas o filas de esta tabla.

PARTICIPACIÓN PARCIAL EN LAS DOS ENTIDADES

RTECAR-2.3: *Si en un tipo de interrelación binaria ambos tipos de entidad participan de forma parcial entonces, y por aplicación de la regla RTECAR-1, cada uno de ellos se transforma en una tabla y se procede de la forma siguiente: Se construye una nueva tabla correspondiente al tipo de interrelación y cuyos atributos serán los identificadores de los dos tipos de entidad, los cuales serán definidos como claves foráneas. La clave principal de la tabla generada será el identificador de uno de los tipos de entidad y, necesariamente, se definirá como clave alterna al identificador del otro tipo de entidad.*

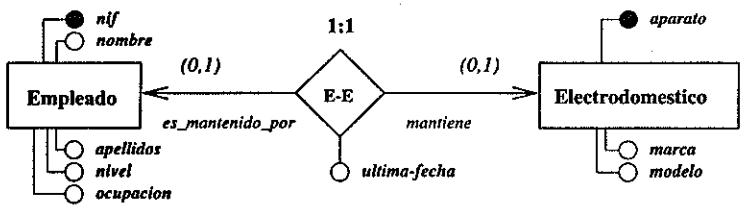


Figura 5.5 Relaciones binarias uno a uno. ***Regla:*** RTECAR-2.3

En la figura 5.5 se representa un ejemplo, para la aplicación de esta regla, en el que se muestra la relación que existe entre los tipos de entidad *Empleado* y *Electrodomestico*. Ambos tipos de entidad participan de forma parcial en el tipo de interrelación y, además, el tipo de interrelación tiene asociado un atributo que informa de la última fecha en que un empleado tuvo relación con un electrodoméstico.

Para este ejemplo, el esquema relacional que se obtiene después de la transformación será:

Empleado *(nif, nombre, apellidos, nivel, ocupacion)*
Electrodomestico *(aparato, marca, modelo)*
Empl_Elect *(**nif**, **aparato**, ultima_fecha)*

en el que se ha tomado el criterio de que el atributo *nif* sea la clave principal y el atributo *aparato* la clave secundaria de la tabla *Empl_Elect* (aunque podía haberse tomado el criterio contrario), siendo ambos atributos claves foráneas en esta tabla y, por tanto, manteniendo referencias con las claves principales de las tablas *Empleado* y *Electrodomestico*, respectivamente.

Si se hubiera utilizado el criterio de la *RTECAR-2.2* en el cual no se crea una tabla para el tipo de interrelación, se tendría:

Empleado *(nif, nombre, apellidos, nivel, ocupacion,* **aparato***)*
Electrodomestico *(aparato), marca, modelo,* **nif**, *ultima-fecha)*

Que como se aprecia es una derivación inadecuada debido a la existencia de numerosos valores nulos para los atributos en la extensión del esquema, y a no poder definir los atributos *nif* y *aparato* como claves alternas además de foráneas, lo que viola los requisitos del problema planteado.

5.2.3. Transformación de tipos de interrelación uno a muchos

La transformación de esquemas conceptuales en los que estén presentes tipos de interrelaciones binarias en los que un tipo de entidad participa con cardinalidad máxima uno y el otro tipo de entidad participa con cardinalidad máxima muchos, es muy similar a las transformaciones descritas anteriormente. Y, al igual que en ellas, se aplicarán reglas diferentes en función de las cardinalidades mínimas con las cuales participa cada tipo de entidad en el tipo de interrelación.

RTECAR-3.1: *Si en un tipo de interrelación binaria 1:N ambos tipos de entidad participan de forma total, o el tipo de entidad que interviene con cardinalidad máxima muchos participa de forma parcial, entonces, cada tipo de entidad se transforma en una tabla por aplicación de la regla RTECAR-1, y el identificador del tipo de entidad que participa con cardinalidad máxima uno pasa a formar parte de la tabla correspondiente al tipo de entidad que participa con cardinalidad máxima muchos. Este atributo será definido como clave foránea de esta tabla (no pudiendo tomar valores nulos) manteniendo una referencia con la tabla correspondiente al tipo de entidad que participa con cardinalidad máxima uno. Si el tipo de interrelación tuviera atributos asociados, estos atributos pasan a formar parte de la tabla correspondiente al tipo de entidad que participa con cardinalidad máxima muchos.*

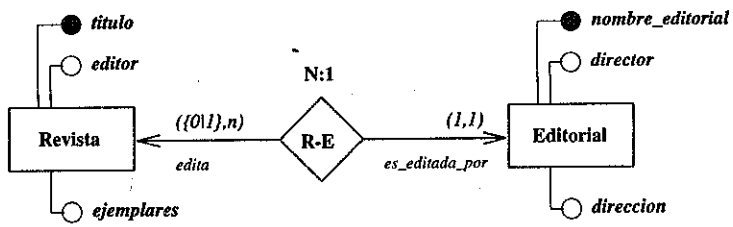


Figura 5.6 Relaciones binarias uno a muchos. ***Regla:*** RTECAR-3.1

En la figura 5.6 se muestra la relación *1:N* existente entre los tipos de entidad *Revista* y *Editorial*. En este esquema conceptual se muestra la relación existente entre una editorial que puede publicar de cero o una hasta muchas revistas, mientras que una revista es publicada por una sola editorial. Aplicando a este ejemplo la regla *RTECAR-3.1* se obtiene el siguiente esquema relacional:

Editorial *(nombre_editorial, direccion, director)*
Revista *(titulo, editor, ejemplares,* **nombre_editorial***)*

RTECAR-3.2: *Si en un tipo de interrelación binaria 1:N ambos tipos de entidad participan de forma parcial, o únicamente el tipo de entidad que interviene con cardinalidad máxima uno participa de forma parcial, entonces, cada tipo de entidad se transforma en una tabla por aplicación de la regla RTECAR-1 y se genera una nueva tabla correspondiente al tipo de interrelación. Esta tabla estará formada por los identificadores de los tipos de entidad que intervienen en el tipo de interrelación y por todos los atributos asociados al tipo de interrelación. La clave principal de esta tabla será el atributo identificador correspondiente al tipo de entidad que interviene con cardinalidad máxima muchos, y será necesario definir como claves foráneas los atributos identificadores correspondientes a los dos tipos de entidad.*

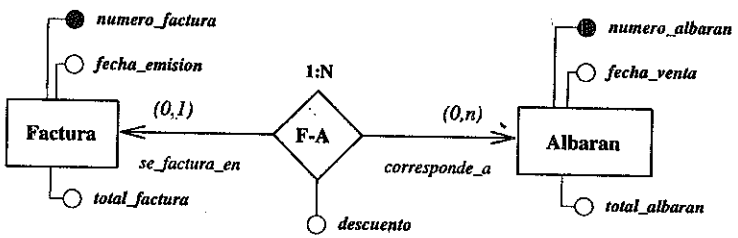


Figura 5.7 Relaciones binarias uno a muchos. ***Regla:*** RTECAR-3.2

Veamos el ejemplo que se muestra en la figura 5.7 en el que se ha representado la relación existente entre los tipos de entidad *Factura* y *Albaran*. El esquema de esta figura representa que una factura es editada para cero o muchos albaranes, mientras que un albarán es facturado en ninguna o una sola factura. Es decir, en el sistema que representa la figura 5.7 se admite la emisión de facturas sin la realización previa de un albarán (una venta directa) y, por supuesto, que los albaranes sólo formen parte de una factura o bien de ninguna debido a que no se expidan facturas para esos albaranes. El tipo de interrelación que relaciona estos dos tipos de entidad está cualificado por el atributo *descuento* que representa el descuento aplicado sobre el total de la cantidad de cada uno de los albaranes en la factura.

Si aplicamos a este esquema conceptual la regla *RTECAR-3.2*, se obtendría el siguiente esquema relacional:

Factura *(numero_factura, fecha_emision, total_factura)*
Albaran *(numero_albaran, fecha_venta, total_albaran)*
Fact_Alba **(número_albaran, numero_factura,** *descuento)*

en el que el atributo *numero_factura* de la tabla *Fact_Alba* no podrá tomar valores nulos para garantizar las características del problema.

Se puede observar que la regla *RTECAR-3.1* puede ser también aplicada a este caso. El que se aplique una u otra regla va a depender, principalmente, del número de

instancias del tipo de interrelación y de los atributos asociados al mismo. Por aplicación de la regla *RTECAR-3.1*, el esquema relacional quedaría de la forma:

Factura *(numero_factura, fecha_emision, total_factura)*
Albaran *(numero_albaran, fecha_venta, total_albaran,* **numero_factura,** *descuento)*

en el que el atributo *numero_factura* de la tabla *Albaran* en este caso podrá tomar valores nulos, siendo menos adecuada esta derivación, aunque en ambos casos describen el problema representado en el esquema conceptual de la figura 5.7.

5.2.4. Transformación de tipos de interrelación muchos a muchos

En esta clase de interrelaciones el proceso de transformación no depende de la cardinalidad mínima con la que interviene cada tipo de entidad en el tipo de interrelación, sino que siempre se aplicará la siguiente regla:

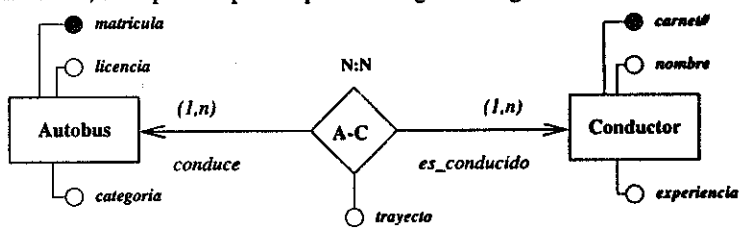


Figura 5.8 Relaciones binarias muchos a muchos. ***Regla:*** RTECAR-4

RTECAR-4: *En un tipo de interrelación binaria N:N cada tipo de entidad se transforma en una tabla por aplicación de la regla RTECAR-1 y se genera una nueva tabla para representar al tipo de interrelación. Esta tabla estará formada por los identificadores de los tipos de entidad que intervienen en el tipo de interrelación y por todos los atributos asociados al tipo de interrelación. La clave principal de esta tabla será la agregación de los atributos identificadores correspondientes a los tipos de entidad que intervienen en el tipo de interrelación.*

La figura 5.8 muestra un ejemplo del tipo de interrelación muchos a muchos. En este caso se muestra la relación que existe entre los autobuses y los conductores de los mismos, en la que un conductor puede conducir varios autobuses y un autobús puede ser conducido por varios conductores. El tipo de interrelación entre los tipos de entidad *Autobus* y *Conductor* tiene asociado el atributo *trayecto* que representa el trayecto o recorrido realizado por un conductor con un autobús determinado. Por aplicación de la regla *RTECAR-4*, el esquema relacional queda de la forma:

Autobus *(matricula, licencia, categoria)*
Conductor *(carnet#, nombre, experiencia)*
Condu_Auto **(carnet#, matricula,** *trayecto)*

5.2.4.1. ORDEN DE LOS ATRIBUTOS EN LAS CLAVES COMPUESTAS

Se ha podido observar que por la aplicación de la regla *RTECAR-4* es necesaria la construcción de una tabla que represente el tipo de interrelación *N:N* que existe entre dos tipos de entidad. Para esta tabla es necesario definir como clave compuesta la formada por la agregación de los atributos identificadores de los tipos de entidad que participan en el tipo de interrelación.

La definición de una clave compuesta supone un problema añadido, no trivial, para el diseñador de la base de datos. En el problema presentado en la figura 5.8 la tabla *Condu_Auto* se podría haber definido de dos formas diferentes:

Condu_Auto *(carnet#, matricula, trayecto)*
Auto_Condu *(matricula, carnet#, trayecto)*

Ambas tablas representan la misma relación entre los tipos de entidad *Autobus* y *Conductor*, tienen la misma estructura —el mismo número de atributos que representan la misma información— y la clave de ambas tablas está formada por la agregación de los mismos atributos pero, sin embargo, estas claves son distintas.

Aunque las claves de las tablas *Condu_Auto* y *Auto_Condu* están formadas por la agregación de los mismos atributos (*carnet#* y *matricula*) el orden de estos atributos es diferente para cada una de las tablas.

Si bien el orden de los atributos que forman parte de una clave compuesta debe ser, y de hecho lo es, transparente para el usuario final, sí es importante para el diseñador de la base de datos y, sobre todo, para el administrador de la misma y la razón la expondremos a continuación.

El que un conjunto de atributos (a veces uno sólo) se defina como clave implica una serie de compromisos que afectan tanto a la visión lógica como física de la base de datos, compromisos entre los que se puede citar (ya se han tratado algunos de ellos):

- Que no puedan existir dos objetos (tuplas de la tabla) con el mismo valor de ellos.
- Que al identificar sin ambigüedad a una tupla, los métodos de acceso a esta información se realizarán, generalmente, basándose en criterios de selección en los cuales sólo intervenga uno o todos los atributos que forman parte de la clave.
- Y, por tanto, que el diseñador establecerá una organización física tal que estos procedimientos de acceso tengan un desempeño aceptable.

Si nos centramos en los aspectos físicos y consideramos que se organiza un índice por la clave para un acceso rápido a la información, el orden de los atributos en este índice es importante. No es lo mismo, ni se obtendrán los mismos resultados, si el índice se define como la agregación de *carnet#* y *matricula*, que si se define como la agregación de *matricula* y *carnet#*. En el primer caso las entradas del índice estarán

ordenadas por los identificadores de los conductores, mientras que en el segundo lo estarán ordenadas por los de los autobuses.

Una solicitud de información ordenada de todos los autobuses o una búsqueda aproximada será muy efectiva (en cuanto a tiempo de respuesta) en el segundo de los casos y no en el primero, y viceversa.

Por todo ello, la disposición de los atributos en las claves compuestas, sean principales o alternas y, por supuesto, en cada uno de los índices que se definan para las tablas de la base de datos, será en parte responsable del desempeño de los procedimientos (algunos de ellos, al menos) que solicite una recuperación de esta información.

El diseñador de la base de datos deberá, por ello, tener en cuenta los requisitos operacionales impuestos y definidos en el análisis de requisitos del sistema para la definición del orden de los atributos.

5.2.5. Transformación de tipos de interrelación N-arias

En el proceso de transformación de las relaciones en las que intervienen más de dos tipos de entidad se debe aplicar la misma regla (*RTECAR-4*) que la aplicada para los tipos de interrelaciones binarias muchos a muchos. Es decir, cada tipo de entidad se transforma en una tabla y el tipo de interrelación se transforma en una nueva tabla cuyos atributos son los atributos identificadores de los tipos de entidad participantes en el tipo de interrelación y los atributos asociados al tipo de interrelación. La clave de esta tabla será la agregación de los atributos identificadores de los tipos de entidad.

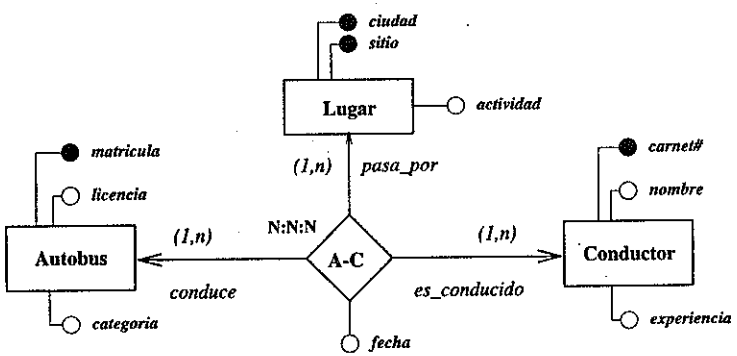


Figura 5.9 Transformación de relaciones n-arias

Si en el problema anterior sobre autobuses y conductores se desea representar los lugares que son recorridos por cada autobús conducido por cada conductor y la

fecha en que se visita ese lugar, se tiene entonces el esquema representado en la figura 5.9 en el cual, si se aplica la regla anterior, se obtiene el siguiente esquema relacional:

Autobus *(matricula, licencia, categoria)*
Conductor *(carnet#, nombre, experiencia)*
Lugar *(ciudad, sitio, actividad)*
Aut_Con_Lug **(carnet#, matricula, ciudad, sitio**, *fecha)*

Es necesario tener en cuenta que la regla *RTECAR-4* debe aplicarse cuando todos los tipos de entidad participan en el tipo de interrelación con cardinalidad máxima muchos. En caso contrario, es decir, cuando algún tipo de entidad participa en el tipo de interrelación con cardinalidad máxima uno, entonces, como es lógico, el identificador de este tipo de entidad no pasa a formar parte de la clave de la tabla que se deriva del tipo de interrelación. Obsérvese que el que un tipo de entidad participe con cardinalidad máxima uno, es igual que considerar que el tipo de interrelación está cualificado por el atributo que identifica a este tipo de entidad y, por tanto, es como un atributo más, como hay que considerar su participación de este tipo de entidad en el tipo de interrelación.

5.2.6. Transformación de tipos de interrelación reflexivas

Los tipos de interrelación reflexivas, también denominadas *recursivas*, representan aquellas relaciones que se establecen entre un mismo tipo de entidad consigo misma. Es decir, son relaciones binarias en las que únicamente interviene un tipo de entidad.

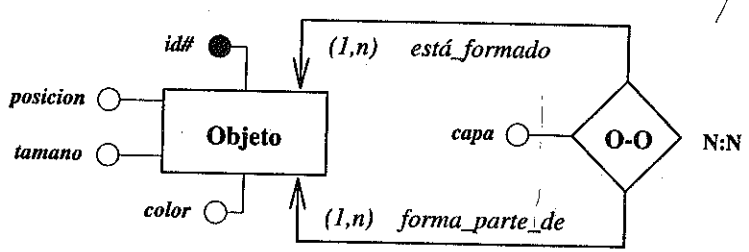


Figura 5.10 Transformación de relaciones reflexivas

Para esta clase de interrelaciones se pueden presentar dos casos:

1. El tipo de entidad participa en uno de sus papeles con cardinalidad máxima uno, y en el otro papel con cardinalidad máxima muchos, o bien ambos participan con cardinalidad máxima uno.
2. El tipo de entidad participa, en sus dos papeles, con cardinalidad máxima muchos.

En el segundo caso, este tipo de interrelación se trata igual que en el caso de las relaciones binarias muchos a muchos. Es decir, se genera una tabla para el tipo de

entidad y otra tabla para el tipo de interrelación (regla ***RTECAR-4***), siendo los atributos de esta tabla:

- Los atributos asociados al tipo de interrelación, y
- El identificador del tipo de entidad desempeñando uno de los papeles en el tipo de interrelación, y de nuevo
- El identificador del tipo de entidad desempeñando el otro papel (desempeña dos papeles) en el tipo de interrelación.

La clave de esta tabla será el identificador del tipo de entidad (por duplicado). La figura 5.10 muestra un ejemplo de este caso. El tipo de entidad *Objeto* mantiene un tipo de interrelación reflexiva con él mismo, de forma que un objeto puede estar formado por uno (al menos, él mismo) o varios objetos y, a la vez, un objeto puede formar parte de uno (al menos, él mismo) o muchos objetos. El tipo de interrelación tiene asociado el atributo *capa* que representa el nivel en el que se asocia un objeto a otro. Si aplicamos el proceso de transformación a este esquema se tendrá:

Objeto *(id#_objeto, posicion, tamano, color)*
Objeto_Objeto **(id#_continente, id#_contenido**, *capa)*

en donde: *id#_continente* representa al identificador del objeto que contiene a otros objetos, y *id#_contenido* representa al identificador del objeto contenido que forma parte de otros objetos.

Si, para este mismo ejemplo, se considera que un objeto sólo puede formar parte de otro objeto de mayor jerarquía (relación uno a muchos), como es el primer caso que se consideró, entonces se procede de alguna de las dos formas siguientes:

1. Se genera una tabla para el tipo de entidad, y en esta tabla se añade como clave foránea el identificador del tipo de entidad para representar que un objeto forma parte de un solo objeto (de forma similar a lo descrito por aplicación de la regla *RTECAR-3.1*). El esquema quedaría de la forma (cuando la cardinalidad mínima es uno):

   Objeto *(id#_contenido, posicion, tamano, color,* **id#_continente**, *capa),*

2. Se genera una tabla para el tipo de entidad y otra para el tipo de interrelación formada por los atributos asociados al tipo de interrelación, el identificador del tipo de entidad el cual será la clave de la tabla, representando al objeto que forma parte de otros objetos, y de nuevo el identificador del tipo de entidad representando al objeto del cual forman parte otros objetos de menor categoría (de forma similar al uso de la regla *RTECAR-3.2*). En este caso el esquema relacional queda de la forma (cuando la cardinalidad mínima es cero):

   Objeto *(id#_objeto, posicion, tamano, color)*
   Objeto_Objeto *(id#_contenido, capa,* **id#_continente***)*

5.3. ELIMINACIÓN DE LAS RELACIONES JERÁRQUICAS

En los esquemas conceptuales se pueden utilizar todos los principios de la abstracción para representar las relaciones existentes entre los tipos de entidad o tipos de objetos del universo de discurso. Uno de los recursos más potentes de la abstracción es la *generalización* y, por lo tanto, la *especialización* (véase el Capítulo 2), mediante la cual se pueden representar las relaciones jerárquicas que existen entre los tipos de entidad.

El problema se plantea cuando se desean transformar esquemas conceptuales en los que están presentes este tipo de relaciones entre los tipos de entidad, a esquemas lógicos relacionales. El modelo relacional no dispone de mecanismos fáciles de usar que permitan la representación de relaciones jerárquicas y, por tanto, es conveniente y necesario la eliminación de las relaciones jerárquicas como paso previo al proceso de transformación de los esquemas conceptuales a relacionales.

En el proceso de eliminación de las relaciones jerárquicas se deberá aplicar alguna de las reglas que se describirán a continuación y cuya elección va a depender de:

- *La magnitud de la especialización* que los subtipos de entidad tienen con respecto al supertipo o tipo de entidad más general. La especialización supone que los subtipos de entidad tienen asociados diferentes atributos que diferencian a un subtipo de entidad del resto de los subtipos que mantienen un tipo de interrelación jerárquica con un mismo supertipo de entidad.
- *El tipo de especialización* que representa el tipo de interrelación jerárquica, la cual puede ser alguna de las siguientes: total exclusiva, parcial exclusiva, total inclusiva y parcial inclusiva.
- *Otros tipos de interrelación* que mantengan tanto los subtipos como el supertipo de entidad.
- *Criterios de procesamiento* y, sobre todo, la forma en la que se va a acceder a la información que representan tanto el supertipo y los subtipos de entidad como los tipos de interrelación que mantienen.

Teniendo en cuenta todos los puntos señalados, en el proceso de eliminación de los tipos de interrelación jerárquicas se aplicará alguna de las siguientes reglas:

> **PRTECAR-3:** Eliminación del supertipo de entidad: *En un tipo de interrelación jerárquica se desestimará el supertipo de entidad, transfiriendo todos los atributos del supertipo a cada uno de los subtipos y cada uno de los tipos de interrelación que mantuviera el supertipo de entidad serán considerados para cada uno de los subtipos, manteniéndose, por supuesto, los tipos de interrelación en los que intervengan cada uno de los subtipos de entidad. Además, el atributo cualificador del tipo de interrelación, si estuviera presente, se puede desestimar.*
>
> *Si el tipo de interrelación jerárquica es exclusivo, los subtipos intervendrán de forma parcial (cardinalidad mínima cero) en los tipos de interrelación transferidos desde el supertipo.*

Esta regla sólo puede ser aplicada a tipos de interrelación jerárquicas totales, puesto que implica la eliminación del supertipo de entidad y, por tanto, del conjunto de entidades no especializadas en los diferentes subtipos. Además, si el tipo de interrelación jerárquica es inclusiva no es fácil representar, a priori, esta inclusividad en los nuevos tipos de interrelación que se generan con los subtipos por la transferencia de los tipos de interrelación mantenidos por el supertipo de entidad. Por lo tanto, esta regla sólo es conveniente aplicarla cuando las interrelaciones jerárquicas son totales y exclusivas.

Una transformación haciendo uso de esta regla introduce algunos inconvenientes al nuevo esquema conceptual generado:

1. Como los atributos del supertipo de entidad se trasladan a cada uno de los subtipos introduce una redundancia de información, cuando el tipo de interrelación es inclusivo.
2. Al desaparecer el supertipo de entidad desaparece la relación semántica existente entre los subtipos, la cual representaba el que todos ellos eran tipos de entidad de un mismo supertipo.
3. El número de tipos de interrelación aumenta en cuanto se transfieren los tipos de interrelación que mantenía el supertipo de entidad a cada uno de los subtipos.
4. Naturalmente, las operaciones generales de acceso a esta información (a la información transferida desde el supertipo de entidad) requieren ahora el manejo de varios tipos de entidad en lugar de sólo uno.

Esta regla sólo deberá aplicarse cuando se minimicen estos inconvenientes, es decir, cuando el número de atributos transferidos sea pequeño y no existan muchos tipos de interrelación en los que participe el supertipo de entidad.

> **PRTECAR-4:** Eliminación de los subtipos de entidad: *En un tipo de interrelación jerárquica se desestimarán los subtipos de entidad, transfiriéndose todos los atributos de los subtipos al supertipo y cada uno de los tipos de interrelación que mantuvieran los subtipos de entidad serán considerados para el supertipo, manteniéndose, por supuesto, los tipos de interrelación en los que intervenga el supertipo de entidad.*
>
> *Si el tipo de interrelación jerárquica es exclusivo, el supertipo de entidad participará de forma parcial (cardinalidad mínima cero) en aquellos tipos de interrelación transferidos desde los subtipos de entidad. En caso contrario (inclusiva) participará con las cardinalidades que participaba cada subtipo de entidad en los tipos de interrelación transferidos por aplicación de esta regla.*

El atributo cualificador del tipo de interrelación pasa a formar parte del supertipo de entidad de la forma siguiente:

1. *Si el tipo de interrelación es exclusivo no formará parte de la clave.*
2. *Si el tipo de interrelación es inclusivo formará parte de la clave, originando redundancia de los atributos del supertipo para cada instancia de los subtipos.*
3. *Si el tipo de interrelación es parcial, podrá tomar valores nulos para representar a entidades que no se especializan.*

El uso de esta regla va a dar lugar a un esquema mucho más simple, pero en el caso de tipos de interrelaciones exclusivas y/o parciales se van a presentar muchos posibles valores nulos para aquellos atributos transferidos desde los subtipos al supertipo de entidad. Así, si el número de instancias o entidades del supertipo va a ser elevado (todas las instancias o entidades de cada uno de los subtipos) el número de valores nulos será muy elevado. Además, en los procesos de acceso a la información se va a transferir, en muchos casos, información no requerida por el interrogante, debido a que en el supertipo de entidad se está manteniendo información particular de cada subtipo y en muchos interrogantes sólo será necesaria información general o bien referente a alguno de los subtipos.

En principio, esta regla puede aplicarse a cualquiera de los cuatro tipos de interrelaciones jerárquicas, aunque no en todos los casos, como se ha comentado, las ventajas de la simplicidad compensan los inconvenientes que presenta su uso, no siendo muy adecuada en el caso de tipos de interrelaciones parciales inclusivas.

PRTECAR-5: Eliminación de la jerarquía: *El tipo de interrelación jerárquica se transformará en tantos tipos de interrelación uno a uno como subtipos de entidad estén presentes, manteniéndose los tipos de interrelación en los que intervienen tanto los subtipos, como el supertipo de entidad. En los tipos de interrelación generados por la transformación, los subtipos de entidad participarán:*

1. *Si el tipo de interrelación jerárquica es exclusivo, participarán con cardinalidad mínima cero.*
2. *Si el tipo de interrelación jerárquica es inclusivo, participarán con cardinalidad mínima cero o uno.*

En estos tipos de interrelación el supertipo participa con cardinalidades mínima y máxima igual a uno, pudiendo considerarse que los subtipos de entidad actúan como tipos de entidad débiles por identificación con respecto al supertipo.

La regla *PRTECAR-5* es la de aplicación más general para la transformación de las relaciones jerárquicas, pues tiene la ventaja de que el esquema resultante preserva la representación de las relaciones existentes entre el supertipo y los subtipos de

entidad a través de los nuevos tipos de interrelación débiles creados, pudiendo ser aplicada para cualquier tipo de interrelación jerárquica de los cuatro tipos posibles. La regla presenta algún problema en el caso de tipos de interrelaciones parciales e inclusivas, debido al dominio que debe asignarse al cualificador del tipo de interrelación para garantizar la representación correcta de la especialización.

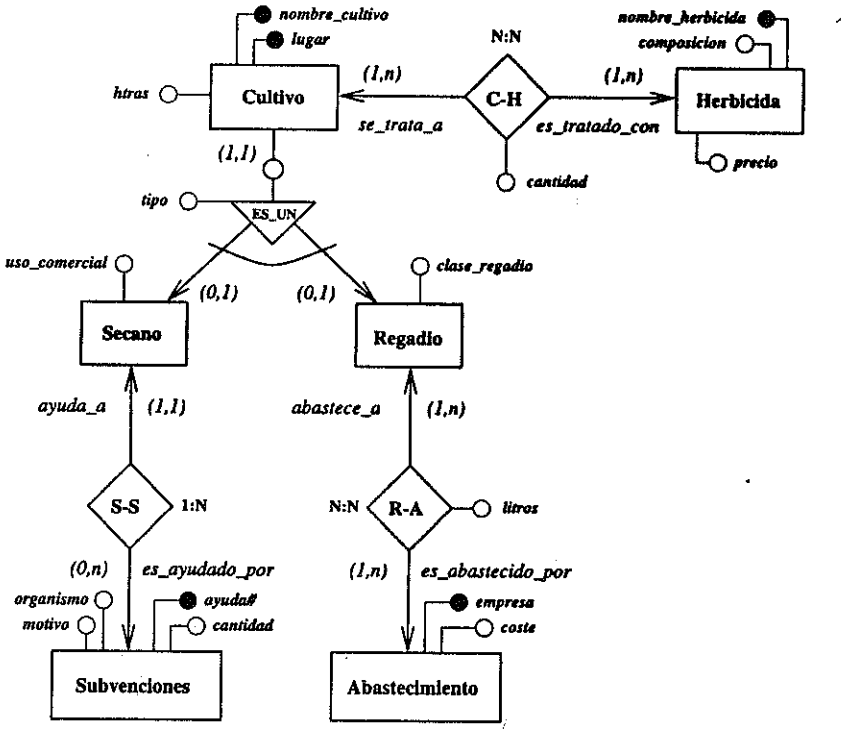


Figura 5.11 Ejemplo de relaciones jerárquicas

Su principal inconveniente es que el nuevo esquema conceptual generado es bastante más complejo que el original e introduce redundancia lógica (no por ello física) en la información representada.

En la figura 5.11 se ha representado un esquema conceptual a partir del cual se va a presentar un ejemplo de aplicación de cada una de estas reglas. El ejemplo representa a una serie de cultivos agrícolas sobre los cuales se desea mantener la información correspondiente al tipo de cultivo y, dependiendo de éste, las ayudas que reciben de los organismos oficiales o el abastecimiento de agua que necesita. Además,

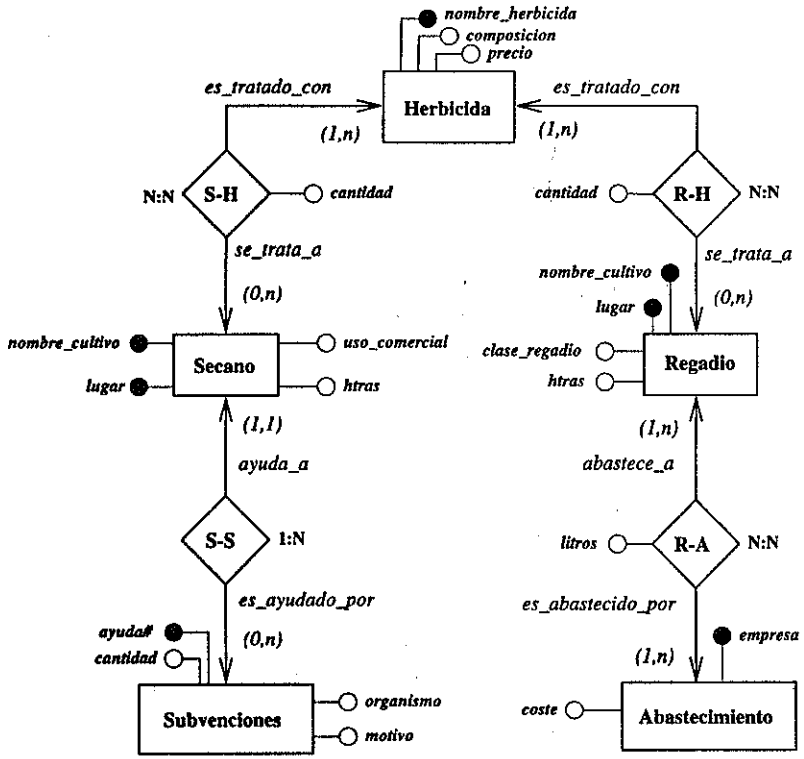


Figura 5.12 Relaciones jerárquicas. ***Regla:*** PRTECAR-3

se representa el consumo de herbicidas de cada uno de los cultivos. La eliminación del tipo de interrelación jerárquica que se muestra en la figura 5.11 dará lugar a:

Aplicación de la regla *PRTECAR-3*: desaparece el supertipo de entidad (*Cultivo*), pasando sus atributos a los subtipos de entidad (*Secano* y *Regadio*), y el tipo de interrelación *(C-H)* se transfiere a cada uno de los subtipos. Como se muestra en la figura 5.12, la cardinalidad máxima con la que participaba el tipo de entidad *Cultivo* en el tipo de interrelación *(C-H)*, es transferida, como cardinalidad máxima, a las relaciones generadas entre el tipo de entidad *Herbicida* con *Secano* y *Regadio*. De esta forma se conserva la relación existente entre los herbicidas y cada uno de los tipos de cultivos, independientemente de sus características. Además, el atributo que cualifica el tipo de interrelación *(C-H)* entre los tipos de entidad *Cultivo* y *Herbicida* es transferido a cada uno de los nuevos tipos de interrelación generados entre el tipo de entidad *Herbicida* y los subtipos de entidad.

El atributo que caracterizaba el tipo de interrelación jerárquica es eliminado, puesto que se ha eliminado la semántica y, por tanto, la necesidad de la especialización.

Los tipos de interrelación que mantenían los subtipos de entidad se siguen manteniendo sin experimentar ninguna modificación. Basándose en esta transformación el esquema relacional resultante que se puede derivar aplicando las reglas de transformación sería el siguiente (para cada tabla se hace indicación de las reglas que se han aplicado para su derivación):

REGLAS	RELACIONES	
1	**Herbicida**	(*nombre_herbicida, composicion, precio*)
1	**Secano**	(*nombre_cultivo, lugar, htras, uso_comercial*)
1	**Regadio**	(*nombre_cultivo, lugar, htras, clase_regadio*)
1, 3.1	**Subvenciones**	(*ayuda#, organismo, motivo, cantidad,* **nombre_cultivo, lugar**)
1	**Abastecimiento**	(*empresa, coste*)
4	**Herb_Rega**	(**nombre_herbicida, nombre_cultivo, lugar**, *cantidad*)
4	**Herb_Seca**	(**nombre_herbicida, nombre_cultivo, lugar**, *cantidad*)
4	**Rega_Abas**	(**nombre_cultivo, lugar, empresa**, *litros*)

Si se analiza este esquema relacional detenidamente se puede observar lo siguiente:

- Las tablas *Secano* y *Regadio* tienen la misma clave y, por tanto, no existe ningún condicionante lógico que impida que un mismo cultivo pueda aparecer como tuplas de ambas tablas, algo que va en contra de las restricciones del problema en las que se indicaba que un cultivo es o bien de secano o bien de regadío, pero no de los dos tipos al mismo tiempo. Así, este control deberá ser llevado externamente por los usuarios y los correspondientes programas de aplicación que manejen estas tablas.
- Las tablas *Herb_Rega* y *Herb_Seca* parecen idénticas, aunque no lo son por lo que no es posible agruparlas en una sola con objeto de intentar simplificar el esquema, puesto que se presentarían los siguientes problemas:

Herb_Cult (*nombre_herbicida, nombre_cultivo, lugar, cantidad*)

  - Sería necesario incluir además el cualificador del tipo de interrelación jerárquica para diferenciar a los cultivos de secano y los de regadío.
  - Presenta el problema del control de la integridad puesto que sería necesario definir la clave de esta tabla como clave foránea de las tablas *Secano* y *Regadio*, y esto sería incorrecto.
  - Para poder considerar la tabla *Herb_Cult* sería también necesario unificar las tablas *Secano* y *Regadio* en una única tabla que agrupara todos los atributos de ambas tablas y esto sería también incorrecto por

los diferentes tipos de interrelación en que participan los tipos de entidad que han dado lugar a estas tablas.

Luego la agrupación considerada no sería correcta, lo que se pone mucho más claramente de manifiesto si el tipo de interrelación jerárquica hubiera sido inclusivo.

- En ningún caso se puede aplicar para tipos de interrelación jerárquicas parciales, pues al eliminarse el supertipo se eliminaría la representación de las entidades que no pertenecen a ningún subtipo.

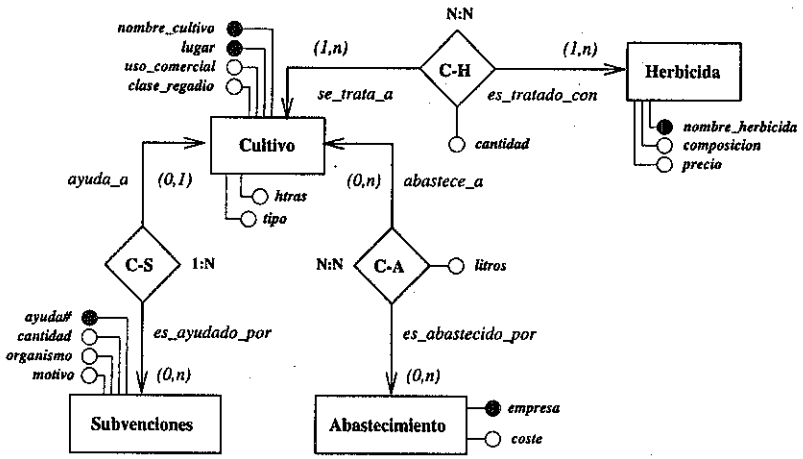


Figura 5.13 Relaciones jerárquicas. ***Regla:*** PRTECAR-4

Aplicación de la regla *PRTECAR-4*: desaparecen los subtipos de entidad *Secano* y *Regadío*, transfiriéndose los atributos de los subtipos al supertipo de entidad, y los tipos de interrelación que mantenía cada subtipo son transferidos al tipo de entidad.

En este caso, el atributo *tipo* no puede ser eliminado, pues sigue manteniendo la función de representar la clase de cultivo. Ahora bien, si el tipo de interrelación jerárquica fuera inclusivo (en lugar de exclusiva) este atributo debe formar parte del identificador del tipo de entidad *Cultivo* para permitir representar que un mismo cultivo puede ser a la vez de secano y regadío (por lo que se vería afectado también el esquema relacional).

La figura 5.13 muestra el esquema conceptual resultante después de la aplicación de esta regla al esquema conceptual representado en la figura 5.11. Si se aplican, ahora, las reglas de transformación *RTECAR* se obtendrá el siguiente esquema relacional:

REGLAS	RELACIONES	
1	**Herbicida**	*(nombre_herbicida, composicion, precio)*
1	**Cultivo**	*(nombre_cultivo, lugar, tipo, htras, uso_comercial, clase_regadio)*
1, 3.1	**Subvenciones**	*(ayuda#, organismo, motivo, cantidad,* **nombre_cultivo, lugar***)*
1	**Abastecimiento**	*(empresa, coste)*
4	**Herb_Cult**	*(***nombre_herbicida, nombre_cultivo, lugar***, cantidad)*
4	**Abas_Cult**	*(***nombre_cultivo, lugar, empresa***, litros)*

Cuyo análisis da lugar a los siguientes comentarios:

- En la tabla *Subvenciones* existe una clave foránea (la pareja *nombre_cultivo, lugar*) que representa que una subvención es asignada a un único cultivo, pero es difícil restringir que ese cultivo debe ser un cultivo de secano (sólo mediante programación a través del valor del atributo *tipo* en la tabla *Cultivo* y no en la representación del esquema).
- Para poder representar un tipo de interrelación jerárquica inclusiva, debe considerarse que el atributo *tipo* forma parte de la clave de la tabla *Cultivo*, como se ha comentado anteriormente, y además, se deberán modificar las tablas en las que se haga referencia a la clave de la tabla *Cultivo*. En este caso el esquema relacional quedaría de la forma:

REGLAS	RELACIONES	
1	**Herbicida**	*(nombre_herbicida, composicion, precio)*
1	**Cultivo**	*(nombre_cultivo, lugar, tipo, htras, uso_comercial, clase_regadio)*
1, 3.1	**Subvenciones**	*(ayuda#, organismo, motivo, cantidad,* **nombre_cultivo, lugar, S***)*
1	**Abastecimiento**	*(empresa, coste)*
4	**Abas_Cult**	*(***nombre_cultivo, lugar, R, empresa***, litros)*
4	**Herb_Cult**	*(***nombre_herbicida, nombre_cultivo, lugar, tipo***, cantidad)*

En donde, como se observa, se obliga a que en las tablas *Subvenciones* y *Abas-Cult* el atributo *tipo* tome el valor correspondiente a la especialización, y están presentes los problemas señalados anteriormente.

Aplicación de la regla *PRTECAR-5*: desaparece el tipo de interrelación jerárquica, transformándose en tipos de interrelación débiles por identificación entre el supertipo y los subtipos de entidad y manteniendo su estado los demás objetos del esquema conceptual.

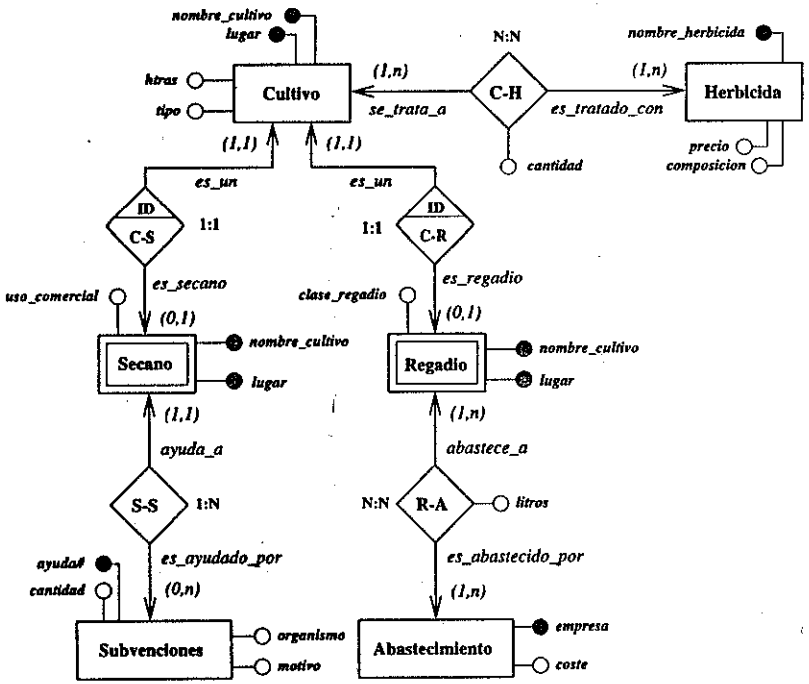


Figura 5.14 Relaciones jerárquicas. ***Regla:*** PRTECAR-5

Al igual que en el caso anterior, el atributo *tipo* pasa a formar parte del tipo de entidad ***Cultivo***. Mediante su uso (incorporándolo al identificador, o no), se podrá seguir representando la **exclusividad** o inclusividad del tipo de interrelación jerárquica eliminado por el uso de esta regla. La figura 5.14 muestra el esquema conceptual resultante a partir del cual se deriva el siguiente esquema relacional:

REGLAS	RELACIONES	
1	**Herbicida**	*(nombre_herbicida, composicion, precio)*
1	**Cultivo**	*(nombre_cultivo, lugar, htras, tipo)*
1, 2.2	**Secano**	*(**nombre_cultivo, lugar**, uso_comercial)*
1, 2.2	**Regadio**	*(**nombre_cultivo, lugar**, clase_regadio)*
1, 3.1	**Subvenciones**	*(ayuda#, organismo, motivo, cantidad,* **nombre_cultivo, lugar***)*
1	**Abastecimiento**	*(empresa, coste)*
4	**Herb_Cult**	*(**nombre_herbicida, nombre_cultivo, lugar**, cantidad)*
4	**Abas_Rega**	*(**nombre_cultivo, lugar, empresa**, litros)*

De cuyo análisis se deduce que:

- Se sigue representando la semántica de la relación jerárquica, y dependiendo de si el atributo *tipo* es clave o no de la tabla *Cultivo* se pueden representar tipos de interrelación jerárquicas exclusivas e inclusivas.
- La tabla *Subvenciones* mantendrá una referencia con la clave de la tabla *Secano*, por lo que ahora sí es fácil controlar que una subvención sólo se adjudica a un cultivo de secano. Igual razonamiento se puede hacer con la tabla *Abas_Rega*.

CAPÍTULO 6

EL CATASTRO MUNICIPAL

6.1. ENUNCIADO DEL PROBLEMA

Se desea considerar la información correspondiente al catastro de viviendas de un determinado municipio. En el municipio existe una serie de zonas urbanas en las cuales se ha edificado un conjunto de viviendas, las cuales pueden ser:

- **Viviendas unifamiliares** o casas en las que sólo habita una familia y,
- **Bloques de pisos** en los cuales existe un conjunto de viviendas, indeterminado a priori, en cada una de las cuales habita una familia.

En el sistema es necesario mantener la información correspondiente a las personas que viven en cada una de las viviendas, así como el cabeza de familia de las personas que habitan o son propietarias de las viviendas. Para cada vivienda, además de la información correspondiente a las características de las mismas, es necesario conocer al propietario.

Se van a considerar, los siguientes supuestos semánticos en el problema:

SUPUESTO 1: Toda persona habita en una y sólo una vivienda, la cual es considerada como su vivienda o residencia principal.

SUPUESTO 2: Cada vivienda tiene uno y sólo un propietario.

SUPUESTO 3: Las viviendas se encuentran en una única zona urbana correspondiente al municipio, de las cuales interesa mantener información.

SUPUESTO 4: Las zonas urbanas en las que está dividido geográficamente el municipio tienen nombres diferentes.

SUPUESTO 5: En cada zona urbana del municipio existen una serie de calles en las que se construyen las viviendas. Los nombres de las calles son únicos para el municipio con independencia de la zona urbana en la que se encuentren (para simplificar el problema no se considerará información sobre las calles).

SUPUESTO 6: En el contexto del problema, una familia es un conjunto de personas que tienen una relación familiar directa y que habita, o no, en una misma vivienda. Este conjunto podrá ser unario.

SUPUESTO 7: Como se indica en el enunciado del problema las viviendas pueden ser casas unifamiliares o bloques de pisos en los cuales existen una serie de viviendas unifamiliares.

6.2. MODELO CONCEPTUAL

Se trata de representar la información de las viviendas construidas en un municipio incluyendo tanto la de los propietarios de las mismas como la de las personas que habitan en ellas. También se solicita representar la relación familiar de las personas consideradas en el problema.

6.2.1. Análisis de los tipos de entidad

En principio, del análisis del enunciado del problema pueden ser extraídos los siguientes tipos de entidad, como se muestra en la figura 6.1:

Tipo de entidad *ZonaUrbana*: el cual representa al objeto del mundo real "*barrio*" o "*distrito*", y que se puede definir como "*la zona geográfica que forma parte del municipio y donde se encuentran las viviendas*". Se van a considerar dos atributos para este tipo de entidad: *nombre_zona* y *od_zona*.

El atributo *nombre_zona* identifica a la zona geográfica, pues como indica el enunciado no puede repetirse, y el atributo *od_zona* hace referencia a cualquier otro dato que se considere que sea de interés representar (SUPUESTOS 3-4).

Tipo de entidad *Vivienda*: el cual representa al objeto del mundo real "*vivienda*", y que representa al "*solar*" o "*zona en la que se ha construido un edificio con el objetivo de ser utilizado como vivienda*". Como se puede observar en el enunciado, existen dos tipos de viviendas: aquellas que son unifamiliares y aquellas que son bloques de pisos.

Para el tipo de entidad *Vivienda* se van a considerar los atributos *calle* y *numero*, cuya agregación identifica a las entidades de este tipo. Además, consideraremos

otros atributos como *codigo_postal, metros* y *od_vivienda* los cuales representan el código postal asignado a la dirección de la vivienda, los metros cuadrados del solar y/o edificio y cualquier otro conjunto de datos que se desee representar con respecto a la vivienda, respectivamente.

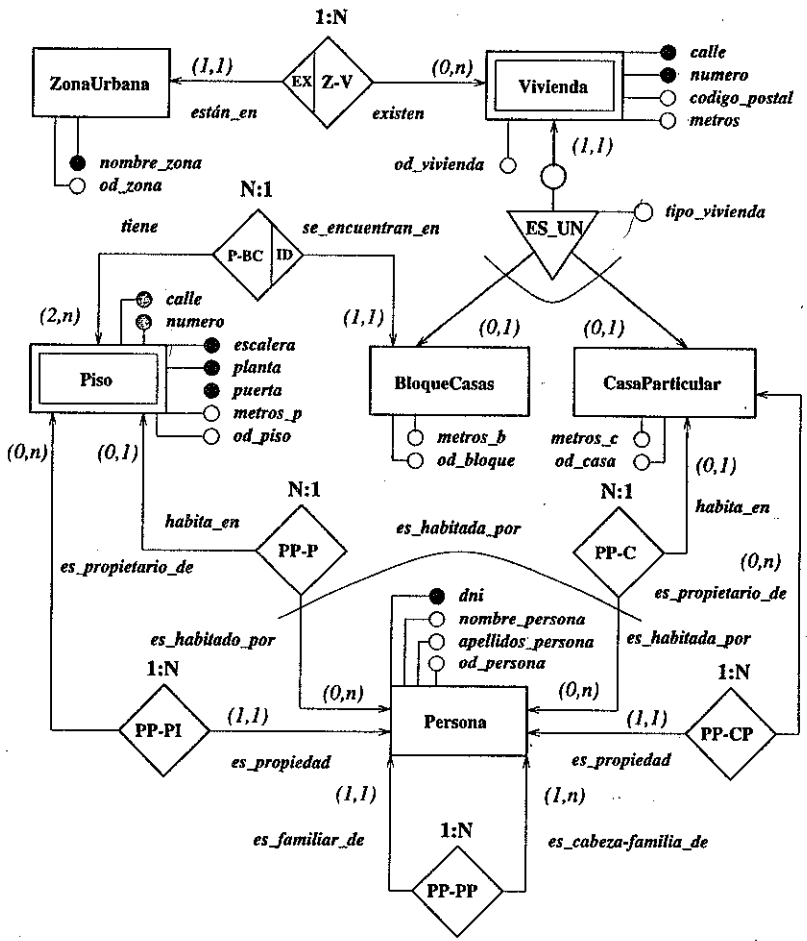


Figura 6.1 Esquema EE-R del modelo conceptual del problema del catastro

El tipo de entidad *Vivienda* podría considerarse como un tipo de entidad débil por existencia con respecto al tipo de entidad *ZonaUrbana*, pues es lógico considerar que una vivienda no existe si no existe una zona urbana en la que se pueda

construir (SUPUESTO 3). Sin embargo, y si consideramos que no existen calles con el mismo nombre ni números iguales en una misma calle, la debilidad no será por identificación, puesto que es posible identificar una vivienda con independencia de la identificación de la zona urbana en la que se encuentre (SUPUESTO 5).

Por otra parte, el tipo de entidad *Vivienda* es posible especializarlo en dos subtipos de entidad: el tipo de entidad *CasaParticular*, el cual hace referencia a aquellas viviendas en las que sólo vive una familia y que puede ser identificada por la agregación de los atributos *calle* y *numero*; y el tipo de entidad *BloqueCasas*, para los cuales es conveniente considerar otros atributos que representen información sobre los metros construidos (*metros_c* y *metros_b*) y cualquier otra información de interés (*od_casa* y *od_bloque*)

Tipo de entidad *Piso*: el cual representa a "*cada una de las viviendas que pueden existir en un bloque de pisos*" y que necesitan para su identificación los atributos *escalera, planta* y *puerta*. Se trata de un tipo de entidad débil por identificación con respecto al tipo de entidad *BloqueCasas* puesto que no existe un piso si no existe un bloque, y además es necesario conocer la calle y el número en el cual se encuentra el bloque para identificar sin ambigüedad un piso del municipio (SUPUESTO 5).

Tipo de entidad *Persona*: el cual representa al objeto del mundo real "*personas que viven o son propietarias de una determinada vivienda*". Para este tipo de entidad se van a considerar cuatro atributos: *dni, nombre_persona, apellidos_persona* y *od_persona* (SUPUESTO 1).

Consideraremos que el atributo *dni* puede servir para identificar sin ambigüedad las entidades de este tipo, por lo que lo consideraremos identificador principal[17]. El atributo *od_persona* representa, en este caso, a cualquier conjunto de atributos que se deseen considerar también.

6.2.2. Análisis de los tipos de interrelación

Los tipos de entidad antes mencionados se encuentran relacionados de la siguiente forma:

Tipo de interrelación *ZonaUrbana/Vivienda* **(Z-V)**: el cual relaciona los tipos de entidad *ZonaUrbana* y *Vivienda*, la interrelación es del tipo *1:N*, ya que una vivienda se encuentra en una única zona urbana, por tanto, el tipo de entidad *ZonaUrbana* participa con las cardinalidades *(1,1)* en este tipo de interrelación, mientras que en una zona urbana pueden existir muchas viviendas. Se puede considerar que una zona urbana tiene existencia independientemente de que

[17] Es evidente que entre las personas consideradas en este problema existirá un gran número de ellas que carezca de documento nacional de identidad (los menores de edad), por lo que el lector puede pensar que no es adecuado tomar este atributo como identificador principal. La elección se ha realizado simplemente para no añadir una mayor complejidad a este primer ejercicio.

existan o no viviendas construidas en la misma, por lo que el tipo de entidad *Vivienda* participa en el tipo de interrelación *Z-V* con las cardinalidades *(0,n)*, como se puede observar en la figura 6.1 (SUPUESTO 3).

Tipo de interrelación *Vivienda/Persona*: el cual representa la relación existente entre las viviendas y las personas que habitan en las mismas. Se puede ver este tipo de interrelación como una relación exclusiva entre el tipo de entidad *Persona* y los tipos de entidad *CasaParticular* y *Piso*, puesto que una persona vivirá en una, y sólo en una, vivienda de un tipo al mismo tiempo (la residencia principal) (SUPUESTO 1).

Así, los tipos de entidad *CasaParticular* y *Piso* participan en los tipos de interrelación *(PP-P)* y *(PP-C)* con las cardinalidades *(0,1)*, mientras que el tipo de entidad *Persona* participa de forma exclusiva con las cardinalidades *(0,n)* (SUPUESTO 1), considerándose que las viviendas pueden estar deshabitadas o bien habitadas por varias personas (SUPUESTOS 1 y 6).

Tipo de interrelación *Propietario/Vivienda*: el cual representa a los propietarios de cada una de las viviendas, tanto pisos como casas particulares. Se trata, como se aprecia en la figura 6.1, de dos tipos de interrelación en los que participa el tipo de entidad *Persona*:, uno con el tipo de entidad *Piso (PP-PI)*, y con el tipo de entidad *CasaParticular (PP-CP)*, puesto que una persona puede ser propietaria de cualquier tipo de vivienda. Como se ha considerado que una vivienda sólo puede ser propiedad de una y sólo una persona, el tipo de entidad *Persona* participa en estos tipos de interrelación con las cardinalidades *(1,1)* (SUPUESTO 2). Por otro lado, una persona puede ser propietaria de ninguna, una o varias viviendas, por lo que los tipos de entidad *CasaParticular* y *Piso* participan con la cardinalidad *(0,n)*.

Tipo de interrelación *Piso/BloqueCasas* **(P-BC)**: el cual representa a los pisos que existen en cada uno de los bloques de casas. Se considera que un bloque de casas está formado por dos o más pisos, por lo que el tipo de entidad *Piso* interviene en este tipo de interrelación con las cardinalidades *(2,n)*, mientras que un piso sólo se encuentra ubicado en un bloque de casas, interviniendo el tipo de entidad *BloqueCasas* con las cardinalidades *(1,1)* (SUPUESTO 7).

Tipo de interrelación *Persona/Persona* **(PP-PP)**: el cual representa la relación existente entre cada persona con su cabeza de familia. Se trata de un tipo de interrelación reflexiva en la que, por un lado, una persona puede ser cabeza de familia de una o varias personas (cada persona es al menos cabeza de familia de una persona, ella misma) —cardinalidad *(1,n)*— mientras, por otro lado, una persona sólo tiene como cabeza de familia a una y sólo una persona —cardinalidad *(1,1)*—. Por tanto, y como se muestra en la figura 6.1, se trata de una relación reflexiva *1:N* entre el tipo de entidad *Persona* con él mismo.

Este tipo de interrelación tiene su razón de ser en base al enunciado del problema en el cual se indica que es de interés el conocimiento de la relación familiar de las personas que habitan en la misma o distinta vivienda (SUPUESTO 6).

6.3. MODELO RELACIONAL

Una vez realizado el modelo conceptual se va a construir el modelo relacional del problema considerado.

6.3.1. Tabla *ZonaUrbana*

La tabla *ZonaUrbana* se forma a partir del tipo de entidad *ZonaUrbana*. Esta tabla incluye los atributos *nombre_zona* y *od_zona* que son tomados del tipo de entidad antes mencionado (regla *RTECAR-1*).

La clave principal de la tabla es el atributo *nombre_zona*, no considerándose, en este caso, ningún otro atributo que pueda ser utilizado como identificador alternativo, y ninguno de los atributos de esta tabla mantiene referencia alguna con los atributos de las otras tablas. La tabla *ZonaUrbana* queda de la forma siguiente:

ZonaUrbana *(nombre_zona, od_zona)*

6.3.2. Tabla *Vivienda*

La tabla *Vivienda* se forma a partir del tipo de entidad del mismo nombre tomando los atributos de este tipo de entidad (regla *RTECAR-1*), además de:

- El atributo identificador de la tabla *ZonaUrbana* con la cual mantiene una relación uno a muchos (regla *RTECAR-3.1*).
- El atributo que caracteriza el tipo de vivienda, y que cualifica la especialización de este tipo de entidad en dos subtipos. Determinándose, de esta forma, si se trata de una casa particular o un bloque de pisos (regla *PRTECAR-5*).

El identificador de esta tabla es la agregación de los atributos *calle* y *numero*, que representan la localización del bloque de pisos o de la casa particular, no existiendo ningún otro identificador alternativo. La estructura de esta tabla queda de la forma:

Vivienda *(calle, numero, tipo_vivienda, metros, codigo_postal,* **nombre_zona**, *od_vivienda)*

6.3.3. Tabla *CasaParticular*

La tabla *CasaParticular* se forma a partir del tipo de entidad *CasaParticular* (regla *RTECAR-1*). Esta tabla hereda del tipo de entidad *Vivienda* todos los atributos identificadores de ésta, por lo que mantendrá una referencia a la misma (regla *PRTECAR-5*). Además, incorpora, por el tipo de interrelación *(PP-CP)*, el atributo *dni_cp*, el cual mantiene una referencia con el identificador de la tabla *Persona*, lo que permite representar al propietario de la casa particular (reglas *RTECAR-2.2* y *3.1*). La estructura de esta tabla queda, por tanto, de la forma:

CasaParticular *(calle, numero, metros_c, od_casa,* **dni_cp***)*

6.3.4. Tabla *BloqueCasas*

Esta tabla, al igual que la anterior, surge en base al tipo de entidad del mismo nombre (regla *RTECAR-1*), tratándose de una especialización del tipo de entidad *Vivienda*. Incorpora los atributos *calle* y *numero* a través de los cuales mantiene una referencia con la tabla *Vivienda* y, además, otros atributos como *metros_b* y *od_bloque*, como se ha indicado en la descripción de este tipo de entidad (reglas *PRTECAR-5* y *2.2*). Su estructura, por tanto, queda de la forma:

BloqueCasas *(calle, numero, metros_b, od_bloque)*

Puede observarse que no incorpora el atributo *dni* del tipo de entidad *Persona*, puesto que las personas son propietarias de pisos o casas pero no de bloques completos (sólo en el caso de que sean propietarios de cada uno de los pisos del bloque), como así se ha expuesto en la descripción del problema.

6.3.5. Tabla *Piso*

La tabla *Piso* se forma a partir del tipo de entidad *Piso* (regla *RTECAR-1*). Esta tabla mantiene una referencia con la tabla *BloqueCasas* a través de los atributos *calle* y *numero* (regla *RTECAR-3.1*), además incorpora los atributos *escalera, planta, puerta*, como parte de la identificación de las tuplas, y el resto de los atributos del tipo de entidad *Piso*. La clave principal de la tabla *Piso* es la agregación de los atributos *calle, numero, escalera, planta* y *puerta*. Además, incorpora el atributo *dni_p*, el cual mantiene una referencia con el identificador de la tabla *Persona*, lo que permite representar al propietario del piso (regla *RTECAR-3.1*). La tabla *Piso* queda, por tanto, de la forma:

Piso *(calle, numero, escalera, planta, puerta, metros_p, od_piso,* **dni_p***)*

6.3.6. Tabla *Persona*

La tabla *Persona* se forma a partir del tipo de entidad *Persona*, y de ésta toma los atributos *dni, nombre_persona, apellidos_persona* y *od_persona*. También incorpora los atributos identificadores de los tipos de entidad *CasaParticular* y *Piso* debido a los tipos de interrelación *(PP-C)* y *(PP-P)* que mantiene con ellos respectivamente, y que, por tanto, permite representar a la única vivienda considerada en la que habita cada persona (ver figura 6.1) (regla *RTECAR-3.1*).

También en la tabla *Persona* existe otro atributo, *dni_c*, que es tomado del tipo de entidad *Persona* por la interrelación reflexiva *(PP-PP)* que se utiliza para representar la interrelación de una determinada persona con el cabeza de familia.

La clave de la tabla *Persona* es *dni*, la cual identifica a cada una de las ocurrencias de la tabla, puesto que se ha considerado que no hay dos personas con el mismo número de documento nacional de identidad.

La tabla *Persona* mantiene referencias con las tablas *CasaParticular y Piso* a través de los atributos *calle, numero, escalera, planta* y *puerta* respectivamente. La tabla *Persona* tiene la siguiente estructura:

Persona *(dni, nombre_persona, apellidos_persona, od_persona,* **dni_c, calle, numero, escalera, planta, puerta***)*

Puede observarse que para aquellas personas que habiten en casas unifamiliares los atributos *escalera, planta* y *puerta* tomarán valores *nulos*. En este caso no existirá ninguna tupla de la tabla *Piso* a la que haga referencia la tupla de la tabla *Persona*, puesto que la única referencia será con alguna tupla de la tabla *CasaParticular* a través de los atributos *calle, numero*. Pero en el caso en que una persona habitara en un piso, al ser los atributos *calle* y *numero* no nulos, se debería satisfacer la referencia a *CasaParticular*, la cual nunca se cumpliría.

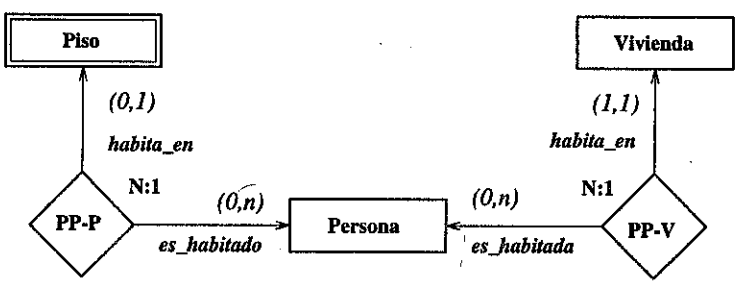


Figura 6.2 La vivienda habitual de las personas

Obsérvese, que en la definición de esta tabla será necesario, para garantizar la integridad de referencia, realizar dos restricciones de referencia: una a la tabla *CasaParticular* a través de los atributos *calle, numero*, y otra a la tabla *Piso* a través de los atributos *calle, numero, escalera, planta, puerta*, pero estas dos referencias son incompatibles desde el punto de vista del problema, puesto que una persona habita en un piso o bien habita en una casa particular.

La causa de este problema está en los tipos de interrelación exclusiva que mantiene el tipo de entidad *Persona* con los tipos de entidad *Piso* y *CasaParticular*, exclusividad que es necesario eliminar en la derivación del esquema conceptual al relacional.

La manera obvia de eliminar esta exclusividad es como se muestra en la figura 6.2; es decir, considerando que el tipo de entidad *Persona* participa con el tipo de entidad *Vivienda* en un tipo de interrelación total *N:1*, puesto que una persona, para ser considerada en el sistema, debe habitar en una y sólo una vivienda. Y por otro lado, considerar un tipo de interrelación parcial entre los tipos de entidad *Persona* y *Piso*, representando que una persona habitará, o no, un piso, puesto que puede habitar en una casa particular.

Si se derivan estos dos tipos de interrelación se tiene:

- Que por aplicación de la regla *RTECAR-3.1*, la tabla *Persona* queda de la forma:

Persona *(dni, nombre_persona, apellidos_persona, od_persona,* **calle, numero, dni_c***)*

representando la vivienda en la que habita una persona.

- Y que por aplicación de la regla *RTECAR-3.2*, se obtiene la tabla:

HabitaPiso *(dni,* **calle, numero, escalera, planta, puerta***)*

representando, en su caso, el piso habitado por cada persona.

6.4. NORMALIZACIÓN DEL MODELO

Tabla *ZonaUrbana:* esta tabla se encuentra en *FNBC*, pues todos los atributos son atómicos (*FN1*) y las dependencias funcionales son completas entre los atributos no primos con el único determinante funcional.

nombre_zona → *od_zona*

Tabla *Vivienda:* esta tabla se encuentra en *FNBC* puesto que el único determinante funcional existente es el identificador principal y todas las dependencias con el resto de atributos es completa.

calle, numero → *codigo_postal, metros, tipo_vivienda, od_vivienda, nombre_zona*

Se puede observar que para considerar a esta tabla normalizada en *FNBC* se debe tener en cuenta lo siguiente:

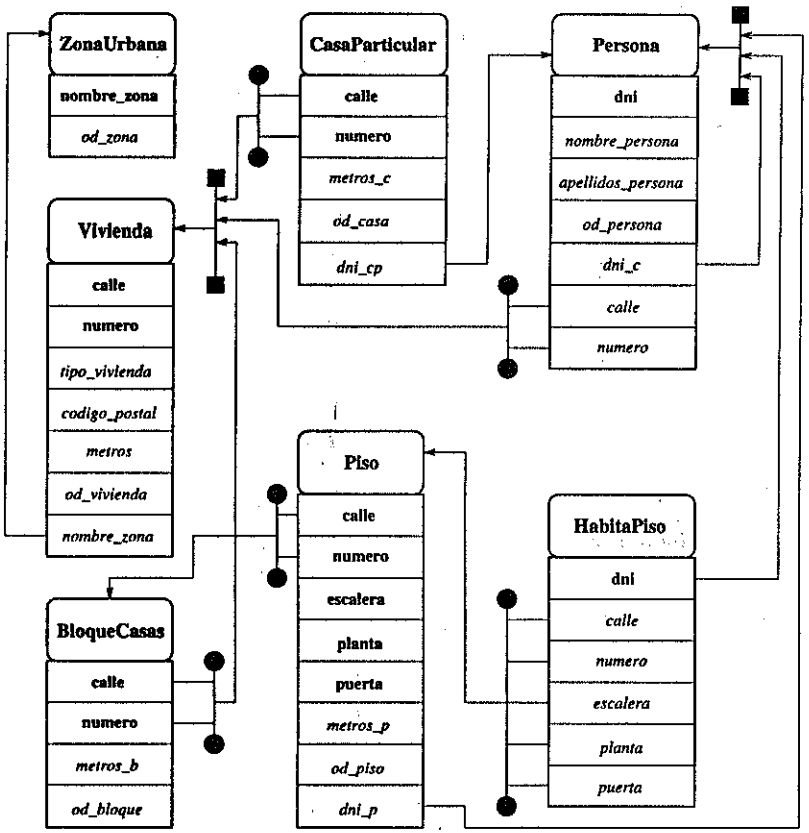


Figura 6.3 Diagrama relacional para el problema catastral

- Que una calle no pertenece a un único código postal. Es decir, que puede ocurrir que parte de las viviendas de una calle tengan asignado un código postal (hasta un cierto tramo de la calle) y el resto otro código postal distinto.
- Que a una calle no le corresponde una única zona urbana. Es decir, que se puede dar el caso de que un tramo de la calle se encuentre en una zona urbana y otro tramo en otra zona diferente.

En definitiva, que los atributos *codigo_postal* y *nombre_zona* dependen de forma completa de la clave de esta tabla. Es decir, la agregación de los atributos *calle* y *numero* determinan los valores de estos otros.

Si no se consideran los supuestos anteriores, entonces la tabla *Vivienda* no se encontraría en *FN2*, pues la dependencia funcional no sería completa, siendo necesario descomponerla de la forma siguiente:

Vivienda-1 *(calle, numero, tipo_vivienda, metros)*

Vivienda-2 *(calle, codigo_postal, nombre_zona)*

Subesquema en el cual se ha considerado en la tabla *Vivienda-2* que los atributos *codigo_postal* y *nombre_zona* son independientes. Es decir, que un código postal puede corresponder a más de una zona urbana.

Si, por el contrario, se considera que los códigos postales están asignados a zonas urbanas; es decir, que a un valor del atributo *codigo_postal* le corresponde uno y sólo un valor del atributo *nombre_zona*, entonces la tabla *Vivienda-2* no se encontraría en *FN3*, puesto que existe una dependencia transitiva entre los atributos de la misma. Por ello, sería necesario descomponerla, con lo que el subesquema final quedaría de la forma:

Vivienda-1 *(calle, numero, tipo_vivienda, metros)*

Vivienda-2 *(calle,* **codigo_postal***)*

Vivienda-3 *(codigo_postal, nombre_zona)*

Subesquema en el que las tres tablas se encuentran en *FNBC*.

Como en los supuestos semánticos de este problema no se ha hecho ninguna aclaración específica acerca de las características de las relaciones entre estos atributos, y con el objetivo de no complicar excesivamente este primer problema práctico con el que se encuentra el lector, se va a considerar que estos atributos son independientes y que la tabla *Vivienda* se encuentra normalizada en *FNBC*.

Tabla *BloqueCasas*: al igual que en el caso anterior, esta tabla se encuentra en *FNBC*.

calle, numero → *metros_b, od_bloque*

Tabla *CasaParticular*: al igual que en el caso anterior, esta tabla se encuentra en *FNBC*.

calle, numero → *metros_c, od_casa, dni_cp*

Tabla *Piso*: por el mismo razonamiento esta tabla también se encuentra en *FNBC*.

calle, numero, escalera, planta, puerta → *metros_p, od_piso, dni_p*

Tabla *Persona*: esta tabla se encuentra en *FN1*, puesto que todos los atributos son atómicos. También se encuentra en *FN2* por no existir dependencias funcionales no completas entre los atributos no primos y la única clave de esta tabla (el

atributo *dni*). Satisface la *FN3* al no existir dependencias funcionales transitivas, y al no existir más que un determinante funcional, también satisface la *FNBC*.

dni → *nombre_persona, apellidos_persona, od_persona dni_c, calle, numero*

Tabla *HabitaPiso*: esta tabla se encuentra en *FNBC*, pues todos los atributos no primos dependen de forma completa del único determinante funcional.

dni → *calle, numero, escalera, planta, puerta*

6.5. CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

Bajo esta nueva visión más refinada del problema, el esquema relacional propuesto quedaría de la forma siguiente (ver figura 6.3):

BASE DE DATOS DEL CATASTRO MUNICIPAL

ZonaUrbana	*(nombre_zona, od_zona)*
Vivienda	*(calle, numero, tipo_vivienda, codigo_postal, metros, od_vivienda,* **nombre_zona***)*
BloqueCasas	*(calle, numero, metros_b, od_bloque)*
CasaParticular	*(calle, numero, metros_c, od_casa,* **dni_cp***)*
Persona	*(dni, nombre_persona, apellidos_persona, od_persona,* **dni_c, calle, numero***)*
Piso	*(calle, numero, escalera, planta, puerta, metros_p, od_piso,* **dni_p***)*
HabitaPiso	*(dni,* **calle, numero, escalera, planta, puerta***)*

6.5.1. Definición sintáctica de las tablas

```
/* Inicialmente, se borran las tablas por si éstas han sido
   creadas con anterioridad. Las tablas deben ser borradas
   teniendo en cuenta las relaciones entre ellas. Como norma
   general es conveniente borrarlas en el orden inverso a como
   se crean */
DROP TABLE HabitaPiso;
DROP TABLE CasaParticular;
DROP TABLE Piso;
DROP TABLE Persona;
DROP TABLE BloqueCasas;
DROP TABLE Vivienda;
DROP TABLE ZonaUrbana;
```



```
/* Se crean las tablas del esquema propuesto */
CREATE TABLE ZonaUrbana (
   nombre_zona VARCHAR2(20) NOT NULL,
   od_zona LONG,
     CONSTRAINT pk_zon
       PRIMARY KEY (nombre_zona),
     CONSTRAINT ck_zon
       CHECK (nombre_zona = UPPER(nombre_zona)) );
CREATE TABLE Vivienda (
   calle VARCHAR2(40) NOT NULL,
   numero NUMBER(3) NOT NULL,
   tipo_vivienda VARCHAR2(1),
   codigo_postal NUMBER(5),
   metros NUMBER(5),
   od_vivienda LONG,
   nombre_zona VARCHAR2(20) NOT NULL,
     CONSTRAINT pk_viv
       PRIMARY KEY (calle, numero),
     CONSTRAINT ck_num
       CHECK (numero > 0),
     CONSTRAINT ck_tv
       CHECK (tipo_vivienda IN ('C', 'B')),
     CONSTRAINT fk_viv_zon
       FOREIGN KEY (nombre_zona)
       REFERENCES ZonaUrbana(nombre_zona) );
CREATE TABLE Persona (
   dni NUMBER(8) NOT NULL,
   nombre_persona VARCHAR2(20) NOT NULL,
   apellidos_persona VARCHAR2(40) NOT NULL,
   od_persona LONG,
   dni_c NUMBER(8) NOT NULL,
   calle VARCHAR2(30) NOT NULL,
   numero NUMBER(3) NOT NULL,
     CONSTRAINT pk_per
       PRIMARY KEY (dni),
     CONSTRAINT fk_per_viv
       FOREIGN KEY (calle, numero)
       REFERENCES Vivienda(calle, numero),
     CONSTRAINT fk_per_per
       FOREIGN KEY (dni_c)
       REFERENCES Persona(dni) );
CREATE TABLE BloqueCasas (
   calle VARCHAR2(30) NOT NULL,
   numero NUMBER(3) NOT NULL,
   metros_b NUMBER(4),
   od_bloque LONG,
     CONSTRAINT pk_blo
       PRIMARY KEY (calle, numero),
```

```
    CONSTRAINT fk_blo_viv
      FOREIGN KEY (calle, numero)
      REFERENCES Vivienda(calle, numero)
      ON DELETE CASCADE,
    CONSTRAINT ck_mb
      CHECK (metros_b > 0) );
CREATE TABLE CasaParticular (
   calle VARCHAR2(30) NOT NULL,
   numero NUMBER(3) NOT NULL,
   metros_c NUMBER(4),
   od_casa LONG,
   dni_cp NUMBER(8) NOT NULL,
    CONSTRAINT pk_cas
      PRIMARY KEY (calle, numero),
    CONSTRAINT fk_cas_per
      FOREIGN KEY (dni_cp)
      REFERENCES Persona(dni),
    CONSTRAINT fk_cas_viv
      FOREIGN KEY (calle, numero)
      REFERENCES Vivienda(calle, numero)
      ON DELETE CASCADE,
    CONSTRAINT ck_mt
      CHECK (metros_c > 0) );
CREATE TABLE Piso (
   calle VARCHAR2(30) NOT NULL,
   numero NUMBER(3) NOT NULL,
   escalera VARCHAR2(1) NOT NULL,
   planta NUMBER(2) NOT NULL,
   puerta VARCHAR2(2) NOT NULL,
   metros_p NUMBER(3),
   od_piso LONG,
   dni_p NUMBER(8) NOT NULL,
    CONSTRAINT pk_pis
      PRIMARY KEY (calle, numero, escalera, planta, puerta),
    CONSTRAINT ck_mep
      CHECK (metros_p > 0),
    CONSTRAINT fk_pis_blo
      FOREIGN KEY (calle, numero)
      REFERENCES BloqueCasas(calle, numero)
      ON DELETE CASCADE,
    CONSTRAINT fk_pis_per
      FOREIGN KEY (dni_p)
      REFERENCES Persona(dni) );
CREATE TABLE HabitaPiso (
   dni NUMBER(8) NOT NULL,
   calle VARCHAR2(30) NOT NULL,
   numero NUMBER(3) NOT NULL,
   escalera VARCHAR2(1) NOT NULL,
   planta NUMBER(2) NOT NULL,
```



```
   puerta VARCHAR2(2) NOT NULL,
    CONSTRAINT pk_hap
      PRIMARY KEY (dni),
    CONSTRAINT fk_hap_cas
      FOREIGN KEY (calle, numero, escalera, planta, puerta)
      REFERENCES Piso(calle, numero, escalera, planta, puerta)
      ON DELETE CASCADE,
    CONSTRAINT fk_hbp_per
      FOREIGN KEY (dni)
      REFERENCES Persona(dni)
      ON DELETE CASCADE );
```

6.5.2. Manipulación de la Base de Datos

Cap06ej01.sql

Obtener el DNI de todos los propietarios de una casa en la zona Centro.

Se obtiene de la tabla Vivienda el número y la calle de todas las viviendas situadas en la zona 'Centro', gracias a lo cual se puede obtener en la consulta sobre la tabla CasaParticular el dni de todos los propietarios de una casa en dicha zona.

```
SELECT DISTINCT dni_cp "D.N.I. Propietarios"
   FROM CasaParticular
   WHERE (calle, numero) IN
     (SELECT calle, numero
        FROM Vivienda
        WHERE nombre_zona = 'CENTRO');
```

Resultado

```
     D.N.I. Propietarios
     -------------------
               70100506
```

Cap06ej02.sql

Obtener todos los propietarios de un piso en el bloque de casas de la calle Damasco, número 20.

De la tabla Piso se obtiene el DNI de todos los propietarios de un piso en la calle 'Damasco', número 20. Esto se utiliza para obtener de la tabla Persona el nombre y los apellidos del titular de dicho dni los cuales se presentan ordenados.

```
SELECT dni "D.N.I.", nombre_persona "Nombre",
       apellidos_persona "Apellidos"
   FROM Persona
   WHERE dni IN
      (SELECT dni_p
       FROM Piso
       WHERE calle = 'Damasco' AND numero = 20)
   ORDER BY apellidos_persona, nombre_persona;
```

Resultado

```
       D.N.I. Nombre               Apellidos
     --------- -------------------- --------------------
     32602050 Antonio              Fernández Fernández
     70800200 Teresa               Gutiérrez Pérez
     60606070 Pedro                Jiménez Cruz
```

Cap06ej03.sql

Obtener todos los pisos de más de 50 metros cuadrados cuyo propietario tiene el D.N.I 44351312.

Se obtienen los campos calle, numero, escalera, planta, puerta y metros cuadrados de la tabla Piso para aquellas tuplas que cumplen las condiciones de que el D.N.I del propietario sea 44351312 y que el valor del campo metros_p sea superior a 50.

```
SELECT calle "Calle", numero "Número", escalera "E",
planta "Planta", puerta "PU", metros_p "Metros"
   FROM Piso
   WHERE dni_p = 44351312 AND metros_p > 50;
```

Resultado

```
Calle                          Número E  Planta PU Metros
------------------------------ ------ - ------ -- -------
Ronda de los Tejares               15 2      2  3      70
```

Cap06ej04.sql

Obtener todas las zonas de la ciudad en las que sólo existen casas particulares.

Debido a que la información sobre las zonas urbanas se encuentra en la tabla Vivienda, se procede a la reunión natural de esta tabla con las tablas CasaParticular y BloqueCasas. El proceso consiste en obtener las zonas en las que existen viviendas y en las que no existen bloques, ayudándonos para ello del operador IN.

```
SELECT DISTINCT nombre_zona
   FROM CasaParticular, Vivienda
   WHERE Vivienda.calle=CasaParticular.calle AND
         Vivienda.numero=CasaParticular.numero AND
         nombre_zona NOT IN
      (SELECT DISTINCT nombre_zona
       FROM BloqueCasas, Vivienda
       WHERE Vivienda.calle = BloqueCasas.calle AND
             Vivienda.numero = BloqueCasas.numero );
```

Resultado

```
NOMBRE_ZONA
--------------------
SUBURBIO TERMINAL
TRASSIERRA
```

Cap06ej05.sql

Obtener el número de domicilios particulares que existen en el bloque Felipe II, número 14.

El operador COUNT devuelve el número de tuplas obtenido como resultado de una consulta. En esta consulta concreta se devolverá el número de tuplas de la tabla Piso que cumplen la condición solicitada.

```
SELECT COUNT (*) "Total Domicilios Particulares"
   FROM Piso
   WHERE calle = 'Felipe II' AND numero = 14;
```

Resultado

```
Total Domicilios Particulares
-----------------------------
                            2
```


Cap06ej06.sql

En la calle Cruz Conde, número 20, la escalera B pasa a denominarse 2.

Para realizar esta actualización en la tabla Piso, es necesario deshabilitar la restricción de referencia que existe con la tabla HabitaPiso. Una vez hecha la actualización se procede a habilitar de nuevo la restricción.

Comprobación *(realizamos una comprobación del estado previo)*

```
SELECT calle "Calle", numero "Número", escalera "Escalera",
       planta "Planta"
   FROM Piso
   WHERE numero = 20 AND calle = 'Cruz Conde';
```

Resultado

```
Calle                          Número Escalera Planta
------------------------------ ------ -------- -------
Cruz Conde                         20 B              1
Cruz Conde                         20 B              2
```

Sentencias

```
ALTER TABLE HabitaPiso
   DISABLE CONSTRAINT fk_hap_cas CASCADE;
UPDATE HabitaPiso
   SET escalera = '2'
      WHERE escalera = 'B' AND calle = 'Cruz Conde'
            AND numero = 20;
UPDATE Piso
   SET escalera = '2'
      WHERE escalera = 'B' AND calle = 'Cruz Conde'
      AND numero = 20;
ALTER TABLE HabitaPiso ENABLE
   CONSTRAINT fk_hap_cas;
```

Comprobación *(realizamos una comprobación del estado final)*

Resultado

```
Calle                          Número Escalera Planta
------------------------------ ------ -------- -------
Cruz Conde                         20 2              1
Cruz Conde                         20 2              2
```

Cap06ej07.sql

Acelerar los accesos a las personas por sus nombres y apellidos.

Para acelerar las consultas sobre una tabla por alguno o algunos de sus campos se recurre a los índices. Un índice es un objeto de la base de datos que contiene una entrada para cada valor que aparece en las columnas de la tabla indexada, proporcionando accesos rápidos y directos a las mismas. Sin embargo, las operaciones de inserción (INSERT), modificación (UPDATE) y borrado (DELETE) son más costosas computacionalmente, ya que Oracle debe gestionar tanto los datos del índice como los de la tabla.

```
DROP INDEX i_persona;
CREATE INDEX i_persona ON
   Persona (apellidos_persona, nombre_persona);
```

Resultado

```
Índice creado.
```

Cap06ej08.sql

Obtener información de los metros cuadrados de solar, agrupados por zonas urbanas y ordenados por el número total de metros de cada zona urbana.

Se obtienen de la tabla Vivienda los atributos metros y nombre_zona, y el resultado es agrupado por nombre_zona gracias a la cláusula GROUP BY, y ordenado en función de los metros gracias a la cláusula ORDER BY.

```
SELECT SUM(metros) "Metros", nombre_zona "Zona Urbana"
   FROM Vivienda
   GROUP BY nombre_zona
   ORDER BY 1;
```

Resultado

```
    Metros Zona Urbana
---------- --------------------
       200 SUBURBIO TERMINAL
       240 BRILLANTE
       390 SECTOR SUR
       400 CENTRO
       500 TRASSIERRA
```

Cap06ej09.sql

Borrar aquellas casas particulares de menos de 100 metros cuadrados construidos que estén situadas en la zona urbana 'SUBURBIO TERMINAL'.

Se obtiene de la tabla Vivienda los atributos calle y numero de todas aquellas viviendas cuyo atributo nombre_zona tiene el valor 'Suburbio Terminal'. El resultado de esta subconsulta se utiliza para borrar de la tabla CasaParticular todas aquellas tuplas que cumplen la condición del problema.

Comprobación *(realizamos una comprobación del estado previo)*

```
SELECT calle "Calle", numero "Número", metros_c "Metros"
   FROM CasaParticular
   WHERE (calle, numero) IN
      (SELECT calle, numero
          FROM Vivienda
          WHERE zona_urbana = 'SUBURBIO TERMINAL');
```

Resultado

```
Calle                          Número  Metros
------------------------------ --------- -------
El Palo                            15      50
Guerra                              2     150
```

Sentencias

```
DELETE CasaParticular
   WHERE (calle, numero) IN
      (SELECT calle, numero
          FROM Vivienda
          WHERE nombre_zona = 'SUBURBIO TERMINAL')
   AND metros_c < 100;
```

Resultado

```
1 fila borrada.
```

Comprobación *(realizamos una comprobación del estado final)*

Resultado

```
CALLE                          NUMERO METROS_C
------------------------------ --------- --------
Guerra                              2      150
```

Cap06ej10.sql

Obtener el tamaño medio de las casas particulares existentes en la zona urbana 'SECTOR SUR'.

Se obtienen de la tabla CasaParticular el valor medio del atributo metros_c de todas las tuplas que cumplen la condición de que los atributos calle y numero tengan el valor devuelto por una subconsulta sobre la tabla Vivienda de aquellas tuplas que cumplen la condición de que nombre_zona sea igual a 'SECTOR SUR'.

```
SELECT AVG(metros_c) "Tamaño medio de las casas"
   FROM CasaParticular
   WHERE (calle, numero) IN
      (SELECT calle, numero
          FROM Vivienda
          WHERE nombre_zona = 'SECTOR SUR');
```

Resultado

```
Tamaño medio de las casas
-------------------------
                      105
```

Cap06ejP1.sql

Preparar un procedimiento PL/SQL que inserte una vivienda en la base de datos.

Insertar una vivienda supone insertar una tupla en la tabla Vivienda y otra o bien en la tabla CasaParticular o en BloqueCasas, en función del valor del atributo tipo_vivienda. Si en alguna de las inserciones se produce algún error (clave primaria repetida, clave foránea no existente, tipo de vivienda incorrecto, etc.) Oracle genera una excepción cuyo mensaje es contenido dentro de la función SQLERRM. De esta forma comprobando dentro del bloque de tratamiento de las excepciones el nombre de cada una de las restricciones que se pueden violar en el proceso de inserción, se sabrá donde se ha producido el error y se gestionará el mensaje correspondiente.

```
CREATE OR REPLACE PROCEDURE ins_vivienda
   /* Se definen las variables de la interfaz */
   (p_nombre_zona VARCHAR2,
   p_calle VARCHAR2,
   p_numero NUMBER,
   p_tipo_vivienda CHAR,
   p_codigo_postal NUMBER,
   p_metros NUMBER,
   p_od_vivienda VARCHAR2,
   p_dni NUMBER DEFAULT NULL)
IS
BEGIN
   /* Se inserta en la tabla Vivienda */
   INSERT INTO Vivienda
              (calle,numero,tipo_vivienda,codigo_postal,
              metros, od_vivienda, nombre_zona)
      VALUES (UPPER(p_calle),p_numero, p_tipo_vivienda,
```

```
            p_codigo_postal, p_metros, p_od_vivienda,
            UPPER(p_nombre_zona));
   /* Se comprueba el tipo de vivienda */
   IF p_tipo_vivienda='B' THEN
   /* Se inserta en Bloque de viviendas */
      INSERT INTO BloqueCasas (calle, numero, metros_b,
                  od_bloque)
         VALUES (UPPER(p_calle), p_numero, p_metros,
                 p_od_vivienda);
   ELSE
   /* Se inserta en Casa Particular */
      INSERT INTO CasaParticular (calle, numero, metros_c,
                  od_casa,dni_cp)
         VALUES (UPPER(p_calle), p_numero, p_metros,
                 p_od_vivienda, p_dni);
   END IF;
COMMIT;
   /* Se comprueban los posibles errores en el proceso */
EXCEPTION
   /*  Se comparan las cadenas de error que devuelve el sistema
   con las diferentes restricciones (CONSTRAINTS) definidas en
   el esquema. De esta forma se puede reconocer el tipo de
   error producido e informar al usuario del mismo, Cada
   restricción es reconocida por el nombre de la base de datos
   seguida por el nombre de la restricción. En los ejemplos que
   utilizará el lector haciendo uso del material incluido en el
   CD la base de datos se denomina CHENCODD */
   WHEN OTHERS THEN
      IF INSTR(SQLERRM,'CHENCODD.FK_VIV_ZON')<> 0 THEN
         DBMS_OUTPUT.PUT_LINE('La zona de la vivienda
         introducida no existe');
      END IF;
      IF INSTR(SQLERRM,'CHENCODD.PK_VIV')<> 0 THEN
         DBMS_OUTPUT.PUT_LINE('Esta vivienda ya ha sido
         introducida en la Base de Datos');
      END IF;
      IF INSTR(SQLERRM,'CHENCODD.CK_TV')<> 0 THEN
         DBMS_OUTPUT.PUT_LINE('El tipo de vivienda:
         '||p_tipo_vivienda||' no existe');
      END IF;
      IF INSTR(SQLERRM,'CHENCODD.FK_CAS_PER')<> 0 THEN
         DBMS_OUTPUT.PUT_LINE('La persona que se introduce como
         dueño no existe');
      END IF;
   ROLLBACK;
END;
/
-- No existe zona
    EXEC ins_vivienda ('KKD','LaS Paz',5,'C',14600,200,'Prueba de
    error para el caso de una zona no existente',44351312);
```



```
-- Existe la vivienda
    EXEC ins_vivienda ('BRILLANTE','Avenida el Brillante', 15,'C',
    14600,200,'Prueba de error para el caso de una vivienda no
    existente',44351312);
-- Tipo de vivienda no existente
    EXEC ins_vivienda ('BRILLANTE','La Paz',523,'K',14600,200,'',
    44351312);
-- Propietario no existente
    EXEC ins_vivienda ('BRILLANTE','La Paz',25,'C',14600,200,'',
    443534);
-- Datos correctos
    EXEC ins_vivienda ('BRILLANTE','La Paz',25,'C',14600,200,'',
    44351312);
```

CAPÍTULO 7

LOS RESIDUOS TÓXICOS

7.1. ENUNCIADO DEL PROBLEMA

Se desea abordar la problemática ambiental de los residuos tóxicos y peligrosos cuya incorrecta gestión produce daños de gran importancia en el medio ambiente y en la salud del ser humano. La información a contemplar es la que corresponde desde que es producido el residuo por un centro o empresa productora hasta que éste se encuentra en lugar seguro, en donde recibe un tratamiento especial como puede ser la incineración, almacenamiento en depósitos de seguridad, etc. En el sistema de información se desea considerar la información de los productores de residuos, los residuos, las empresas que transportan los residuos hasta los lugares seguros y el traslado de los residuos teniendo en cuenta el tipo de transporte, el envase, etc.

Es conveniente tener en cuenta los siguientes supuestos semánticos:

SUPUESTO 1: Una empresa productora produce un número amplio de residuos constituidos por un número variable de constituyentes químicos.

SUPUESTO 2: Más de una empresa productora puede producir residuos con igual número de constituyentes químicos y con las mismas o distintas cantidades.

SUPUESTO 3: Las empresas productoras asignan un código único a los residuos que producen, lo que les permite diferenciar distintas producciones de los mismos

productos. Además, más de una empresa puede asignar el mismo código a los residuos que produce.

SUPUESTO 4: Los residuos pueden ser trasladados en su totalidad (cantidad total del mismo) o en partes, o no ser trasladados nunca.

SUPUESTO 5: En cada traslado de residuos la cantidad que se traslada de los mismos es enviada a un único destino.

SUPUESTO 6: En una misma fecha las empresas productoras pueden ordenar más de un traslado de un mismo o distinto residuo (cantidades parciales del mismo) a un mismo o distinto destino.

SUPUESTO 7: En cada traslado puede intervenir más de una empresa transportista usando el mismo o distinto transporte, por lo que resulta interesante conocer tanto el medio de transporte utilizado como los kilómetros realizados, así como el coste del trabajo.

SUPUESTO 8: El residuo se traslada en un tipo de envase determinado por la empresa productora y que no varía a lo largo de su traslado.

SUPUESTO 9: Es interesante conocer la fecha de llegada a destino y el tratamiento a que se someten los residuos una vez alcanzado el mismo.

SUPUESTO 10: Por seguridad se considera que en un traslado sólo puede trasladarse un residuo de una empresa productora.

7.2. MODELO CONCEPTUAL

Se trata de representar la información correspondiente al traslado de los residuos producidos por las empresas productoras de los mismos hasta un lugar seguro donde son tratados adecuadamente. Considerando el enunciado del problema abordado, se puede extraer los siguientes tipos de entidad, como se muestra en la figura 7.1:

7.2.1. Análisis de los tipos de entidad

Tipo de entidad *EmpresaProductora*: el cual representa al objeto del mundo real *"empresa que se dedica a una cierta actividad que produce unos determinados residuos tóxicos"*. Para este tipo de entidad se consideran los atributos *nif_empresa, nombre_empresa, ciudad_empresa, actividad* y *od_empresa* (ENUNCIADO).

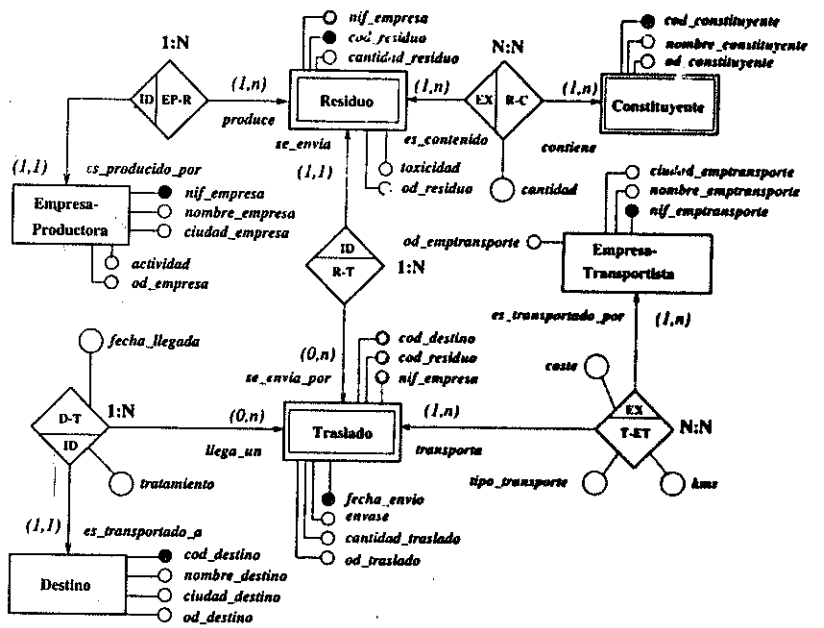


Figura 7.1 Esquema E-R del modelo conceptual del problema de los residuos

El atributo *nif_empresa* es el identificador de este tipo de entidad, ya que no pueden existir dos empresas con el mismo número de identificación fiscal, el atributo *actividad* representa la actividad a la que se dedica la empresa y *od_empresa* hace referencia a cualquier otro dato que pueda ser de importancia para el sistema de información.

Tipo de entidad *Residuo*: el cual representa al objeto del mundo real *"material peligroso o tóxico producido por la actividad de una determinada empresa"* (SUPUESTO 1). Este tipo de entidad se puede considerar como un tipo de entidad débil por identificación respecto al tipo de entidad *EmpresaProductora*, ya que un residuo no puede existir si no existe una empresa que lo produzca; además para identificar una entidad de este tipo es necesario conocer la empresa que lo produce, ya que esto permite diferenciar entre los mismos residuos producidos por diferentes empresas (SUPUESTO 3).

Para este tipo de entidad se consideran los atributos *nif_empresa, cod_residuo, toxicidad, cantidad_residuo* y *od_residuo*. El identificador del tipo de entidad es la agregación de los atributos *nif_empresa* y *cod_residuo*, puesto que para identificar una entidad de este tipo, además de conocer la empresa que lo produce es necesario un código para diferenciar los residuos producidos por una misma empresa, ya que un residuo puede ser producido más de una vez, y aunque el nombre es el mismo, el código es diferente (SUPUESTO 3).

El atributo *od_residuo* hace referencia a cualquier información que se desee considerar de este tipo de entidad, el atributo *toxicidad* hace referencia al nivel de toxicidad o peligrosidad que pueda tener un determinado residuo dentro de un rango establecido, y el atributo *cantidad_residuo* hace referencia a la cantidad total de residuo presente. Este último atributo es importante, ya que un determinado residuo puede ser dividido en partes para su traslado (SUPUESTO 4).

Tipo de entidad *Constituyente*: el cual representa al objeto del mundo real "***compuesto químico o elemento que da al residuo su carácter peligroso***". Se consideran los atributos *cod_constituyente*, *nombre_constituyente* y *od_constituyente*. El atributo *cod_constituyente* identifica sin ambigüedad a un determinado elemento o compuesto químico, por lo que se considera identificador del tipo de entidad. El atributo *od_constituyente* representa cualquier información que se desee considerar (SUPUESTOS 1 y 2).

Se trata de un tipo de entidad débil por existencia con respecto al tipo de entidad *Residuo*, pues se ha considerado que el atributo *cod_constituyente* identifica sin ambigüedad a una entidad de este tipo, y en los supuestos no se indica que se mantenga información de otros constituyentes que no estén presentes en los residuos existentes.

Tipo de entidad *EmpresaTransportista*: el cual representa al objeto del mundo real "***empresa encargada del traslado del residuo desde la empresa productora al destino seguro***". Para este tipo de entidad se consideran los atributos *nif_emptransporte*, *nombre_emptransporte*, *ciudad_emptransporte* y *od_emptransporte*. El identificador de este tipo de entidad es el atributo *nif_emptransporte*, ya que no existen dos empresas con el mismo número de identificación fiscal. El atributo *od_emptransporte* es cualquier información de interés que se desee considerar (SUPUESTO 7).

Tipo de entidad *Destino*: representa al objeto del mundo real "*empresa o lugar determinado donde se trasladan los residuos para su posterior tratamiento*". En este tipo de entidad se consideran los atributos *cod_destino*, *nombre_destino*, *ciudad_destino* y *od_destino*. El atributo *cod_destino* es el identificador de este tipo de entidad. Este atributo puede representar el número de identificación fiscal si el destino se trata de una empresa, o un código especial si por ejemplo se trata de un cementerio nuclear. El atributo *od_destino* representa cualquier otra información de interés (SUPUESTO 5).

Tipo de entidad *Traslado*: el cual representa al objeto del mundo real "*envío de un determinado residuo desde la empresa productora hasta un lugar seguro*". Es el tipo de entidad más complejo de representar. Es obvio que si no existe un residuo no podrá existir un traslado del mismo y, según el SUPUESTO 10, sólo es trasladado un residuo en un traslado/envío, pero según el SUPUESTO 6, en una misma fecha se pueden ordenar varios traslados del mismo o distinto residuo y, para añadir complejidad al problema, se ha considerado que distintas empresas productoras pueden ordenar traslados en la misma fecha de cualquier residuo y,

no se puede olvidar, que según el SUPUESTO 7, más de una empresa transportista puede intervenir en cada traslado, aunque sólo se considera el último destino del residuo (SUPUESTO 5). Por todo ello, está claro que:

- El tipo de entidad *Traslado* es débil por existencia con respecto a los tipos de entidad *Residuo*, *EmpresaTransportista* y *Destino*, pues si ninguno de ellos existiera no se podría realizar un traslado.
- Si se considera que el tipo de entidad *Traslado* es débil por identificación con respecto al tipo de entidad *Residuo* debido a los: SUPUESTOS 2, 3 y 6, entonces este tipo de entidad heredará los atributos identificadores del tipo de entidad *Residuo*.
- Además, incorporará el atributo *cantidad_residuo* debido al SUPUESTO 4, y el atributo *envase* debido al SUPUESTO 8.

Pero se plantea un problema debido al SUPUESTO 6, que informa que un residuo puede trasladarse en la misma fecha, luego la inclusión del atributo *fecha* como identificador del tipo de entidad *Traslado* es necesario aunque no suficiente a no ser que se introduzca una nueva restricción.

SUPUESTO 11: Si una empresa en una misma fecha realiza más de un traslado de partes de un mismo residuo, éstos deberán ir a destinos diferentes. En caso contrario se considerarán como el mismo traslado

Esta restricción es lógica ya que se considera que en un traslado pueden intervenir varias empresas transportistas (SUPUESTO 7). Luego si en un mismo día se realizan varios traslados de un mismo residuo, puede considerarse que se trata de un mismo traslado en el cual intervienen varias empresas para su transporte. Claro está, a no ser que los traslados se realicen a destinos diferentes, en cuyo caso sí deben considerarse como distintos traslados.

En base a estas premisas se puede considerar que el identificador del tipo de entidad *Traslado* es la agregación de los atributos:

1. *nif_empresa* y *cod_residuo* heredados del tipo de entidad *Residuo* con el cual mantiene una relación débil por identificación (SUPUESTO 6).
2. *fecha_envio* o fecha en que se ordenó el traslado del residuo.
3. *cod_destino* heredado del tipo de entidad *Destino* por imposición del SUPUESTO 11.

Además, este tipo de entidad incorporará los atributos *cantidad_residuo* (SUPUESTO 4), *envase* (SUPUESTO 8) y *od_traslado*, para representar la cantidad de

residuo trasladado, el envase en el que se traslada y cualquier otra información de interés sobre el traslado, respectivamente[18].

7.2.2. Análisis de los tipos de interrelación

Los tipos de entidad antes mencionados se encuentran relacionados de la siguiente forma:

Tipo de interrelación *EmpresaProductora/Residuo* **(EP-R)**: el cual relaciona los tipos de entidad *EmpresaProductora* y *Residuo*. La relación es uno a muchos, como se muestra en la figura 7.1, ya que un determinado residuo sólo es producido por una y sólo una empresa, participando, por tanto, el tipo de entidad *EmpresaProductora* con las cardinalidades *(1,1)* (SUPUESTO 1).

Por otro lado, para que una determinada empresa tenga existencia en el sistema de información debe producir como mínimo un residuo, pudiendo producir muchos residuos, por lo que el tipo de entidad *Residuo* participa en la interrelación con cardinalidades *(1,n)*.

Tipo de interrelación *Residuo/Constituyente* **(R-C)**: este tipo de interrelación *N:N* relaciona los tipos de entidad *Residuo* y *Constituyente*, puesto que en un determinado residuo existe como mínimo un constituyente que le da cierta peligrosidad o bien pueden ser varios los que se incluyan en el residuo, por lo que el tipo de entidad *Constituyente* participa en la interrelación con cardinalidades *(1,n)* (SUPUESTOS 1 y 2). Un determinado constituyente o compuesto químico no tiene existencia propia a no ser que se encuentre incluido en un residuo, por lo que el tipo de entidad *Residuo* participa en la interrelación con cardinalidades *(1,n)*.

Este tipo de interrelación tiene un atributo llamado *cantidad*, que representa la cantidad (tomada en el rango que se desee) de un determinado constituyente que está incluida en un residuo (ver figura 7.1).

Tipo de interrelación *Residuo/Traslado* **(R-T)**: el cual representa la relación existente entre los tipos de entidad *Residuo* y *Traslado*. Se considera que un residuo producido por una empresa puede no ser transportado, transportado de una sola vez o ser transportado en partes, por lo que el tipo de entidad *Traslado* participa en la interrelación con cardinalidades *(0,n)* (SUPUESTO 4). Además, se ha considerado que en un determinado traslado se envía uno y sólo un residuo, por lo que el tipo de entidad *Residuo* participa en la interrelación con cardinalidades *(1,1)* (SUPUESTO 10).

Tipo de interrelación *Traslado/EmpresaTransportista* **(T-ET)**: el cual representa la relación existente entre los tipos de entidad *Traslado* y *EmpresaTransportista*, el tipo de interrelación es del tipo *N:N*. Se considera que un traslado o envío de

[18] El lector puede apreciar que este tipo de entidad constituye básicamente una abstracción del tipo de interrelación ternaria que existe entre los tipos de entidad *Residuo, Destino* y *EmpresaTransportista*.

cierto residuo puede ser realizado por varias empresas de transporte, pero siempre por una como mínimo (si no, no habría traslado), por lo que el tipo de entidad *EmpresaTransportista* participa en la interrelación con cardinalidades *(1,n)* (SUPUESTO 7). El número de traslados que puede realizar una determinada empresa de transportes son muchos, pero para que una empresa de transportes tenga existencia dentro del sistema de información ha de realizar como mínimo un traslado, por lo que el tipo de entidad *Traslado* participa en la interrelación con cardinalidades *(1,n)*.

Este tipo de interrelación tiene tres atributos llamados *tipo_transporte*, *kms* y *coste*, en los que se considera el tipo de transporte que ha realizado un determinado traslado de un residuo o parte de éste, los *kilómetros* que lo ha transportado y el coste que supone ese traslado, respectivamente.

Tipo de interrelación *Destino/Traslado* **(D-T)**: el cual representa la relación existente entre los tipos de entidad *Destino* y *Traslado*. Se considera que un envío sólo va dirigido a un determinado destino, por tanto, el tipo de entidad *Destino* participa en la interrelación con cardinalidades *(1,1)*, mientras que a un destino puede que no llegue ningún residuo o, por el contrario, sean varios los que lleguen. Así, el tipo de entidad *Traslado* participa en la interrelación con cardinalidades *(0,n)* (SUPUESTOS 6 y 11).

Se trata de un tipo de interrelación *1:N*, y tiene dos atributos, *fecha_llegada* y *tratamiento*, representando la fecha en la que llega un determinado residuo al destino así como el tratamiento que se le efectúa, respectivamente (SUPUESTO 9).

Se va a presentar a continuación otra visión posible del problema únicamente para intentar introducir al lector en el proceso de análisis de problemas y descripción de los elementos del mismo. Como se podrá observar a lo largo de esta sección, cualquier modificación en algún requisito puede dar lugar a un cambio más o menos significativo en las propiedades del sistema que se intenta describir.

Una opción que simplificaría el esquema conceptual del problema que está siendo tratado sería el considerar que cada traslado puede ser identificado a través de un atributo *cod_traslado*. Un código que es asignado por cada empresa productora de residuos. En este caso se tendría que:

- El tipo de entidad *Traslado* mantiene una relación de debilidad por identificación con el tipo de entidad *EmpresaProductora* a través de la cual hereda su identificador que, agregado con un identificador propio (por ejemplo, un número consecutivo que diferenciara los traslados), identificará las diferentes instancias de este tipo de entidad. Esta estructura del identificador del tipo de entidad *Traslado* es necesaria debido a la imposición del SUPUESTO 6.
- El tipo de entidad *Traslado* mantiene una relación de debilidad por existencia con los tipos de entidad *Destino*, *Residuos* y *EmpresaTransportista*.
- Y, entonces, los atributos del tipo de entidad *Traslado* serían:

- *nif_empresa* + *cod_traslado* como identificadores, puesto que hereda el atributo *nif_empresa* del tipo de entidad *EmpresaProductora* con el que participa en un tipo de interrelación débil por identificación[19].
- *fecha_envio*, impuesto por el SUPUESTO 6.
- *envase*, impuesto por el SUPUESTO 8.
- *cantidad_traslado*, impuesto por el SUPUESTO 4.
- Y, *od_traslado* para representar cualquier otra información de interés.

Aunque se trata de una visión algo más simplificada, se puede observar que no aporta ninguna mejora al esquema el introducir un atributo externo no explicitado en ningún supuesto al dominio del problema, puesto que el atributo *cod_traslado* debe representar a los distintos residuos que son traslados por cada empresa productora. Además, si se utiliza este criterio sería necesario realizar otros cambios en el esquema conceptual para que no se obtuviera en el esquema relacional resultante una duplicidad del atributo *nif_empresa* en la tabla generada para el tipo de entidad *Traslado*. Por ello, se seguirá la resolución del problema planteado sin considerar esta simplificación.

7.3. MODELO RELACIONAL

Una vez realizado el modelo conceptual del problema abordado se va a realizar el modelo relacional del mismo, el cual se muestra en la figura 7.2.

7.3.1. Tabla *EmpresaProductora*

La tabla *EmpresaProductora* se forma a partir del tipo de entidad del mismo nombre, y toma de ésta los atributos *nif_empresa*, *nombre_empresa*, *ciudad_empresa*, *actividad* y *od_empresa* (regla *RTECAR-1*). Del conjunto de atributos de la tabla, la clave principal es el atributo *nif_empresa*, no considerándose ningún otro atributo como clave alterna.

Ninguno de los atributos de esta tabla mantiene referencia alguna con atributos de las otras tablas. Por lo tanto, la tabla *EmpresaProductora* queda de la forma siguiente:

EmpresaProductora *(nif_empresa, nombre_empresa, ciudad_empresa, actividad, od_empresa)*

7.3.2. Tabla *Residuo*

La tabla *Residuo* se forma a partir del tipo de entidad *Residuo (RTECAR-1)* y por la aplicación de la regla *RTECAR-3.1* al tipo de interrelación *(EP-R)*, estando

[19] Recuerde que el signo + se utiliza para representar la agregación de atributos.

formada por los atributos *nif_empresa*, *cod_residuo*, *toxicidad*, *cantidad_residuo* y *od_residuo*.

El atributo *nif_empresa* mantiene referencia con el identificador de la tabla *EmpresaProductora* debido a la interrelación de identificación que existe entre los dos tipos de entidad. El identificador de la tabla *Residuo* es la agregación de los atributos *nif_empresa* y *cod_residuo*, no considerándose ningún otro atributo como clave alterna. La tabla queda de la forma:

Residuo *(nif_empresa, cod_residuo, toxicidad, cantidad_residuo, od_residuo)*

7.3.3. Tabla *Constituyente*

La tabla *Constituyente* se forma a partir del tipo de entidad *Constituyente* (regla *RTECAR-1*). Esta tabla tiene los atributos *cod_constituyente*, *nombre_constituyente* y *od_constituyente*, que son tomados del tipo de entidad antes mencionado.

Ninguno de los atributos mantiene referencia alguna con los atributos de las otras tablas. El identificador de la tabla es el atributo *cod_constituyente*. La tabla *Constituyente* queda de la forma siguiente:

Constituyente *(cod_constituyente, nombre_constituyente, od_constituyente)*

7.3.4. Tabla *Residuo_Constituyente*

Esta tabla se forma por la existencia del tipo de interrelación *(R-C)* entre los tipos de entidad *Residuo* y *Constituyente*, y está formada por los atributos *nif_empresa*, *cod_residuo*, *cod-constituyente* y *cantidad* (regla *RTECAR-4*), de la forma siguiente:

- Los atributos *nif_empresa* y *cod_residuo* son tomados del tipo de entidad *Residuo* por ser los identificadores de éste, y a través de estos atributos mantiene referencia con la tabla *Residuo*.
- El atributo *cod_constituyente* es tomado del tipo de entidad *Constituyente* por ser identificador de éste, y a través de este atributo mantiene referencia con la tabla *Constituyente*.
- El atributo *cantidad* es tomado del atributo del tipo de interrelación *(R-C)* que lleva el mismo nombre.

La clave principal de la tabla es la agregación de los atributos *nif_empresa*, *cod_residuo* y *cod_constituyente*. La tabla *Residuo_Constituyente* queda de la forma:

Residuo_Constituyente *(nif_empresa, cod_residuo, cod_constituyente, cantidad)*

7.3.5. Tabla *EmpresaTransportista*

La tabla *EmpresaTransportista* se forma a partir del tipo de entidad que lleva el mismo nombre, y de éste toma todos sus atributos que no mantienen referencia alguna con atributos de otras tablas (regla *RTECAR-1*). El identificador de esta tabla es el atributo *nif_emptransporte*.

EmpresaTransportista *(nif_emptransporte, nombre_emptransporte, ciudad_emptransporte, od_emptransporte)*

7.3.6. Tabla *Destino*

La tabla *Destino* se forma a partir del tipo de entidad *Destino*, y de él toma los atributos *cod_destino*, *nombre_destino*, *ciudad_destino* y *od_destino* (regla *RTECAR-1*). Ninguno de los atributos antes mencionados mantiene referencia alguna con los atributos de otras tablas siendo la clave de esta relación el atributo *cod_destino*.

Destino *(cod_destino, nombre_destino, ciudad_destino, od_destino)*

7.3.7. Tabla *Traslado*

La tabla *Traslado* se forma a partir del tipo de entidad *Traslado*, incorporando todos sus atributos, además almacena los atributos *fecha_llegada* y *tratamiento* del tipo de interrelación *(D-T)* que pasan a la tabla *Traslado* por participar ésta con cardinalidad máxima muchos en la interrelación (ver figura 7.1).

Los atributos *nif_empresa* y *cod_residuo* mantienen una referencia con el identificador de la tabla *Residuo*, mientras que el atributo *cod_destino* mantiene una referencia con el identificador de la tabla *Destino*. El identificador de la tabla *Traslado* es la agregación de los atributos *nif_empresa*, *cod_residuo*, *cod_destino* y *fecha_envio*. La tabla *Traslado* queda de la forma siguiente:

Traslado *(**nif_empresa, cod_residuo**, fecha_envio, **cod_destino**, envase, fecha_llegada, tratamiento, cantidad_traslado, od_traslado)*

7.3.8. Tabla *Traslado_EmpresaTransportista*

Esta tabla se forma por la existencia de la interrelación *(T-ET)* que existe entre los tipos de entidad *Traslado* y *EmpresaTransportista*. Esta tabla incorpora los atributos *nif_empresa, cod_residuo, cod_destino, fecha_envio, nif_emptransporte, kms, coste y tipo_transporte* (regla *RTECAR-4*):

- Los atributos *nif_empresa, cod_residuo, cod_destino, fecha_envio* y *cod_destino* son derivados del tipo de entidad *Traslado* por ser los identificadores de éste, y a través de ellos mantiene referencia con la tabla *Traslado*.

- El atributo *nif_emptransporte* es derivado del tipo de entidad *EmpresaTransportista* por ser el identificador en éste, y a través de este atributo mantiene referencia con la tabla del mismo nombre.
- Los atributos *tipo_transporte, kms* y *coste* son derivados de los atributos del tipo de la interrelación *(T-ET)* que llevan el mismo nombre.

La clave principal de la tabla es la agregación de los atributos *nif_empresa, cod_residuo, cod_destino, fecha_envio* y *nif_emptransporte*.

Traslado_EmpresaTransportista *(**nif_empresa, cod_residuo, fecha_envio, cod_destino, nif_emptransporte**, tipo_transporte, kms, coste)*

7.4. NORMALIZACIÓN DEL MODELO

Tabla *EmpresaProductora*: esta tabla se encuentra en *FNBC*, ya que todos los atributos son atómicos (*FN1*), y las únicas dependencias existentes son entre los atributos no primos y los primos.

nif_empresa → *nombre_empresa, ciudad_empresa, actividad, od_empresa*

Tabla *Residuo*: esta tabla también está en *FNBC* puesto que el único determinante funcional existente es el identificador de la tabla, y las dependencias con el resto de atributos son completas.

nif_empresa, cod_residuo → *toxicidad, cantidad_residuo, od_residuo*

Tabla *Constituyente*: esta tabla, al igual que la anterior, se encuentra en *FNBC*.

cod_constituyente → *nombre_constituyente, od_constituyente*

Tabla *Residuo_Constituyente*: esta tabla, al igual que las anteriores, se encuentra en *FNBC*.

nif_empresa, cod_residuo, cod_constituyente → *cantidad*

Tabla *EmpresaTransportista*: en esta tabla los atributos son atómicos, por lo tanto la tabla está en *FN1*, además no hay dependencias entre los atributos no primos, no existiendo dependencias funcionales transitivas y, al no existir más que un determinante funcional, la tabla está en *FNBC*.

nif_emptransporte → *nombre_emptransporte, ciudad_emptransporte, od_emptransporte*

Tabla *Traslado_EmpresaTransportista*: esta tabla, al igual que las anteriores, se encuentra en *FNBC*.

nif_empresa, cod_residuo, fecha_envio, cod_destino, nif_emptransporte → *tipo_transporte, kms, coste*

Tabla *Destino*: esta tabla también está en *FNBC*.

cod_destino → *nombre_destino, ciudad_destino, od_destino*

Tabla *Traslado*: los atributos de esta tabla son atómicos, por lo que la tabla se encuentra en *FN1*, no existen dependencias funcionales no completas entre atributos no primos y la clave de la tabla, *FN2*, satisface la *FN3* al no existir dependencias funcionales transitivas, y también satisface la *FNBC*.

nif_empresa, cod_residuo, fecha_envio, cod_destino → *envase, fecha_llegada, tratamiento, od_traslado, cantidad_traslado*

7.5. CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

Una vez que el esquema está normalizado, el modelo relacional propuesto queda de la forma siguiente:

BASE DE DATOS DE LOS RESIDUOS TÓXICOS

EmpresaProductora	*(nif_empresa, nombre_empresa, ciudad_empresa, actividad, od_empresa)*
Residuo	**(nif_empresa,** *cod_residuo,* *toxicidad, cantidad_residuo, od_residuo)*
Constituyente	*(cod_constituyente, nombre_constituyente, od_constituyente)*
Residuo_Constituyente	**(nif_empresa, cod_residuo, cod_constituyente,** *cantidad)*
EmpresaTransportista	*(nif_emptransporte, nombre_emptransporte, ciudad_emptrasnporte, od_emptransporte)*
Traslado	**(nif_empresa, cod_residuo,** *fecha_envio,* **cod_destino,** *envase, fecha_llegada, tratamiento, cantidad_traslado, od_traslado)*
Destino	*(cod_destino, nombre_destino, ciudad_destino, od_destino)*
Traslado_ EmpresaTransportista	**(nif_empresa, cod_residuo, fecha_envio, cod_destino, nif_emptransporte,** *tipo_transporte, kms, coste)*

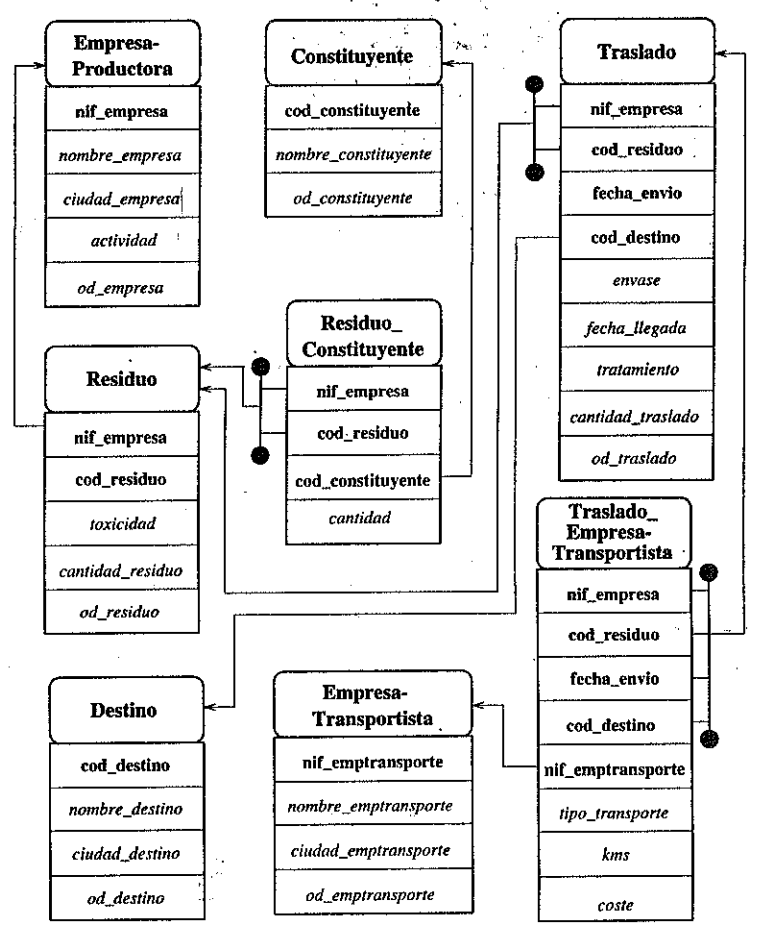


Figura 7.2 Diagrama relacional de la base de datos del problema de los residuos

7.5.1. Definición sintáctica de las tablas

```
   /* Inicialmente, se borran las tablas */
DROP TABLE Traslado_EmpresaTransportista;
DROP TABLE Traslado;
DROP TABLE Destino;
DROP TABLE EmpresaTransportista;
DROP TABLE Residuo_Constituyente;
DROP TABLE Constituyente;
DROP TABLE Residuo;
```

```
DROP TABLE EmpresaProductora;

   /* Se crean las tablas del esquema propuesto */
CREATE TABLE EmpresaProductora (
   nif_empresa VARCHAR2(12) NOT NULL,
   nombre_empresa VARCHAR2(40) NOT NULL,
   ciudad_empresa VARCHAR2(20),
   actividad VARCHAR2(80),
   od_empresa LONG,
     CONSTRAINT pk_emp
       PRIMARY KEY (nif_empresa),
     CONSTRAINT ck_emp
       CHECK (nif_empresa = UPPER(nif_empresa)) );
CREATE TABLE Residuo (
   nif_empresa VARCHAR2(12) NOT NULL,
   cod_residuo NUMBER(6,3) NOT NULL,
   toxicidad NUMBER(3),
   cantidad_residuo NUMBER(6),
   od_residuo LONG,
   CONSTRAINT pk_res
     PRIMARY KEY (nif_empresa, cod_residuo),
   CONSTRAINT fk_res_emp
     FOREIGN KEY (nif_empresa)
     REFERENCES EmpresaProductora(nif_empresa)
     ON DELETE CASCADE,
   CONSTRAINT ck_res
     CHECK (cod_residuo > = 0),
   CONSTRAINT ck_ctd
     CHECK (cantidad_residuo > 0) );
CREATE TABLE Constituyente (
   cod_constituyente NUMBER(3) NOT NULL,
   nombre_constituyente VARCHAR2(20) NOT NULL,
   od_constituyente LONG,
   CONSTRAINT pk_con
     PRIMARY KEY (cod_constituyente),
   CONSTRAINT ck_cod
     CHECK (cod_constituyente > = 0) );
CREATE TABLE Residuo_Constituyente (
   nif_empresa VARCHAR2(12) NOT NULL,
   cod_residuo NUMBER(6,3) NOT NULL,
   cod_constituyente NUMBER(3) NOT NULL,
   cantidad NUMBER(3),
   CONSTRAINT pk_rec
     PRIMARY KEY (nif_empresa, cod_residuo, cod_constituyente),
   CONSTRAINT fk_rec_res
     FOREIGN KEY (nif_empresa,cod_residuo)
     REFERENCES Residuo(nif_empresa, cod_residuo)
     ON DELETE CASCADE,
```



```
   CONSTRAINT fk_rec_con
     FOREIGN KEY (cod_constituyente)
     REFERENCES Constituyente(cod_constituyente)
     ON DELETE CASCADE,
   CONSTRAINT ck_cant
     CHECK (cantidad > 0) );
CREATE TABLE EmpresaTransportista (
   nif_emptransporte VARCHAR2(12) NOT NULL,
   nombre_emptransporte VARCHAR2(40) NOT NULL,
   ciudad_emptransporte VARCHAR2(30),
   od_emptransporte LONG,
   CONSTRAINT pk_tra
     PRIMARY KEY (nif_emptransporte),
   CONSTRAINT ck_tra
     CHECK (nif_emptransporte = UPPER(nif_emptransporte)) );
CREATE TABLE Destino (
   cod_destino VARCHAR2(12) NOT NULL,
   nombre_destino VARCHAR2(20) NOT NULL,
   ciudad_destino VARCHAR2(15),
   od_destino LONG,
   CONSTRAINT pk_des
     PRIMARY KEY (cod_destino),
   CONSTRAINT ck_des
     CHECK (cod_destino = UPPER(cod_destino)) );
CREATE TABLE Traslado (
   nif_empresa VARCHAR2(12) NOT NULL,
   cod_residuo NUMBER(6,3) NOT NULL,
   fecha_envio DATE NOT NULL,
   cod_destino VARCHAR2(12) NOT NULL,
   envase VARCHAR2(10),
   fecha_llegada DATE,
   tratamiento VARCHAR2(120),
   cantidad_traslado NUMBER(6),
   od_traslado LONG,
   CONSTRAINT pk_trl
     PRIMARY KEY (nif_empresa, cod_residuo, fecha_envio,
                  cod_destino),
   CONSTRAINT fk_trl_res
     FOREIGN KEY (nif_empresa, cod_residuo)
     REFERENCES Residuo(nif_empresa, cod_residuo),
   CONSTRAINT fk_trl_des
     FOREIGN KEY (cod_destino)
     REFERENCES Destino(cod_destino),
   CONSTRAINT ck_fecha_llegada
     CHECK (fecha_llegada > = fecha_envio),
   CONSTRAINT ck_can
     CHECK (cantidad_traslado > 0) );
```

```
CREATE TABLE Traslado_EmpresaTransportista (
   nif_empresa VARCHAR2(12) NOT NULL,
   cod_residuo NUMBER(6,3) NOT NULL,
   fecha_envio DATE NOT NULL,
   cod_destino VARCHAR2(12) NOT NULL,
   nif_emptransporte VARCHAR2(12) NOT NULL,
   tipo_transporte VARCHAR2(15),
   kms NUMBER(4),
   coste NUMBER(5),
   CONSTRAINT pk_tet
     PRIMARY KEY (nif_empresa, cod_residuo,
                  fecha_envio, cod_destino, nif_emptransporte),
   CONSTRAINT fk_tet_trl
     FOREIGN KEY (nif_empresa, cod_residuo, fecha_envio,
                  cod_destino)
     REFERENCES Traslado(nif_empresa, cod_residuo,
                fecha_envio, cod_destino)
     ON DELETE CASCADE,
   CONSTRAINT fk_tet_tra
     FOREIGN KEY (nif_emptransporte)
     REFERENCES EmpresaTransportista(nif_emptransporte),
   CONSTRAINT ck_cos
     CHECK (coste > = 0),
   CONSTRAINT ck_kms
     CHECK (kms > 0) );
```

7.5.2. Manipulación de la Base de Datos

Cap07ej01.sql

Obtener el nombre de las empresas que generan residuos con el constituyente de código 118.

De la tabla Residuo_Constituyente se obtiene el nif_empresa de todas las empresas entre cuyos residuos se encuentra alguno compuesto, al menos, por el constituyente que tiene por código 118. La información generada por esta consulta se utiliza para obtener de la tabla EmpresaProductora el nombre_empresa de todas esas empresas.

```
SELECT nif_empresa, nombre_empresa
   FROM EmpresaProductora
   WHERE nif_empresa IN
     (SELECT DISTINCT nif_empresa
       FROM Residuo_Constituyente
       WHERE cod_constituyente = 118 )
   ORDER BY nombre_empresa;
```

Resultado

```
NIF_EMPRESA  NOMBRE_EMPRESA
------------ ---------------
C-34534522-R Aceioro
A-17655551-Q Fermezquita
```


Cap07ej02.sql

Obtener el nombre de todas las empresas transportistas que han realizado un traslado desde empresas ubicadas en Madrid.

De la tabla EmpresaProductora se obtiene el nif_empresa de todas las empresas ubicadas en Madrid. Esta información se utiliza para obtener el NIF de las empresas de transporte que han realizado algún traslado para estas empresas. El resultado de esta subconsulta se emplea para obtener los nombres de esas empresas de transporte, que aparecen ordenados alfabéticamente.

```
SELECT nif_emptransporte, nombre_emptransporte
   FROM EmpresaTransportista
   WHERE nif_emptransporte IN
      (SELECT DISTINCT nif_emptransporte
       FROM Traslado_EmpresaTransportista
       WHERE nif_empresa IN
          (SELECT nif_empresa
            FROM EmpresaProductora
            WHERE ciudad_empresa = 'Madrid' ))
   ORDER BY nombre_emptransporte;
```

Resultado

```
NIF_EMPTRANS NOMBRE_EMPTRANSPORTE
------------ -------------------------
F-98987667-R AceSur
A-98987067-V HuelResi
```

Cap07ej03.sql

Obtener el importe de todos los traslados que ha hecho la empresa de NIF 'F-98987667-R' a la productora 'A-98989998-Q' con destino 'DESTIO-N15'.

La función de grupos SUM devuelve como resultado la suma de valores del campo bajo el que actúa. Así, el importe total de todos los traslados es el valor resultante de sumar el campo coste de todas las tuplas de la tabla Traslado_EmpresaTransportista que cumplan la condición de que el NIF de la empresa generadora de residuos sea 'F-98987667-R', el NIF de la empresa transportista sea 'A-98989998-Q', y que el código del destino sea 'DESTIO-N15'.

```
SELECT SUM(coste) "Importe de los traslados"
   FROM Traslado_EmpresaTransportista
   WHERE nif_empresa = 'A-98989998-Q' AND nif_emptransporte =
         'F-98987667-R' AND cod_destino = 'DESTIO-N15';
```

Resultado

```
Importe de los traslados
------------------------
                  165000
```

Cap07ej04.sql

Obtener los nombres de las ciudades destino de todos los traslados que ha realizado la empresa transportista 'A-97654567-S' de una distancia superior a los 300 kms.

De la tabla Traslado_EmpresaTransportista se obtiene el cod_destino de aquellas tuplas que cumplen la condición que en el traslado participó la empresa de NIF 'A-97654567-S' y la distancia recorrida fue superior a los 300 kilómetros. Gracias al resultado de esta subconsulta se obtiene de la tabla Destino el nombre y la ciudad

asociada a dichos destinos. El resultado aparece bajo las etiquetas 'Destino' y 'Ciudad de destino'.

```
SELECT nombre_destino "Destino", ciudad_destino
    "Ciudad de destino"
  FROM Destino
  WHERE cod_destino IN
    (SELECT DISTINCT cod_destino
     FROM Traslado_EmpresaTransportista
     WHERE nif_emptransporte = 'A-97654567-S' AND kms > 300);
```

Resultado

```
Destino              Ciudad de destino
-------------------- --------------------
MASUR                Madrid
```

Cap07ej05.sql

Obtener el nombre de todas las ciudades a las que van a parar residuos que incluyen el constituyente 'Renio'.

De la tabla Residuo_Constituyente se obtienen los campos nif_empresa, cod_residuo (es decir, la clave del residuo) de aquellas tuplas que cumplen la condición de que el nombre del constituyente es 'Renio'. El resultado de esta subconsulta se utiliza para obtener de la tabla Traslado el código de todos los destinos de los traslados realizados en los que el constituyente 'Renio' haya sido transportado. Esta información se utiliza para obtener de la tabla Destino el nombre de las ciudades en que están ubicados dichos destinos. El resultado aparece bajo la etiqueta 'Ciudad de destino'.

```
SELECT DISTINCT ciudad_destino "Ciudad de destino"
  FROM Destino
  WHERE cod_destino IN
    (SELECT DISTINCT cod_destino
     FROM Traslado
     WHERE (nif_empresa, cod_residuo) IN
       (SELECT DISTINCT nif_empresa, cod_residuo
          FROM Residuo_Constituyente R_C, Constituyente C
          WHERE C.nombre_constituyente = 'Renio' AND
                C.cod_constituyente=R_C.cod_constituyente))
  ORDER BY ciudad_destino;
```

Resultado

```
Ciudad de destino
--------------------
Barcelona
Córdoba
Madrid
Valencia
```

Cap07ej06.sql

Obtener todas las empresas generadoras de residuos y las transportistas ubicadas en la misma ciudad.

De la tabla EmpresaProductora se obtiene el campo nif_empresa y de la tabla EmpresaTransportista se obtienen los campos nif_emptransporte y ciudad_emptransporte, y el resultado de ambas consultas se combina mediante una reunión bajo la condición de que la ciudad en que se encuentran ambas empresas sea la misma. El resultado aparece bajo las etiquetas 'Empresa Productora', 'Empresa Transportista' y 'Ciudad' y, gracias a la reunión realizada, emparejadas las empresas que comparten ciudad.


```
SELECT nif_empresa "Empresa Productora",
       nif_emptransporte "Empresa Transportista",
       ciudad_emptransporte "Ciudad"
  FROM EmpresaProductora, EmpresaTransportista
  WHERE ciudad_empresa = ciudad_emptransporte;
```

Resultado

```
Empresa Productora   Empresa Transportista  Ciudad
-------------------- -------------------- --------------
A-17655551-Q         F-98987667-R         Córdoba
A-98989998-Q         A-98987067-V         Huelva
A-11111111-R         A-97654567-S         Madrid
A-11111111-R         A-98985367-V         Madrid
```

Cap07ej07.sql

Obtener la empresa transportista de Madrid que haya participado en un traslado a Valencia con el menor coste.

Se realiza una reunión natural de las tablas implicadas: Destino, Traslado_EmpresaTransportista y EmpresaTransportista sobre los atributos comunes, aplicando la restricciones impuestas sobre las ciudades y proyectando sobre el coste mínimo haciendo uso de la función MIN. Para simplificar la sentencia a cada una de las tablas se le aplica un alias.

```
SELECT TET.nif_emptransporte "Empresa_Transportista",
       TET.coste "Importe"
  FROM Traslado_EmpresaTransportista TET, Destino D,
       EmpresaTransportista ET
  WHERE D.cod_destino = TET.cod_destino AND
       TET.nif_emptransporte = ET.nif_emptransporte AND
       D.ciudad_destino = 'Valencia' AND
       ET.ciudad_emptransporte = 'Madrid' AND TET.coste=
      (SELECT MIN(TET.coste)
       FROM Traslado_EmpresaTransportista TET, Destino D,
            EmpresaTransportista ET
       WHERE D.cod_destino = TET.cod_destino AND
            TET.nif_emptransporte = ET.nif_emptransporte AND
            D.ciudad_destino = 'Valencia' AND
            ET.ciudad_emptransporte = 'Madrid');
```

Resultado

```
Empresa Transportista     Importe
--------------------- ----------
A-98985367-V               20000
```

Cap07ej08.sql

Introducir una tupla en la tabla Traslado_EmpresaTransportista.

El interés de esta operación se encuentra en introducir el valor del campo fecha_envio, que es de tipo DATE. Al realizar una operación del tipo INSERT, el valor que se le asignará a este campo será una cadena de caracteres que deberá ser convertida al tipo de dato DATE. Esta labor la desempeña la función TO_DATE.

```
INSERT INTO Traslado_EmpresaTransportista
```

```
      (nif_empresa, cod_residuo, fecha_envio, cod_destino,
      nif_emptransporte, tipo_transporte, kms, coste)
   VALUES ('A-98989998-Q', 2.040, TO_DATE('21/12/1991',
          'DD/MM/YYYY'), 'DESTIO-N05', 'F-98987667-R',
          'A',700,35000);
```

Resultado

```
1 fila creada.
```

Cap07ej09.sql

Borrar todos los traslados que llegaron entre 17/04/1994 el 07/05/1996, sin incluir ambas fechas.

Se borran todas las tuplas de la tabla Traslado que cumplan la condición de que el campo fecha_llegada sea mayor que la cadena 17/04/1994 y menor a la cadena de caracteres 07/05/1996, ambas han sido convertidas al tipo de dato DATE.

```
DELETE Traslado
   WHERE fecha_llegada < TO_DATE('07/05/1996','DD/MM/YYYY')
         AND fecha_llegada >TO_DATE('17/04/1994','DD/MM/YYYY');
```

Resultado

```
12 filas borradas.
```

Cap07ej10.sql

Impedid que se den de alta más destinos en Sevilla.

Para lograrlo, es necesario modificar la tabla Destino añadiendo una nueva restricción. Es necesario emplear la función ALTER TABLE que irá acompañada de la cláusula ADD para indicar que se va a añadir algo a su estructura (en este caso un CONSTRAINT sobre el campo ciudad_destino). Es necesario indicar que si en el momento de intentar añadir a la estructura de la tabla el CONSTRAINT indicado existe alguna tupla en la que el campo ciudad_destino tiene el valor 'Sevilla', el sistema impedirá que se efectúe la operación hasta que dichas tuplas sean eliminadas.

```
ALTER TABLE Destino
   ADD CONSTRAINT ck_cid
    CHECK (ciudad_destino != 'Sevilla');
```

Resultado

```
Tabla modificada.
```

Cap07ejP1.sql

Crear un procedimiento PL/SQL que modifique el NIF de una empresa de transporte introducida por el usuario como argumento del procedimiento.

El procedimiento recibirá como primer argumento el nif de la empresa que hay que modificar, y como segundo argumento el nuevo nif. Inicialmente, con el mandato SELECT...INTO se almacena en la variable inf_empresa toda la información de la empresa. Esa información es insertada en la tabla EmpresaTransportista con el nuevo nif.

```
CREATE OR REPLACE PROCEDURE act_niv_emp_trans
   /* Se definen las variables de la interfaz */
   (p_nif_antiguo VARCHAR2,
   p_nif_nuevo VARCHAR2)
AS
   /* Se definen las variables internas de trabajo y el cursor
   para recorrer la tabla */
```



```
   inf_empresa empresatransportista%ROWTYPE;
-- Se define una variable que almacena la cadena de error
   error VARCHAR2(100);
BEGIN
   SELECT *
    INTO inf_empresa
    FROM EmpresaTransportista
    WHERE nif_emptransporte=p_nif_antiguo;
   /* Debido a la existencia del CONSTRAINT definido en la
   tabla, el orden de los procesos es: primero insertar la
   nueva tupla, después modificar la referencia y, por último,
   borrar la tupla antigua */
   INSERT INTO EmpresaTransportista (nif_emptransporte,
               nombre_emptransporte, ciudad_emptransporte,
               od_emptransporte)
    VALUES (p_nif_nuevo,inf_empresa.nombre_emptransporte,
           inf_empresa.ciudad_emptransporte,
           inf_empresa.od_emptransporte);
   UPDATE Traslado_EmpresaTransportista
    SET nif_emptransporte=p_nif_nuevo
     WHERE nif_emptransporte=p_nif_antiguo;
   DELETE FROM EmpresaTransportista
    WHERE nif_emptransporte=p_nif_antiguo;
COMMIT;
   /* Se controlan los posibles errores de operación */
EXCEPTION
   WHEN OTHERS THEN
    IF SQLCODE=100 THEN
    /* El select no ha devuelto datos, no existe el nif */
     error:='NO EXISTE UNA EMPRESATRANSPORTISTA CON ESE NIF';
    ELSIF SQLCODE=-1 THEN
     error:='YA EXISTE UNA EMPRESATRANSPORTISTA CON ESE NIF';
     ELSE
      error:=SQLERRM;
    END IF;
    DBMS_OUTPUT.PUT_LINE(error);
   ROLLBACK;
END;
/
```

Pruebas de ejecución

```
-- NO existe el nif
    EXEC act_niv_emp_trans('783638','82872');
-- Se introduce un nuevo nif que ya existe
    EXEC act_niv_emp_trans('F-98987667-R','A-87684567-B');
-- Una orden correcta
    EXEC act_niv_emp_trans('F-98987667-R','A-08080808-B');
```

CAPÍTULO 8

DIETAS GANADERAS

8.1. ENUNCIADO DEL PROBLEMA

En una determinada granja se desea mantener la información correspondiente a la alimentación que se suministra a los animales que son explotados en la misma.

El control y seguimiento de la alimentación que se proporciona a cada uno de los animales de la granja tiene como objetivo el estudio y análisis de los resultados de la misma sobre la producción y beneficio que se obtiene de los animales. En este sentido, cada animal sigue una dieta alimenticia en base a una serie de criterios determinados por los veterinarios y gestores de la granja (necesidades de nutrientes dependiendo del tipo de animal, edad, etc., disponibilidad de los alimentos, coste, etc.).

La dieta seguida por cada uno de los animales de la granja puede ser variada a lo largo de su vida, siendo de interés para los gestores de la misma el conocimiento de esta información.

Cada dieta está compuesta por una serie de alimentos que son ingeridos por los animales en diferentes tomas a lo largo del día. En cada toma, cada uno de los animales ingiere unas cantidades determinadas en la dieta de uno o varios alimentos.

Los alimentos y las cantidades ingeridas de los mismos por cada uno de los animales en su dieta son determinados basándose en criterios veterinarios y económicos.

Se consideran además los siguientes supuestos semánticos en el problema:

SUPUESTO 1: Una dieta se establece para un animal y no para todos los animales de la misma especie. Si bien, una misma dieta puede ser seguida al mismo tiempo por varios animales de la granja.

SUPUESTO 2: Cada animal sigue una y sólo una dieta a la vez.

SUPUESTO 3: Se desea guardar información histórica sobre todas las dietas que han seguido los animales.

SUPUESTO 4: Cada uno de los animales tiene un código asociado que puede ir impreso bien en la oreja o bien en el lomo.

SUPUESTO 5: Una dieta no existe en el sistema de información hasta que no se establece como mínimo para un animal.

SUPUESTO 6: Se desea mantener información de los nutrientes sean o no necesitados por los animales de la granja.

SUPUESTO 7: Se entiende por una dieta al conjunto de alimentos que recibe el animal a lo largo del día, los cuales son ingeridos en diferentes tomas. En este sentido no se considerará, por simplicidad, regímenes alimenticios distribuidos en un tiempo superior a veinticuatro horas.

SUPUESTO 8: Las dietas están constituidas por un conjunto de alimentos o productos alimenticios que contienen los nutrientes que necesitan los animales en su alimentación.

8.2. MODELO CONCEPTUAL

Se trata de representar la información correspondiente a las pautas en la alimentación que siguen los animales en una determinada granja.

8.2.1. Análisis de los tipos de entidad

Considerando el enunciado del problema, podemos extraer los siguientes tipos de entidad, como se muestra en la figura 8.4:

Tipo de entidad *Animal*: el cual representa al objeto del mundo real *"animal que es criado en la granja"*. Para este tipo de entidad se consideran los atributos *cod_animal*, *tipo_animal*, *peso*, *ano_nacimiento*, *utilidad_animal*, *produccion_animal* y *od_animal* (ENUNCIADO).

El atributo *tipo_animal* representa la especie a la que pertenece el animal, el atributo *utilidad_animal* representa el fin por el que el animal se está criando en la granja (por ejemplo, para producir leche, poner huevos, etc.), *produccion_animal* representa la cantidad de cualquier producto que produzca el animal (por ejemplo, litros de leche, número de huevos/día, etc.) y el atributo *od_animal* representa cualquier otra información que se desee considerar en el sistema de información.

El identificador del tipo de entidad es el atributo *cod_animal*, ya que se ha considerado que todos los animales de la granja tienen un código para diferenciarlos, por lo que no hay dos animales con el mismo código (SUPUESTO 4).

Tipo de entidad *Nutriente*: el cual representa al objeto del mundo real ***"sustancia orgánica o inorgánica que es necesaria para la correcta crianza de los animales"***. Para este tipo de entidad se consideran los atributos *nombre_nutriente*, *magnitud_nutriente*, *estado* y *od_nutriente* (SUPUESTO 8).

El atributo *nombre_nutriente* se considera identificador de este tipo de entidad, ya que no existen dos nutrientes con el mismo nombre. El atributo *magnitud_nutriente* representa la unidad en la que se mide el nutriente, el atributo *estado* representa el estado físico en el que se encuentra este nutriente (puede ser sólido, líquido, etc.) y *od_nutriente* representa cualquier otra información de interés (SUPUESTO 6).

Tipo de entidad *Alimento*: el cual representa al objeto del mundo real ***"sustancia nutritiva para el organismo del animal y que puede ser ingerida por éste en su dieta"***. Los atributos de este tipo de entidad son: *nombre_alimento*, *tipo_alimento*, *magnitud_alimento*, *coste_alimento* y *od_alimento* (SUPUESTO 7).

El atributo *nombre_alimento* se considera identificador del tipo de entidad *Alimento*, ya que representa sin ambigüedad cualquier entidad de este tipo. El atributo *tipo_alimento* representa la clase a la que pertenece un alimento determinado (fruta, carne, etc.), el atributo *magnitud_alimento* representa la unidad en la que se mide el alimento (litros, kilos, etc.) y el atributo *coste_alimento* representa el precio de coste de ese alimento por unidad de magnitud de medida del mismo. El atributo *od_alimento* representa cualquier información añadida que se desee considerar (SUPUESTO 8).

Tipo de entidad *Toma*: el cual representa al objeto del mundo real *"momento del día en el cual un animal recibe alimentos"*. Este tipo de entidad considera los atributos *cod_toma*, *nombre_toma*, *hora_inicio*, *hora_fin* y *od_toma*. El atributo *nombre_toma* representa el nombre que recibe la comida que toma un cierto animal a una determinada hora (por ejemplo, desayuno, cena, merienda, etc.), los atributos *hora_inicio* y *hora_fin* representan la hora de comienzo y la hora de finalización de la comida, respectivamente (SUPUESTO 7). El identificador de este tipo de entidad es el atributo *cod_toma*, ya que representa sin ambigüedad a una entidad de este tipo.

Tipo de entidad *Dieta*: el cual representa el objeto del mundo real ***"combinación de alimentos que se hace seguir cada día a los animales para que den el máximo rendimiento"***. Este tipo de entidad se puede considerar como un tipo de entidad

débil por existencia respecto a los tipos de entidad *Animal* y *Alimento*, ya que no puede existir una dieta si no existe un animal que la siga, así como alimentos que la compongan (SUPUESTO 1 y otros).

Para este tipo de entidad se consideran los atributos *cod_dieta*, *finalidad* y *od_dieta*. El atributo *finalidad* representa el objetivo para el cual se establece la dieta a un animal, y el atributo *od_dieta* representa cualquier otro dato que se desee considerar. El identificador de este tipo de entidad es el atributo *cod_dieta*, ya que se considera que no existe más de una dieta con el mismo código.

8.2.2. Análisis de los tipos de interrelación

Los tipos de entidad antes descritos se encuentran relacionados de la siguiente forma:

Tipo de interrelación *Animal/Nutriente* **(A-N)**: el cual relaciona los tipos de entidad *Animal* y *Nutriente*. Se considera que un determinado animal necesita un nutriente como mínimo o, por el contrario, puede ser que necesite varios, participando por tanto el tipo de entidad *Nutriente* con cardinalidades *(1,n)*. Por otro lado, puede que un determinado nutriente no sea necesitado por ningún animal, o bien puede ser necesitado por varios, por lo que el tipo de entidad *Animal* participa en la interrelación con cardinalidades *(0,n)* (SUPUESTO 6).

Este tipo de interrelación tiene un atributo llamado *cantidad_necesitada* (en la misma magnitud que el nutriente), que representa la cantidad de un nutriente que necesita un animal (ver figura 8.4).

Tipo de interrelación *Nutriente/Alimento* **(N-A)**: el cual relaciona los tipos de entidad *Nutriente* y *Alimento* en una relación muchos a muchos. Se considera que un alimento contiene varios nutrientes o uno como mínimo, participando por tanto el tipo de entidad *Nutriente* con cardinalidades *(1,n)*. Un determinado nutriente tiene existencia propia independientemente de que se encuentre incluido en un alimento, por lo que el tipo de entidad *Alimento* participa en la interrelación con cardinalidades *(0,n)*. Este tipo de interrelación tiene un atributo llamado *cantidad_contenida* (determinada en la magnitud del nutriente), que representa la cantidad de nutriente contenida por unidad de alimento (SUPUESTO 8).

Tipo de interrelación *Alimento/Dieta/Toma* **(A-D-T)**: el cual relaciona los tipos de entidad *Alimento*, *Dieta* y *Toma*. Este tipo de interrelación permite identificar dentro de una dieta qué alimentos son los que recibe un animal y en qué toma los recibe, en una relación de cardinalidad *N:N:N*. Se considera que un alimento puede no estar contenido en una dieta o por el contrario estar contenido en varias, por lo que el tipo de entidad *Dieta* participa en la interrelación con cardinalidades *(0,n)*. El número de alimentos que contiene una dieta son muchos, pero siempre como mínimo uno (si no, no habría dicta), participando por lo tanto el tipo de entidad *Alimento* con cardinalidades *(1,n)* (SUPUESTOS 7 y 8).

Un animal recibe alimentos en una dieta al menos una vez al día, y por el contrario, puede recibirlos varias veces al día, por lo tanto, el tipo de entidad *Toma* participa en la interrelación con cardinalidades *(1,n)*.

Este tipo de interrelación tiene un atributo llamado *cantidad_toma*, que representa la cantidad (determinada en la misma magnitud que alimento) de un determinado alimento que se incluye en una toma de una dieta que es seguida por los animales (ver figura 8.4).

Tipo de interrelación *Dieta/Animal* **(D-A)**: el cual relaciona los tipos de entidad *Dieta* y *Animal*, esta interrelación es del tipo *1:N*. Se considera que una dieta la puede seguir varios animales, pero no tiene existencia en el sistema de información hasta que no la sigue un animal como mínimo, por lo que el tipo de entidad *Animal* participa en la interrelación con cardinalidades *(1,n)*. Por otro lado se ha considerado que es una y sólo una la dieta que sigue un determinado animal en una fecha dada, por lo que el tipo de entidad *Dieta* participa en la interrelación con cardinalidades *(1,1)* (SUPUESTOS 1 y 2).

Debido a la restricción del problema por la cual se necesita mantener información histórica de todas las dietas que ha seguido un determinado animal, en este tipo de interrelación se debe considerar un atributo que represente la fecha en la cual un animal comienza una dieta, y este atributo ha de ser clave de la relación. Para poder representar la fecha como clave de la interrelación se proponen tres soluciones (SUPUESTO 3):

1. Añadir un atributo *fecha_inicio* al tipo de interrelación (*D-A*) en el esquema conceptual, y sombrearlo para que se considere como identificador, así como otro atributo llamado *od_resultado* que represente cualquier información que se desee considerar sobre el resultado de aplicar la dieta en el animal (véase la figura 8.1).

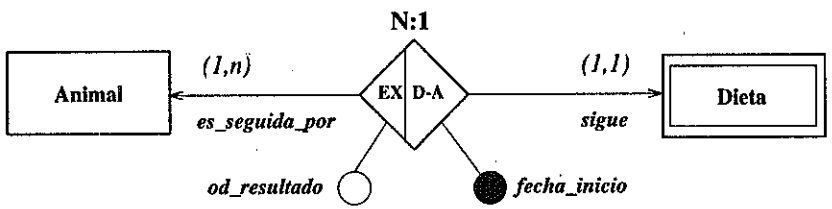


Figura 8.1 Primera solución propuesta (incorrecta)

Es usual la consideración de esta solución, si bien no es acertada, ya que en el modelo *EE-R* no es correcto considerar atributos identificadores (propios) de los tipos de interrelación. Un tipo de interrelación debe ser utilizado para relacionar conjuntos y aunque se puedan considerar propiedades (atributos) en el mismo éstos no pueden tener la propiedad de identificación.

2. Considerar *fecha_inicio* como parte de la clave del tipo de entidad *Dieta*, con lo que el identificador de este tipo de entidad sería la agregación de los atributos *cod_dieta* y *fecha_inicio*. El atributo *od_resultado* quedaría como atributo del tipo de interrelación (*D-A*) (véase la figura 8.2).

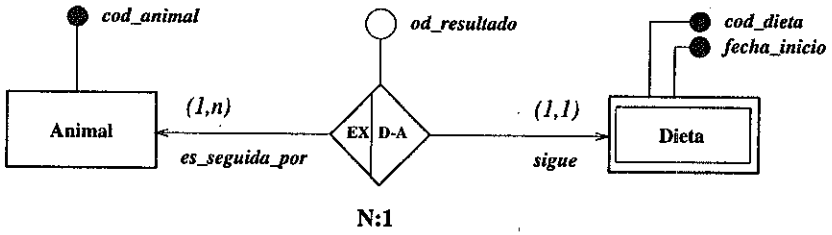


Figura 8.2 Segunda solución propuesta (incorrecta)

Esta solución no es adecuada, ya que la fecha de comienzo de una dieta no se puede considerar como un atributo del tipo de entidad *Dieta*, puesto que una misma dieta puede ser seguida por varios animales a la vez, y la fecha de inicio de la dieta puede ser distinta para cada animal.

3. Considerar un nuevo tipo de entidad llamado *FechaInicio*, esta solución es la que se ha considerado para el problema de las dietas ganaderas, ya que representa de forma adecuada el problema abordado[20] (véase la figura 8.3).

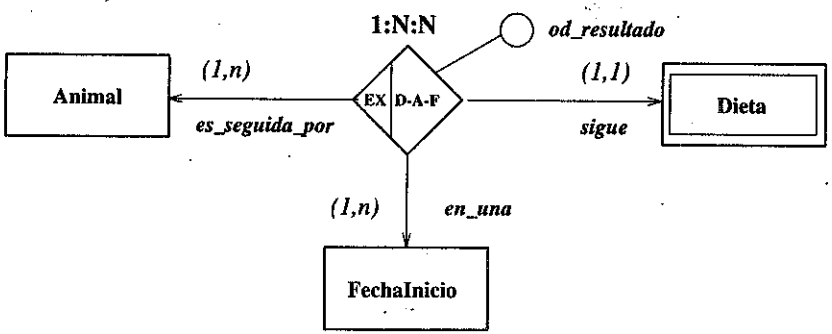


Figura 8.3 Tercera solución propuesta (correcta)

Considerando el nuevo tipo de entidad *FechaInicio* con un atributo identificador llamado *fecha_inicio*, el tipo de interrelación *(D-A)* se convierte en una relación entre tres tipos de entidad: *Dieta*, *Animal* y *FechaInicio*, llamada *(D-A-F)*, en el que interviene un atributo llamado

[20] Se trata de la solución más correcta y elimina cualquier ambigüedad que pudiera introducir la propuesta en el primer apartado (figura 8.1), puesto que representa claramente la implicación del tiempo a través de la participación del tipo de entidad *FechaInicio* en el tipo de interrelación.

od_resultado que representa cualquier información que se desee considerar sobre la aplicación de la dieta a un animal.

Este tipo de interrelación es del tipo *1:N:N*, donde el tipo de interrelación *FechaInicio* participa con cardinalidades *(1,n)*, ya que un animal sigue al menos una dieta, por lo que existe como mínimo una fecha de comienzo; por otro lado, son varias las dietas que puede seguir un animal históricamente.

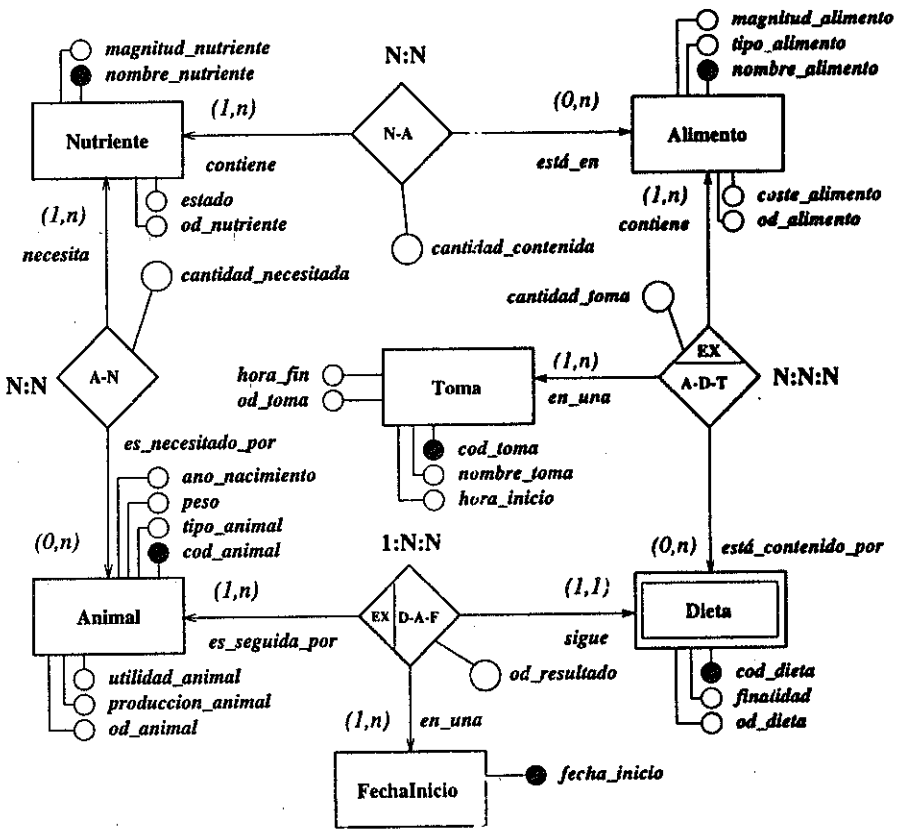


Figura 8.4 Esquema E-R del modelo conceptual de las dietas ganaderas

8.3. MODELO RELACIONAL

Una vez realizado el modelo conceptual del problema abordado se va a realizar el modelo relacional correspondiente (véase la figura 8.5).

8.3.1. Tabla *Animal*

La tabla *Animal* se forma a partir del tipo de entidad que lleva el mismo nombre, y de ella toma los atributos *cod_animal*, *tipo_animal*, *peso*, *ano_nacimiento*, *utilidad_animal*, *produccion_animal* y *od_animal* (regla *RTECAR-1*).

El identificador de esta tabla es el atributo *cod_animal*, no considerándose ningún atributo como clave alterna. La tabla animal queda de la forma siguiente:

Animal *(cod_animal, tipo_animal, peso, ano_nacimiento, utilidad_animal, produccion_animal, od_animal)*

8.3.2. Tabla *Nutriente*

La tabla *Nutriente* se forma a partir del tipo de entidad *Nutriente* y de éste toma los atributos *nombre_nutriente*, *magnitud_nutriente*, *estado* y *od_nutriente* (regla *RTECAR-1*).

Del conjunto de atributos de la tabla el atributo *nombre_nutriente* es el identificador de la misma, no considerándose ningún atributo como clave alterna, y ninguno de los atributos de la tabla *Nutriente* mantiene referencia alguna con atributos de las otras tablas. La tabla *Nutriente* queda de la forma:

Nutriente *(nombre_nutriente, magnitud_nutriente, estado od_nutriente)*

8.3.3. Tabla *Animal_Nutriente*

Esta tabla se forma a partir del tipo de *(A-N)* que existe entre los tipos de entidad *Animal* y *Nutriente* (ver figura 8.4). Esta tabla tiene los atributos *cod_animal*, *nombre_nutriente* y *cantidad_necesitada*, en los cuales (regla *RTECAR-4*):

- El atributo *cod_animal* es derivado del tipo de entidad *Animal* por ser el identificador de éste, y a través de este atributo mantiene referencia con la tabla *Animal*.
- El atributo *nombre_nutriente* es derivado del tipo de entidad *Nutriente* por ser su identificador, y a través de este atributo mantiene referencia con la tabla *Nutriente*.
- El atributo *cantidad_necesitada* es tomado del atributo del tipo de interrelación *(A-N)* que lleva el mismo nombre.

La clave principal de la tabla *Animal_Nutriente* es la agregación de los atributos *cod_animal* y *nombre_nutriente*.

Animal_Nutriente *(cod_animal, nombre_nutriente, cantidad_necesitada)*

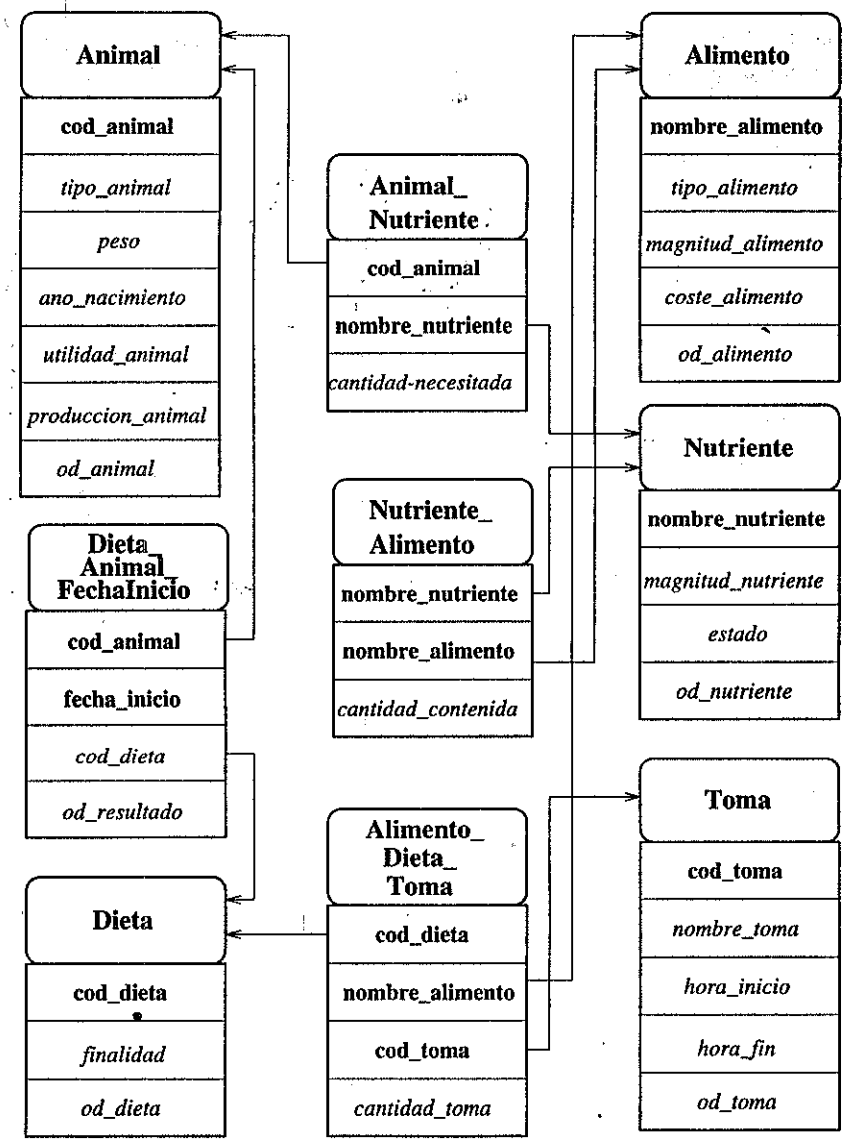


Figura 8.5 Diagrama relacional del problema de las dietas ganaderas

8.3.4. Tabla *Alimento*

La tabla *Alimento* se forma a partir del tipo de entidad que lleva el mismo nombre, y de él toma los atributos *nombre_alimento*, *tipo_alimento*, *magnitud_alimento*, *coste_alimento* y *od_alimento*. Ninguno de estos atributos mantiene referencia alguna con atributos de otras tablas (regla *RTECAR-1*). El

identificador de esta tabla es el atributo *nombre_alimento*. La tabla *Alimento* queda de la forma siguiente:

Alimento *(nombre_alimento, tipo_alimento, magnitud_alimento, coste_alimento, od_alimento)*

8.3.5. Tabla *Nutriente_Alimento*

La tabla *Nutriente_Alimento* se forma a partir del tipo de interrelación *(N-A)* que existe entre los tipos de entidad *Nutriente* y *Alimento* (ver figura 8.4) (regla *RTECAR-4*). Los atributos de esta tabla son: *nombre_nutriente*, *nombre_alimento* y *cantidad_contenida*, los cuales:

- El atributo *nombre_nutriente* es derivado del tipo de entidad *Nutriente*.
- El atributo *nombre_alimento* es derivado del tipo de entidad *Alimento*.
- El atributo *cantidad_contenida* es el atributo del tipo de interrelación *(N-A)* derivándose desde este tipo de interrelación.

La tabla *Nutriente_Alimento* mantiene referencia con las tablas *Nutriente* y *Alimento* a través de los atributos *nombre_nutriente* y *nombre_alimento*, respectivamente. La clave principal de esta tabla es la agregación de los atributos *nombre_nutriente* y *nombre_alimento*, por lo que la tabla queda de la forma:

Nutriente_Alimento *(**nombre_nutriente, nombre_alimento**, cantidad_contenida)*

8.3.6. Tabla *Toma*

La tabla *Toma* se forma a partir del tipo de entidad que lleva el mismo nombre, y de este tipo de entidad toma los atributos *cod_toma*, *nombre_toma*, *hora_inicio*, *hora_fin* y *od_toma* (regla *RTECAR-1*). El atributo identificador de esta tabla es el atributo *cod_toma*, no considerándose ningún otro atributo como clave alterna. Ninguno de los atributos de la tabla mantiene referencia con otras tablas.

Toma *(cod_toma, nombre_toma, hora_inicio, hora_fin, od_toma)*

8.3.7. Tabla *Dieta*

La tabla *Dieta* se forma a partir del tipo de entidad que lleva el mismo nombre, y de él toma los atributos *cod_dieta*, *finalidad* y *od_dieta*, ninguno de estos atributos mantiene referencia alguna con otras tablas (regla *RTECAR-1*). La clave principal de esta tabla es el atributo *cod_dieta*. La tabla *Dieta* queda de la forma:

Dieta *(cod_dieta, finalidad, od_dieta)*

8.3.8. Tabla *Alimento_Dieta_Toma*

La tabla *Alimento_Dieta_Toma* se forma a partir de la relación que existe entre los tipos de entidad *Alimento*, *Dieta* y *Toma* (ver figura 8.4). Esta tabla tiene como atributos: *nombre_alimento*, *cod_dieta*, *cod_toma* y *cantidad_toma* (regla *RTECAR-4*), debido a:

- El atributo *nombre_alimento* es derivado del tipo de entidad *Alimento* por ser el identificador de éste.
- El atributo *cod_dieta* es derivado del tipo de entidad *Dieta* por ser el identificador de éste.
- El atributo *cod_toma* es derivado del identificador del tipo de entidad *Toma*.
- El atributo *cantidad_toma* es tomado del atributo del tipo de interrelación *(A-D-T)* que lleva el mismo nombre.

A través del atributo *nombre_alimento* la tabla mantiene referencia con la tabla *Alimento*, y a través del atributo *cod_dieta* mantiene referencia con la tabla *Dieta*. Análogamente la tabla *Alimento_Dieta_Toma* mantiene referencia con la tabla *Toma* a través del atributo *cod_toma*. La clave de la tabla *Alimento_Dieta_Toma* es la agregación de los atributos *nombre_alimento*, *cod_dieta* y *cod_toma*.

Alimento_Dieta_Toma *(**cod_dieta, nombre_alimento, cod_toma**, cantidad_toma)*

8.3.9. Tabla *Dieta_Animal_FechaInicio*

Esta tabla se forma a partir del tipo de relación *(D-A-F)* que existe entre los tipos de entidad *Dieta*, *Animal* y *FechaInicio*, como se observa en la figura 8.5 (regla *RTECAR-4*).

Los atributos de esta tabla son: *cod_dieta*, *cod_animal*, *fecha_inicio* y *od_resultado*, y el identificador de esta tabla es la agregación de los atributos *cod_animal* y *fecha_inicio*. La tabla *Dieta_Animal_FechaInicio* queda de la forma:

Dieta_Animal_FechaInicio *(**cod_animal, fecha_inicio**, **cod_dieta**, od_resultado)*

Puede observar el lector que no hemos considerado en el modelo relacional ninguna tabla surgida a partir del tipo de entidad *FechaInicio*. Es evidente, este tipo de entidad representa al tiempo, en realidad representa un calendario el cual no es necesario y mucho menos conveniente almacenar en una base de datos.

Debe recordar el lector que cuando desee representar el cambio de un sistema a través del tiempo y para ello haga uso de algún tipo de entidad que represente al tiempo (como *FechaInicio*) estos tipos de entidades nunca se derivan a tabla, a no ser que incorporen, por las características del problema, algún atributo añadido además del atributo que representa el tiempo (el calendario).

8.4. NORMALIZACIÓN DEL MODELO

Como se muestra en los diagramas de dependencias adjuntos, todas las relaciones se encuentran normalizadas en *FNBC*, por lo que no es necesario realizar ninguna operación de normalización sobre las tablas definidas para el problema.

TABLA *ANIMAL*

cod_animal → *tipo_animal, peso, ano_nacimiento, utilidad_animal, produccion_animal, od_animal*

TABLA *NUTRIENTE*

nombre_nutriente → *magnitud_nutriente,estado, od_nutriente*

TABLA *ANIMAL_NUTRIENTE*

cod_animal, nombre_nutriente → *cantidad_necesitada*

TABLA *ALIMENTO*

nombre_alimento → *tipo_alimento, magnitud_alimento, coste_alimento, od_alimento*

TABLA *NUTRIENTE_ALIMENTO*

nombre_nutriente, nombre_alimento → *cantidad_contenida*

TABLA *TOMA*

cod_toma → *nombre_toma, hora_inicio, hora_fin, od_toma*

TABLA *DIETA*

cod_dieta → *finalidad, od_dieta*

TABLA *ALIMENTO_DIETA_TOMA*

cod_dieta, nombre_alimento, cod_toma → *cantidad_toma*

TABLA *DIETA_ANIMAL_FECHAINICIO*

cod_animal, fecha_inicio → *cod_dieta, od_resultado*

8.5: CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

El esquema relacional puede definirse ahora en base a las tablas derivadas en la sección anterior, tablas normalizadas en *FNBC*, que son descritas a continuación haciendo uso de la sintaxis de *Oracle*.

BASE DE DATOS DE LAS DIETAS GANADERAS	
Animal	*(cod_animal, tipo_animal, peso, ano_nacimiento, utilidad_animal, produccion_animal, od_animal)*
Nutriente	*(nombre_nutriente, magnitud_nutriente, estado, od_nutriente)*

Animal_Nutriente	*(cod_animal, nombre_nutriente, cantidad_necesitada)*
Alimento	*(nombre_alimento, tipo_alimento, magnitud_alimento, coste_alimento, od_alimento)*
Toma	*(cod_toma, nombre_toma, hora_inicio, hora_fin, od_toma)*
Dieta	*(cod_dieta, finalidad, od_dieta)*
Nutriente_Alimento	*(nombre_nutriente, nombre_alimento, cantidad_contenida)*
Alimento_Dieta_Toma	*(cod_dieta, nombre_alimento, cod_toma, cantidad_toma)*
Dieta_Animal_FechaInicio	*(cod_animal, fecha_inicio, cod_dieta, od_resultado)*

8.5.1. Definición sintáctica de las tablas

```
   /* Inicialmente se borran las tablas */
DROP TABLE Dieta_Animal_FechaInicio;
DROP TABLE Nutriente_Alimento;
DROP TABLE Animal_Nutriente;
DROP TABLE Nutriente;
DROP TABLE Animal;
DROP TABLE Alimento_Dieta_Toma;
DROP TABLE Toma;
DROP TABLE Dieta;
DROP TABLE Alimento;
DROP SYNONYM Comidas;
   /* Se crean las tablas del esquema propuesto */
CREATE TABLE Animal (
   cod_animal VARCHAR2(10) NOT NULL,
   tipo_animal VARCHAR2(20),
   peso NUMBER(4),
   ano_nacimiento NUMBER(4),
   utilidad_animal VARCHAR2(20),
   produccion_animal VARCHAR2(20),
   od_animal LONG,
    CONSTRAINT pk_ani
      PRIMARY KEY (cod_animal),
    CONSTRAINT ck_ani
      CHECK (cod_animal = UPPER(cod_animal)),
    CONSTRAINT ck_tani
      CHECK (tipo_animal = INITCAP(tipo_animal)) );
CREATE TABLE Nutriente (
   nombre_nutriente VARCHAR2(30) NOT NULL,
   magnitud_nutriente VARCHAR2(10),
   estado VARCHAR2(10),
   od_nutriente LONG,
```

```
    CONSTRAINT pk_nut
      PRIMARY KEY (nombre_nutriente),
    CONSTRAINT ck_nut
      CHECK (nombre_nutriente = INITCAP(nombre_nutriente)) );
CREATE TABLE Animal_Nutriente (
   cod_animal VARCHAR2(10) NOT NULL,
   nombre_nutriente VARCHAR2(30) NOT NULL,
   cantidad_necesitada NUMBER(4),
    CONSTRAINT pk_anu
      PRIMARY KEY (cod_animal, nombre_nutriente),
    CONSTRAINT fk_anu_ani
      FOREIGN KEY (cod_animal)
      REFERENCES Animal(cod_animal)
      ON DELETE CASCADE,
    CONSTRAINT fk_anu_nut
      FOREIGN KEY (nombre_nutriente)
      REFERENCES Nutriente(nombre_nutriente)
      ON DELETE CASCADE,
    CONSTRAINT ck_canu
      CHECK (cantidad_necesitada > 0) );
CREATE TABLE Alimento (
   nombre_alimento VARCHAR2(30) NOT NULL,
   tipo_alimento VARCHAR2(15),
   magnitud_alimento VARCHAR2(10),
   coste_alimento NUMBER(4),
   od_alimento LONG,
    CONSTRAINT pk_ali
      PRIMARY KEY (nombre_alimento),
    CONSTRAINT ck_ali
      CHECK (nombre_alimento = INITCAP(nombre_alimento)),
    CONSTRAINT ck_coa
      CHECK (coste_alimento > = 0) );
CREATE TABLE Nutriente_Alimento (
   nombre_nutriente VARCHAR2(30) NOT NULL,
   nombre_alimento VARCHAR2(30) NOT NULL,
   cantidad_contenida NUMBER(4,2),
    CONSTRAINT pk_nua
      PRIMARY KEY (nombre_nutriente, nombre_alimento),
    CONSTRAINT fk_nua_nut
      FOREIGN KEY (nombre_nutriente)
      REFERENCES Nutriente(nombre_nutriente)
      ON DELETE CASCADE,
    CONSTRAINT fk_nua_ali
      FOREIGN KEY (nombre_alimento)
      REFERENCES Alimento(nombre_alimento)
      ON DELETE CASCADE,
    CONSTRAINT ck_cac
      CHECK (cantidad_contenida > 0) );
```



```
CREATE TABLE Toma (
   cod_toma NUMBER(2) NOT NULL,
   nombre_toma VARCHAR2(20),
   hora_inicio DATE,
   hora_fin DATE,
   od_toma LONG,
   CONSTRAINT pk_tom
     PRIMARY KEY (cod_toma),
   CONSTRAINT ck_cto
     CHECK (cod_toma > = 0),
   CONSTRAINT ck_tom
     CHECK (nombre_toma = INITCAP(nombre_toma)) );
CREATE TABLE Dieta (
   cod_dieta NUMBER(6) NOT NULL,
   finalidad VARCHAR2(60),
   od_dieta LONG,
    CONSTRAINT pk_die
      PRIMARY KEY (cod_dieta),
    CONSTRAINT ck_die
      CHECK (cod_dieta > 0) );
CREATE TABLE Alimento_Dieta_Toma (
   cod_dieta NUMBER(6) NOT NULL,
   nombre_alimento VARCHAR2(30) NOT NULL,
   cod_toma NUMBER(2) NOT NULL,
   cantidad_toma NUMBER(4,2),
    CONSTRAINT pk_adt
      PRIMARY KEY (nombre_alimento, cod_dieta, cod_toma),
    CONSTRAINT fk_adt_ali
      FOREIGN KEY (nombre_alimento)
      REFERENCES Alimento(nombre_alimento)
      ON DELETE CASCADE,
    CONSTRAINT fk_adt_die
      FOREIGN KEY (cod_dieta)
      REFERENCES Dieta(cod_dieta)
      ON DELETE CASCADE,
    CONSTRAINT fk_adt_tom
      FOREIGN KEY (cod_toma)
      REFERENCES Toma(cod_toma)
      ON DELETE CASCADE,
    CONSTRAINT ck_ctdali
      CHECK (cantidad_toma > 0) );
CREATE TABLE Dieta_Animal_FechaInicio (
   cod_animal VARCHAR2(10) NOT NULL,
   fecha_inicio DATE NOT NULL,
   cod_dieta NUMBER(6) NOT NULL,
   od_resultado LONG,
    CONSTRAINT pk_daf
      PRIMARY KEY (cod_animal, fecha_inicio),
    CONSTRAINT fk_daf_die
```

```
    FOREIGN KEY (cod_dieta)
    REFERENCES Dieta(cod_dieta),
  CONSTRAINT fk_daf_ani
    FOREIGN KEY (cod_animal)
    REFERENCES Animal(cod_animal)
    ON DELETE CASCADE );
```

8.5.2. Manipulación de la Base de Datos

Cap08ej01.sql

Obtener el código y la especie de todos los animales que necesitan el nutriente 'Riboflavina' y en qué cantidad.

De la tabla Animal_Nutriente se obtiene el código de todos los animales que necesitan el nutriente 'Riboflavina', y de la tabla Animal el código de todos los animales existentes junto con su tipo. Se procede a realizar una reunión del resultado de ambas consultas bajo la condición de que el campo cod_animal resultante de ambas tenga el mismo valor.

```
SELECT Animal.cod_animal "Animal", tipo_animal "Especie",
       cantidad_necesitada "Cantidad requerida"
  FROM Animal, Animal_Nutriente
  WHERE Animal_Nutriente.nombre_nutriente =
        'Riboflavina' AND Animal.cod_animal =
        Animal_Nutriente.cod_animal;
```

Resultado

```
Animal     Especie              Cantidad requerida
---------- -------------------- ------------------
V-01-D1    Vaca                                 50
L-03-D8    Cordero                              50
L-01-D3    Cordero                              50
V-01-D2    Vaca                                 80
V-02-D2    Vaca                                 80
V-03-D1    Vaca                                100
V-04-D2    Vaca                                100
V-05-D2    Vaca                                100
L-02-D2    Cordero                              80
P-01-D1    Cerdo                                80
P-02-D2    Cerdo                                80
P-03-D3    Cerdo                                80
P-04-D4    Cerdo                                90
```

Cap08ej02.sql

Obtener todas las vacas que comenzaron la dieta 342567 el 01/01/1999 y que pesan más de 550 kg.

De la tabla Animal se obtiene el código de todos los animales de tipo 'Vaca' y cuyo campo peso tiene un valor superior a 550. El resultado de esta subconsulta se usa en una consulta sobre la tabla Dieta_Animal_FechaInicio para obtener, de entre todos las vacas que pesan más de 550 Kg, aquellas que además comenzaron la dieta 342567 desde el 01/01/1999.

```
SELECT cod_animal "Vacas"
  FROM Dieta_Animal_FechaInicio
  WHERE cod_dieta = 342567 AND fecha_inicio =
```



```
        TO_DATE('01/01/1999','DD/MM/YYYY')
        AND cod_animal IN
          (SELECT cod_animal
            FROM Animal
            WHERE tipo_animal = 'Vaca' AND peso > 550);
```

Resultado

```
Vacas
----------
V-01-D1
```

Cap08ej03.sq

Obtener el alimento más barato que puede tomar el animal 'L-03-D8' qu necesita el nutriente 'Vitamina A' para remediar esta carencia.

De la tabla Alimento se obtiene el nombre del alimento y el coste del alimento que tiene un coste mínimo, valor obtenido de la selección de todas aquellas tuplas obtenidas de la reunión de las tablas Nutriente_Alimento y Animal_Nutriente para el nutriente 'Vitamina A', y para el animal 'L-03-D8'.

```
SELECT (coste_alimento) "Precio", nombre_alimento "Alimento"
  FROM Alimento
  WHERE coste_alimento =
    (SELECT MIN(coste_alimento)
     FROM Alimento
     WHERE nombre_alimento IN
       (SELECT nombre_alimento
          FROM Nutriente_Alimento, Animal_Nutriente
          WHERE Nutriente_Alimento.nombre_nutriente =
                'Vitamina A' AND cod_animal = 'L-03-D8'
                AND Nutriente_Alimento.nombre_nutriente =
                Animal_Nutriente.nombre_nutriente));
```

Resultado

```
    Precio Alimento
---------- --------------------------
        30 Forraje
```

Cap08ej04.sql

Obtener todos los caballos que no necesitan avena en su dieta.

De la tabla Animal se obtiene el código de todos los caballos existentes. También se obtiene de la tabla Nutriente_Alimento el nombre de todos los nutrientes que tiene la 'Avena'. El resultado de ambas subconsultas permite conocer mediante una consulta sobre la tabla Animal_Nutriente todos los caballos que, gracias al uso de NOT IN, no necesitan tomar avena pero sí cualquier otro alimento.

```
SELECT DISTINCT cod_animal
  FROM Animal_Nutriente
  WHERE nombre_nutriente NOT IN
          (SELECT nombre_nutriente
             FROM Nutriente_Alimento
             WHERE nombre_alimento = 'Avena')
        AND cod_animal IN
          (SELECT cod_animal
             FROM Animal
```

```
            WHERE tipo_animal = 'Caballo');
```

Resultado

```
COD_ANIMAL
----------
C-01-D1
```

Cap08ej05.sql

Obtener todos los animales y su tipo que comenzaron alguna dieta el 01/01/1999 y cuál es la dieta.

De la tabla Dieta_Animal_FechaInicio se obtiene el código de todos los animales que comenzaron su dieta el 01/01/1999 y el código de esa dieta. Por otro lado, en una consulta sobre la tabla Animal se obtiene el tipo de todos los animales existentes. El resultado de ambas consultas se reúne bajo la condición de que el código del animal que devuelven ambas sea el mismo.

```
SELECT Dieta_Animal_FechaInicio.cod_animal,
       cod_dieta, tipo_animal
   FROM Dieta_Animal_FechaInicio, Animal
   WHERE fecha_inicio = TO_DATE('01/01/1999','DD/MM/YYYY')
         AND Animal.cod_animal =
         Dieta_Animal_FechaInicio.cod_animal;
```

Resultado

```
COD_ANIMAL  COD_DIETA TIPO_ANIMAL
---------- ---------- --------------------
V-01-D1        342567 Vaca
```

Cap08ej06.sql

Obtener todos los alimentos que tienen los nutrientes Calcio o Magnesio.

De la tabla Nutriente_Alimento se obtiene el nombre de aquellos alimentos que están relacionados con el nutriente 'Calcio' o con el nutriente 'Magnesio'. En esta ocasión, el operador de comparación IN se utiliza para comprobar que el campo nombre_nutriente tiene un valor comprendido, no entre los valores devueltos por una subconsulta, sino directamente entre los valores indicados entre los paréntesis que lo acompañan. Es equivalente a escribir: WHERE nombre_nutriente = 'Calcio' OR nombre_nutriente = 'Magnesio'.

```
SELECT nombre_alimento "Alimentos con Ca o Mg"
   FROM Nutriente_Alimento
   WHERE nombre_nutriente IN ('Calcio', 'Magnesio')
   ORDER BY nombre_alimento;
```

Resultado

```
Alimentos con Ca o Mg
------------------------------
Avena
Forraje
Galope
Maiz
Procria A-30
Procria A-51
Procria C-30
Procria C-5
Procria V-40
Procria V-55
```


Cap08ej07.sql

Obtener todos los corderos nacidos en 1999 que han iniciado la dieta 453872 después del 01/11/1999 y cuyo peso se encuentra entre 30 y 35 kilos.

De la tabla Animal se obtiene el código de todos los animales que pertenecen al tipo 'Cordero'. La información de esta subconsulta se utiliza para obtener de la tabla Dieta_Animal_FechaInicio el código de los corderos que, entre todos los devueltos por la consulta anterior, siguen la dieta 453872 desde una fecha posterior al '01/11/1999', que han nacido en 1999 y cuyo peso actual se encuentra comprendido entre 30 y 35. Para verificar esta última condición, se utiliza el operador de comparación BETWEEN, que comprueba que un determinado valor se encuentra comprendido entre los dos extremos que están separados por AND, es decir: atributo BETWEEN extremo_inferior AND extremo_superior.

```
SELECT cod_animal "Corderos faltos de peso"
   FROM Dieta_Animal_FechaInicio
   WHERE cod_dieta = 104256 AND fecha_inicio >
         TO_DATE('01/11/1999', 'DD/MM/YYYY') AND cod_animal IN
    (SELECT cod_animal
     FROM Animal
     WHERE tipo_animal = 'Cordero' AND ano_nacimiento = 1999
           AND peso BETWEEN 30 AND 35 );
```

Resultado

```
Corderos faltos de peso
-----------------------
L-02-D2
```

Cap08ej08.sql

Obtener el peso medio de todas las vacas.

De la tabla Animal se obtiene el campo numérico peso de todos los animales pertenecientes al tipo 'Vaca', y de todos los valores devueltos se obtiene la media aritmética mediante el operador numérico AVG. El resultado aparece bajo la etiqueta 'Peso medio de las vacas'.

```
SELECT AVG(peso) "Peso medio de las vacas"
   FROM Animal
   WHERE tipo_animal = 'Vaca';
```

Resultado

```
Peso medio de las vacas
-----------------------
            453,333333
```

Cap08ej09.sql

Borrar todas las dietas seguidas por Cerdos antes de la fecha de hoy.

De la tabla Dieta_Animal_FechaInicio se borran todas las tuplas que cumplan la condición de que el código del animal que sigue la dieta pertenezca a un 'Cerdo' y que la fecha en que la comenzó sea anterior al día de hoy. Para la primera condición, se obtiene de la tabla Animal el código de todos las ovejas existentes, y para la segunda se recurre a la función de fecha SYSDATE. Dicha función devuelve la fecha y hora del momento en que se ejecuta la consulta. De este modo, se comparan todas las fechas de inicio de dietas, por parte de los cerdos, con la del sistema para comprobar la segunda condición del borrado.

```
DELETE Dieta_Animal_FechaInicio
```

```
   WHERE cod_animal IN
     (SELECT cod_animal
      FROM Animal
      WHERE tipo_animal = 'Cerdo') AND fecha_inicio < SYSDATE;
```

Resultado

```
    4 filas borradas.
```

Cap08ej10.sql

Introducir una toma para un animal.

Se realiza la operación INSERT sobre la tabla Toma. El interés de esta operación se encuentra en el uso de la función TO_DATE para convertir una determinada cadena de caracteres en un tipo dato DATE. Su uso es similar al ya visto para introducir fechas en una tabla, salvo en la cadena que indica el formato en base al cual se debe realizar la conversión de tipos de datos.

```
INSERT INTO Toma
   VALUES (17,'Muestra',TO_DATE('10:30 AM','HH:MI AM'),
          TO_DATE('11:00 AM','HH:MI AM'),
          'Se espera que gane peso');
```

Resultado

```
    1 fila insertada.
```

Cap08ejP1.sql

Crear un procedimiento PL/SQL que inicie una nueva dieta (cod_dieta es el primer parámetro del procedimiento) para un determinado tipo de animal (segundo parámetro del procedimiento), si su peso es inferior a un determinado valor (tercer parámetro del procedimiento).

Se obtiene el código del animal y el peso, recorriendo la tabla animal a través de un cursor y un bucle FOR en el cual se utiliza la variable "cada_animal" de tipo ROWTYPE. Si el peso del animal es inferior a un determinado valor (parámetro peso_animal) se asigna, con fecha actual, una nueva dieta (parámetro codigo_de_dieta) para dicho animal.

```
CREATE OR REPLACE PROCEDURE nueva_dieta
   /* Se definen las variables de la interfaz */
   (codigo_de_dieta dieta.cod_dieta%TYPE,
   tipo_de_animal animal.tipo_animal%TYPE,
   peso_animal animal.peso%TYPE)
AS
   /* Se declara un cursor para recorrer la tabla Animal */
   CURSOR animales IS
     SELECT cod_animal, peso
       FROM Animal
       WHERE tipo_animal = tipo_de_animal;
BEGIN
   /* Se recorre el cursor declarado */
   FOR cada_animal IN animales LOOP
   /* Se eliminan las tuplas que cumplen las condiciones de la
   interfaz */
     IF cada_animal.peso < peso_animal THEN
       INSERT INTO Dieta_Animal_Fechainicio
         VALUES (cada_animal.cod_animal, SYSDATE,
```



```
                codigo_de_dieta, 'Dieta especial');
     END IF;
   END LOOP;
COMMIT;
   /* Se controlan los errores */
EXCEPTION
   WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE(SQLERRM);
   ROLLBACK;
END;
/
```

Comprobación *(Consultamos el estado previo de la base de datos)*

```
SELECT DAF.cod_dieta, DAF.fecha_inicio,
       animal.cod_animal, animal.peso
   FROM dieta_animal_fechainicio DAF, animal
   WHERE DAF.cod_animal=animal.cod_animal
         AND animal.cod_animal IN
     (SELECT cod_animal
      FROM animal
      WHERE tipo_animal = 'Vaca')
   ORDER BY 4,1;
```

COD_DIETA	FECHA_IN	COD_ANIMAL	PESO
342567	01/01/99	V-01-D1	650
642567	01/10/99	V-01-D1	650
642567	01/11/99	V-01-D2	490
642567	01/11/99	V-02-D2	400
642567	01/11/99	V-03-D1	400
342567	01/11/99	V-05-D2	430
342567	01/11/99	V-04-D2	350

Ejecución *(Probamos el procedimiento)*

```
    EXEC nueva_dieta (111111,'Vaca',600);
```

Resultado *(Consultamos el estado final de la base de datos)*

COD_DIETA	FECHA_IN	COD_ANIMAL	PESO
342567	01/01/99	V-01-D1	650
642567	01/10/99	V-01-D1	650
111111	09/11/00	V-01-D2	490
642567	01/11/99	V-01-D2	490
111111	09/11/00	V-05-D2	430
342567	01/11/99	V-05-D2	430
111111	09/11/00	V-03-D1	400
111111	09/11/00	V-02-D2	400
642567	01/11/99	V-02-D2	400
642567	01/11/99	V-03-D1	400
111111	09/11/00	V-04-D2	350
342567	01/11/99	V-04-D2	350

CAPÍTULO 9

COLECCIONES DE MARIPOSAS

9.1. ENUNCIADO DEL PROBLEMA

Actualmente, una de las principales tareas de los biólogos es el estudio de la población de las especies naturales que pueblan España, siendo los insectos uno de los reinos naturales más estudiados, y dentro de éste, el orden de los lepidópteros, mariposas más concretamente. Se desea considerar la información referente al estudio de estos insectos, teniendo en cuenta que se desea representar la información sobre los ejemplares de mariposas que son capturados, bien para su observación o bien para ser incluidos en una colección, considerando la zona donde son capturados, la fecha de la captura y la persona que realiza la misma.

Además es necesario tener en cuenta los siguientes supuestos semánticos:

SUPUESTO 1: Se considera que un ejemplar de mariposa pertenece a una única especie. Una especie pertenece a un único género y un género a una única familia natural.

SUPUESTO 2: El nombre científico de la especie de mariposas es único, pero el nombre común que tiene una especie puede variar según la zona geográfica donde se encuentra la mariposa. Si bien dentro de una determinada zona el nombre común de una especie es único.

SUPUESTO 3: Ya sea para su observación o para formar parte de una colección, la mariposa ha de ser capturada primero. La captura de cada ejemplar la realiza sólo una persona.

SUPUESTO 4: Una determinada persona sólo puede ser propietaria de una colección, y los ejemplares de mariposa que pertenecen a esta colección pueden haber sido capturados por otras personas.

SUPUESTO 5: Se desea mantener información de las familias, géneros y especies a las que pertenecen los ejemplares de mariposas, independientemente de que haya sido capturado algún ejemplar de los mismos.

SUPUESTO 6: Una mariposa sólo puede pertenecer a una colección y una colección estará al menos formada por un ejemplar de mariposa (como es lógico).

SUPUESTO 7: El nombre de una zona geográfica donde es capturado un ejemplar es único; es decir, se considera que no existen dos zonas geográficas con el mismo nombre; no siendo de interés mantener información añadida sobre los lugares en donde las mariposas son capturadas.

9.2. MODELO CONCEPTUAL

Se trata de representar la información sobre el estudio de la población de lepidópteros o mariposas en España.

9.2.1. Análisis de los tipos de entidad

Considerando el enunciado del problema abordado, se puede extraer los siguientes tipos de entidad, como se muestra en la figura 9.1:

Tipo de entidad *Familia*: el cual representa al objeto del mundo real "*grupo taxonómico perteneciente a un orden natural formado por los géneros naturales que poseen gran número de características comunes*". Para este tipo de entidad se consideran los atributos *nombre_familia* y *od_familia* (SUPUESTO 5).

El atributo *od_familia* representa cualquier tipo de información que se desee considerar respecto de una determinada familia. El identificador del tipo de entidad *Familia* es el atributo *nombre_familia*, ya que no existen dos familias naturales con el mismo nombre (SUPUESTO 1).

Tipo de entidad *Genero*: Este tipo de entidad representa al objeto del mundo real "*conjunto de especies que tienen varias características comunes*". El tipo de entidad *Genero* se puede considerar como un tipo de entidad débil por existencia respecto al tipo de entidad *Familia*, ya que un género no existe si no existe una familia a la que pertenecer (SUPUESTO 5). Para este tipo de entidad se consideran los atributos *nombre_genero* y *od_genero*, este último representando cualquier dato que se desee mantener de un determinado género de mariposas.

El identificador de este tipo de entidad es el atributo *nombre_genero*, ya que no existen dos géneros naturales con el mismo nombre (SUPUESTO 1).

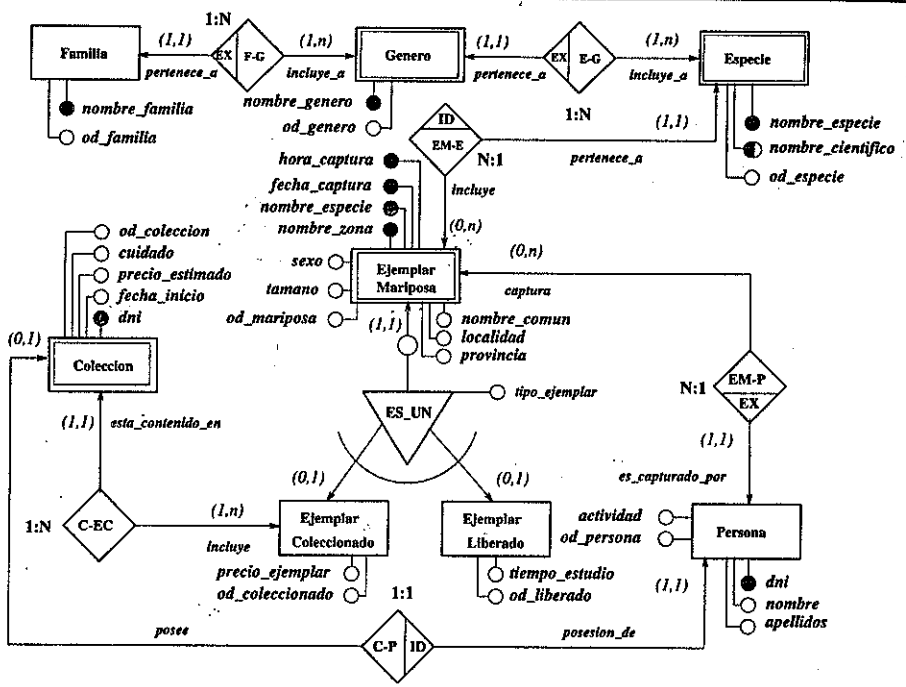


Figura 9.1 Esquema E-R del problema de las colecciones de mariposas

Tipo de entidad *Especie*: el cual representa al objeto del mundo real "*cada uno de los grupos en que se dividen los géneros*". Se trata de un tipo de entidad débil por existencia respecto al tipo de entidad *Genero*, ya que no existe una especie natural si no existe un género al que pertenecer. Para este tipo de entidad se consideran los atributos *nombre_especie*, *nombre_cientifico* y *od_especie* (SUPUESTOS 2 y 5).

El identificador de este tipo de entidad es el atributo *nombre_especie*, ya que no hay dos especies naturales con el mismo nombre. Además, se considera el atributo *nombre_cientifico* como identificador alternativo de este tipo de entidad, debido a que una especie se puede identificar con su nombre científico, puesto que éste es también único (SUPUESTO 2). El atributo *od_especie* representa cualquier otra información de interés.

Tipo de entidad *Persona*: el cual representa al objeto del mundo real "*persona que está interesada en las mariposas para su estudio y/o colección*". Los atributos que se van a considerar para este tipo de entidad son: *dni*, *nombre*, *apellidos*, *actividad*, *fecha_inicio*, *precio_estimado*, *cuidado* y *od_persona*. El identificador de este tipo de entidad es el atributo *dni*, considerándose que no existen dos personas con el mismo número de documento nacional de identidad (SUPUESTOS 3 y 4).

El atributo *actividad* representa la actividad laboral a la que se dedica la persona (puede ser un investigador, profesor, etc.), los atributos *fecha_inicio*, *precio_estimado* y *cuidado* son atributos propios de la colección que posee la persona (si la posee).

Como no todas las personas que capturan un ejemplar de mariposa poseen una colección, el tipo de entidad *Persona* puede ser expandido en dos tipos de entidad: *Persona* y *Coleccion*, donde el primer tipo de entidad representa a cualquier persona relacionada con las mariposas, bien un coleccionista o un mero observador y estudioso de las mariposas, y el tipo de entidad *Coleccion* representa a las colecciones de mariposas que poseen las personas. Por ello, el tipo de entidad *Coleccion* es un tipo de entidad débil por identificación respecto al tipo de entidad *Persona*, puesto que para que exista una colección de mariposas debe existir una persona que la haga. Además, para identificar una entidad del tipo de entidad *Coleccion* es necesario conocer la persona que la posee (SUPUESTO 4).

Para el tipo de entidad *Persona* se consideran los atributos *dni*, *nombre*, *apellidos*, *actividad* y *od_persona*, y para el tipo de entidad *Coleccion* se consideran los atributos *fecha_inicio*, *precio_estimado*, *cuidado* y *od_coleccion*.

El atributo *fecha_inicio* representa la fecha en la que una persona comenzó su colección de mariposas. Este atributo se podría eliminar si la fecha de comienzo de una colección coincidiera con la fecha de captura del primer ejemplar de ésta, pero se va a considerar que la persona que comienza una colección puede no haber capturado el primer ejemplar de la misma, con lo que la fecha de inicio de la colección sería la fecha en la que se adquiere el primer ejemplar de mariposa. El atributo *precio_estimado* representa el precio que puede tener esa colección si fuese vendida (puede ser diferente a la suma de los precios de las mariposas que componen la colección), y el atributo *cuidado* representa cualquier cuidado especial que sea necesario para conservar perfectamente la colección.

El identificador del tipo de entidad *Coleccion* es el atributo *dni*, que es heredado del tipo de entidad *Persona* con el cual mantiene una dependencia de identificación.

Tipo de entidad *EjemplarMariposa*: el cual representa al objeto del mundo real "*ejemplar de lepidóptero que es capturado por una determinada persona para ser observado o bien para ser añadido a una colección*". Este tipo de entidad se considera un tipo de entidad débil por identificación respecto al tipo de entidad *Especie* y por existencia respecto al tipo de entidad *Persona*, ya que para que exista una entidad de este tipo en el sistema de información debe haber sido capturado por alguien (SUPUESTO 3) y debe conocerse la especie a la que pertenece (SUPUESTO 1).

Los atributos de este tipo de entidad son *nombre_especie*, el cual es heredado del tipo de entidad *Especie*, *nombre_zona*, *fecha_captura*, *hora_captura*, *tamano*, *sexo*, *localidad*, *provincia*, *nombre_comun* y *od_mariposa* (ENUNCIADO).

Los atributos *nombre_zona*, *localidad*, *provincia*, *fecha_captura* y *hora_captura* hacen referencia al lugar y momento de la captura de la mariposa. El atributo

nombre_comun representa el nombre vulgar con el que se conoce a una clase de mariposa (hay que tener en cuenta que como este nombre puede variar según la zona geográfica, el atributo tomará el nombre común de la zona donde fue capturado el ejemplar de mariposa) (SUPUESTO 2). El atributo *sexo* representa el sexo del ejemplar de lepidóptero capturado.

El tipo de entidad *EjemplarMariposa* es conveniente especializarlo en dos subtipos de entidad: el tipo de entidad *EjemplarLiberado* que hace referencia a los ejemplares de mariposa que son capturados sólo para su observación y posteriormente son liberados, y el tipo de entidad *EjemplarColeccionado* que hace referencia a los ejemplares de mariposa que pasan a formar parte de la colección de una persona. Ambos tipos de entidad heredan los atributos del tipo de entidad *EjemplarMariposa*, y además de éstos, el tipo de entidad *EjemplarLiberado* incorpora los atributos *tiempo_estudio* y *od_liberado*, los cuales representan el tiempo que se ha dedicado al estudio del ejemplar así como cualquier otra información que se desee considerar sobre el proceso de estudio de la mariposa respectivamente.

El tipo de entidad *EjemplarColeccionado* incorpora el atributo *precio_ejemplar*, que representa el valor que puede tener el ejemplar capturado, y *od_coleccionado*, que representa cualquier información que se desee considerar sobre el proceso de preparación de la mariposa para formar parte de la colección.

La identificación de las entidades del tipo *EjemplarMariposa* y, por tanto, de los subtipos *EjemplarColeccionado* y *EjemplarLiberado*, se realiza mediante la agregación de los atributos *nombre_especie*, *nombre_zona*, *fecha_captura* y *hora_captura*.

9.2.2. Análisis de los tipos de interrelación

Los tipos de entidad antes mencionados se encuentran relacionados de la siguiente forma:

Tipo de interrelación *Familia/Genero* **(F-G)**: el cual relaciona los tipos de entidad *Familia* y *Genero*. Se considera que un género pertenece a una y sólo una familia Por lo tanto, el tipo de entidad *Familia* interviene en la interrelación con cardinalidades *(1,1)*.

A una determinada familia pueden pertenecer varios géneros, pero como mínimo uno, por lo que el tipo de entidad *Genero* participa en la interrelación con cardinalidades *(1,n)* (SUPUESTO 1).

Tipo de interrelación *Especie/Genero* **(E-G)**: el cual relaciona los tipos de entidad *Especie* y *Genero*, y representa el género al que pertenece una especie. Se considera que una determinada especie pertenece a un género y sólo a uno, por lo que el tipo de entidad *Genero* participa en la interrelación con cardinalidades *(1,1)*. En un género se pueden incluir varias especies, pero como mínimo una (si no, no existiría el género), por lo que el tipo de entidad *Especie* participa en la interrelación con cardinalidades *(1,n)* (SUPUESTO 1).

Tipo de interrelación *EjemplarMariposa/Especie* **(EM-E)**: el cual relaciona los tipos de entidad *EjemplarMariposa* y *Especie*, siendo *EjemplarMariposa* un tipo de entidad débil por identificación respecto al tipo de entidad *Especie*. Se considera que un determinado ejemplar de mariposa pertenece a una y sólo una especie, por tanto, el tipo de entidad *Especie* participa con cardinalidades *(1,1)*.

Por otro lado, se considera que una especie tiene existencia propia en el sistema de información exista o no algún ejemplar que pertenezca a ella, por tanto, el tipo de entidad *EjemplarMariposa* participa con cardinalidades *(0,n)* (SUPUESTO 5).

Tipo de interrelación *Coleccion/Persona* **(C-P)**: el cual relaciona los tipos de entidad *Coleccion* y *Persona*. Una determinada persona puede no tener ninguna colección de mariposas y como máximo una, por lo que el tipo de entidad *Coleccion* participa en la interrelación con cardinalidades *(0,1)*. Una colección sólo pertenece a una sola persona, participando el tipo de entidad *Persona* con cardinalidades *(1,1)* (SUPUESTO 4).

Tipo de interrelación *EjemplarMariposa/Persona* **(EM-P)**: este tipo de interrelación relaciona los tipos de entidad *EjemplarMariposa* y *Persona*, representando a la persona que captura un ejemplar de mariposa. Una determinada mariposa sólo es capturada por una persona, por lo que el tipo de entidad *Persona* participa con cardinalidades *(1,1)* (SUPUESTO 3).

Por otro lado, en un mismo momento y por una sola persona, pueden ser varios los ejemplares de mariposa que se capturan, por lo que el tipo de entidad *EjemplarMariposa* participa con cardinalidades *(0,n)* (recuérdese que pueden existir personas en el sistema que no hayan capturado nunca ninguna mariposa, sino que simplemente sean coleccionistas de las mismas). No existe ningún supuesto, en el enunciado del problema, que restrinja el número de ejemplares que pueden ser capturados al mismo tiempo. Por ello, es necesario incluir una nueva restricción de la forma siguiente:

SUPUESTO 8: En una misma zona y por una misma persona pueden ser capturados, para colección y/o para observación, un número de ejemplares no determinado, siempre y cuando los ejemplares pertenezcan a distintas especies de mariposas. Los ejemplares que se capturen y pertenezcan a la misma especie serán considerados como un único ejemplar, o bien que se han capturado a una hora diferente (un segundo más tarde, por ejemplo).

Este supuesto es necesario debido a que la identificación de los ejemplares de mariposa se realiza sobre la base de la agregación de los atributos: *nombre_especie, nombre_zona, fecha_captura, hora_captura* y, por tanto, no existe forma de diferenciar a dos ejemplares de la misma especie capturados al mismo tiempo. Además, en el tipo de entidad *EjemplarMariposa* no se ha considerado ningún atributo que represente al número de ejemplares de la misma especie que se capturan en la misma zona, fecha y hora.

Tipo de interrelación *Coleccion/EjemplarColeccionado* **(C-EC)**: el cual relaciona los tipos de entidad *Coleccion* y *EjemplarColeccionado*, y representa a los ejemplares de mariposa que pertenecen a una determinada colección. Una colección estará formada por varios ejemplares de mariposa o un ejemplar al menos (SUPUESTO 6), por lo que el tipo de entidad *EjemplarColeccionado* participa en la relación con cardinalidades *(1,n)*. Por otro lado, un ejemplar de mariposa pertenece a una y sólo a una colección, por lo que el tipo de entidad *Coleccion* participa en la relación con cardinalidades *(1,1)*.

9.3. MODELO RELACIONAL

Una vez realizado el modelo conceptual del problema abordado se va a realizar el modelo relacional del mismo.

9.3.1. Tabla *Familia*

La tabla *Familia* se forma a partir del tipo de entidad que lleva el mismo nombre, y de este tipo de entidad toma los atributos *nombre_familia* y *od_familia*. Del conjunto de atributos de la tabla, la clave principal es el atributo *nombre_familia* (regla *RTECAR-1*). Ninguno de los atributos de la tabla mantiene referencia alguna con atributos de otras tablas. La tabla *Familia* queda de la forma:

Familia *(nombre_familia, od_familia)*

9.3.2. Tabla *Genero*

La tabla *Genero* se forma a partir del tipo de entidad que lleva el mismo nombre, y de éste toma los atributos *nombre_genero* y *od_genero*. Además, la tabla *Genero* incorpora el atributo *nombre_familia* del tipo de entidad *Familia* por ser el identificador de éste, ya que mantiene una relación uno a muchos en el tipo de interrelación *(F-G)* con este tipo de entidad, y a través de este atributo mantiene referencia con la tabla *Familia* (regla *RTECAR-3.1*). El identificador de esta tabla es el atributo *nombre_genero*, y la tabla *Genero* queda de la forma:

Genero *(nombre_genero,* **nombre_familia**, *od_genero)*

9.3.3. Tabla *Especie*

La tabla *Especie* se forma a partir del tipo de entidad *Especie*. Esta tabla incorpora los atributos *nombre_especie*, *nombre_cientifico*, *nombre_genero* y *od_especie*. Los atributos *nombre_especie*, *nombre_cientifico* y *od_especie* son tomados del tipo de entidad *Especie*, y el atributo *nombre_genero* es tomado del identificador de la tabla *Genero* con la que la tabla *Especie* mantiene una relación uno a muchos por el tipo de interrelación *(E-G)*, y a través de este atributo mantiene referencia con la tabla *Genero* (regla *RTECAR-3.1*).

La clave principal de esta tabla es el atributo *nombre_especie*, considerándose el atributo *nombre_cientifico* como clave alterna. La tabla *Especie* queda de la forma:

Especie *(nombre_especie, nombre_cientifico,* **nombre_genero**, *od_especie)*

9.3.4. Tabla *Aficionado*

La tabla *Aficionado*[21] se forma a partir del tipo de entidad *Persona*. Esta tabla incorpora los atributos *dni*, *nombre*, *apellidos*, *actividad* y *od_persona* (regla *RTECAR-1*). Ninguno de estos atributos mantiene referencia con atributos de las otras tablas. El identificador de esta tabla es el atributo *dni*, no considerándose otro atributo como clave alterna. Esta tabla queda de la forma:

Aficionado *(dni, nombre, apellidos, actividad, od_persona)*

9.3.5. Tabla *Coleccion*

Esta tabla se forma a partir del tipo de entidad que lleva el mismo nombre, y de éste toma los atributos *dni*, *fecha_inicio*, *precio_estimado*, *cuidado* y *od_coleccion*. El identificador de esta tabla es el atributo *dni*, por el cual se mantiene referencia con la tabla *Aficionado*, y es tomado a través del tipo de interrelación débil por identificación que existe con el tipo de entidad *Persona* (regla *RTECAR-2.2*). La tabla *Coleccion* queda de la forma:

Coleccion *(dni, fecha_inicio, precio_estimado, cuidado, od_coleccion)*

9.3.6. Tabla *EjemplarMariposa*

La tabla *EjemplarMariposa* se forma a partir del tipo de entidad que tiene el mismo nombre. Esta tabla toma del tipo de entidad *EjemplarMariposa* los atributos *nombre_especie*, *nombre_zona*, *fecha_captura*, *hora_captura*, *tamano*, *sexo*, *localidad*, *provincia*, *nombre_comun* y *od_mariposa*. Además de éstos, la tabla incorpora el atributo *dni_captura* que representa al identificador de la tabla *Aficionado* con la que mantiene una relación padre-hijo por la interrelación *(EM-P)*, y el atributo *tipo_ejemplar* que caracteriza la especialización del tipo de entidad *EjemplarMariposa*, y que podrá tomar dos valores posibles, "C" —para ejemplares que pasan a formar parte de una colección— y "L" —para ejemplares que son liberados tras su estudio— (regla *PRTECAR-5*).

La tabla *EjemplarMariposa* mantiene las siguientes referencias con otras tablas:

- Con la tabla *Especie* a través del atributo *nombre_especie*, el cual es heredado del tipo de entidad *Especie* por la relación existente con la misma (regla *RTECAR-3.1*).

[21] Como observará el lector, las tablas en el modelo relacional pueden tener el mismo o diferente nombre, como es este caso, que el del tipo de entidad del cual se derivan.

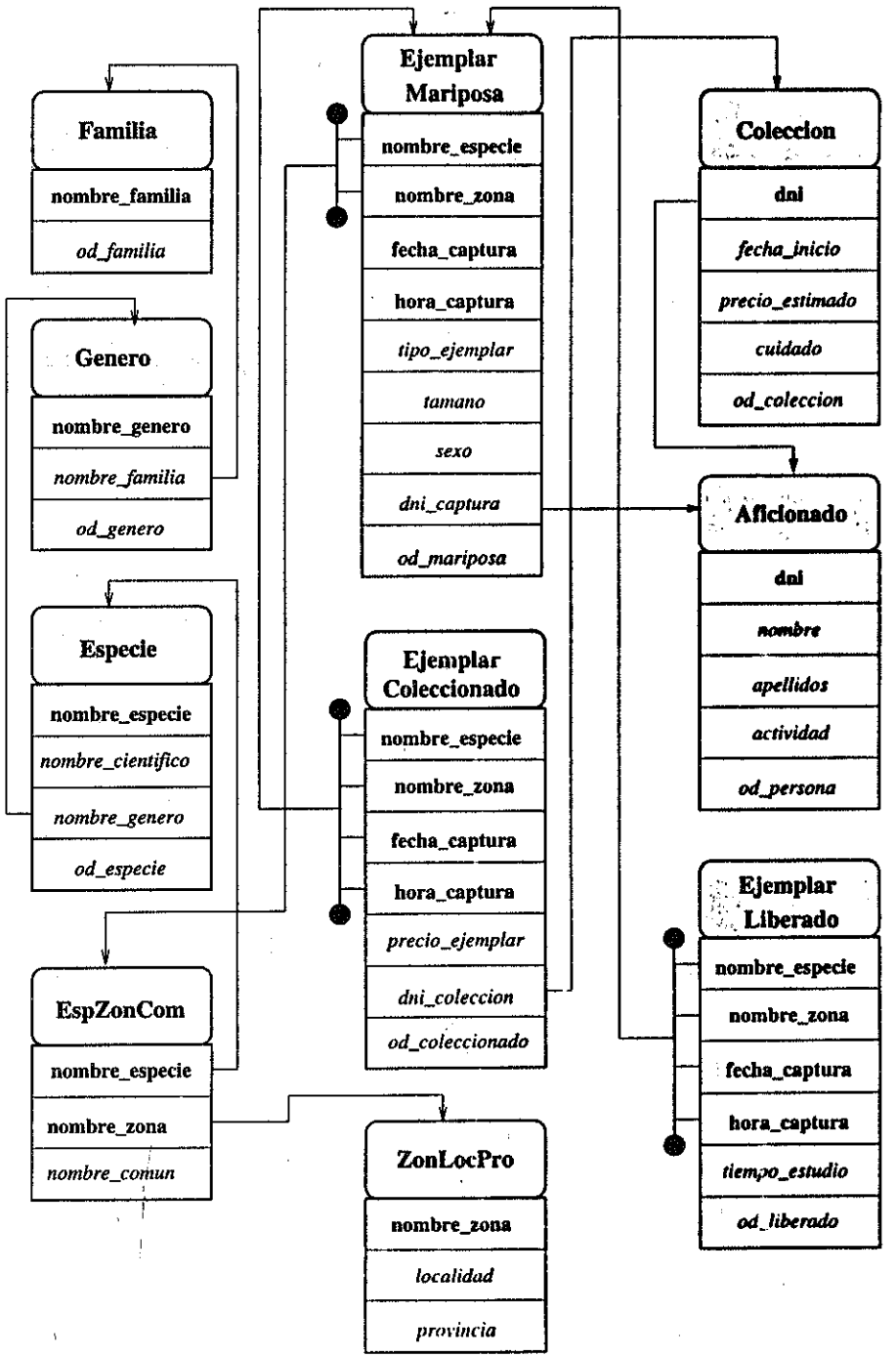


Figura 9.2 Diagrama relacional de las colecciones de mariposas

- Con la tabla *Aficionado* a través del atributo *dni* (regla *RTECAR-3.1*).

La clave principal de esta tabla es la agregación de los atributos *nombre_especie*, *nombre_zona*, *fecha_captura* y *hora_captura*. La tabla *EjemplarMariposa* queda de la forma:

EjemplarMariposa *(**nombre_especie**, nombre_zona, fecha_captura, hora_captura, tipo_ejemplar, tamano, sexo, localidad, provincia, nombre_comun,* **dni_captura**, *od_mariposa)*

9.3.7. Tabla *EjemplarLiberado*

Esta tabla se forma a partir del tipo de entidad que lleva el mismo nombre y se deriva de una especialización del tipo de entidad *EjemplarMariposa* y, por tanto, mantiene una referencia a través de los atributos *nombre_especie*, *nombre_zona*, *fecha_captura* y *hora_captura* con la tabla *EjemplarMariposa*, además de incorporar los atributos *tiempo_estudio* y *od_liberado* (reglas *PRTECAR-5* y *RTECAR-2.2*). La tabla *EjemplarLiberado* queda de la forma:

EjemplarLiberado *(**nombre_especie, nombre_zona, fecha_captura, hora_captura**, tiempo_estudio, od_liberado)*

9.3.8. Tabla *EjemplarColeccionado*

Esta tabla también se forma a partir del tipo de entidad *EjemplarColeccionado* que se deriva de una especialización del tipo de entidad *EjemplarMariposa* (regla *PRTECAR-5*). Esta tabla tiene los atributos *precio_ejemplar* y *od_coleccionado*, además de:

- Los atributos *nombre_especie*, *nombre_zona*, *fecha_captura* y *hora_captura*, con los que mantiene referencia con la tabla *EjemplarMariposa* (regla *RTECAR-2.2*).
- El atributo *dni_coleccion* es tomado del identificador de la tabla *Coleccion*, con la que mantiene una relación padre-hijo en el tipo de interrelación *(C-EC)* (regla *RTECAR-3.1*).

EjemplarColeccionado *(**nombre_especie, nombre_zona, fecha_captura, hora_captura**, precio_ejemplar,* **dni_coleccion**, *od_coleccionado)*.

9.4. NORMALIZACIÓN DEL MODELO

Las tablas *Familia*, *Genero*, *Especie*, *Coleccion*, *Aficionado*, *EjemplarLiberado* y *EjemplarColeccionado* se encuentran en *FNBC*. La tabla *EjemplarMariposa* se encuentra en *FN1* al ser todos los atributos atómicos, pero la tabla no se encuentra en *FNBC* debido a que existen dependencias no completas de atributos no primos con la clave:

- El atributo *nombre_comun* depende sólo de los atributos *nombre_especie* y *nombre_zona* (como se explica en el enunciado del problema, el nombre común de una especie depende del lugar donde el ejemplar de mariposa es capturado), y no de los atributos *fecha_captura* y *hora_captura*, por lo que la dependencia del atributo *nombre_comun* no es completa con la clave de la tabla (SUPUESTO 2).

  Para eliminar esta dependencia parcial se crea una tabla llamada *EspZonCom* formada por los atributos *nombre_especie*, *nombre_zona* y *nombre_comun*, donde la clave es la agregación de los atributos *nombre_especie* y *nombre_zona*. Esta nueva tabla se encuentra en *FNBC*, ya que la dependencia que existe entre el atributo *nombre_comun* y la clave de la tabla es completa.

EspZonCom *(**nombre_especie, nombre_zona**, nombre_comun)*

- Análogamente sucede con los atributos *localidad* y *provincia*. Estos dos atributos no dependen completamente de la clave de la tabla *EjemplarMariposa*, ya que sólo dependen del atributo *nombre_zona* y no del resto de atributos que forman la clave de la tabla, por lo que existe una dependencia no completa.

  Para eliminar esta dependencia parcial se crea una tabla llamada *ZonLocPro* formada por los atributos *nombre_zona*, *localidad* y *provincia*, donde la clave de la tabla es el atributo *nombre_zona*. Esta tabla se encuentra en *FNBC*, ya que las dependencias son completas, y además, aunque el atributo *provincia* depende del atributo *localidad* (una localidad siempre se encuentra en la misma provincia, no se va a considerar esta dependencia), también depende del atributo *nombre_zona*, ya que se ha considerado que no existen dos zonas con el mismo nombre, por lo que para un determinado valor del atributo *nombre_zona* el atributo *provincia* tomará siempre el mismo valor.

ZonLocPro *(nombre_zona, localidad, provincia)*

Considerando las nuevas tablas, la nueva tabla *EjemplarMariposa* resultante se encuentra en *FNBC*, con la siguiente estructura:

EjemplarMariposa *(**nombre_especie**, nombre_zona, fecha_captura, hora_captura, tipo_ejemplar, tamano, sexo, localidad,* **dni_captura**, *od_mariposa)*

9.5. CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

Una vez normalizado, el modelo relacional propuesto queda de la forma siguiente (véase la figura 9.2):

BASE DE DATOS DE LAS MARIPOSAS	
Familia	*(nombre_familia, od_familia)*
Genero	*(nombre_genero,* **nombre_familia**, *od_genero)*
Especie	*(nombre_especie, nombre_cientifico,* **nombre_genero**, *od_especie)*
Aficionado	*(dni, nombre, apellidos, actividad, od_persona)*
Coleccion	*(***dni**, *fecha_inicio, precio_estimado, cuidado, od_coleccion)*
EspZonCom	*(***nombre_especie, nombre_zona**, *nombre_comun)*
ZonLocPro	*(nombre_zona, localidad, provincia)*
EjemplarMariposa	*(***nombre_especie**, *nombre_zona, fecha_captura, hora_captura, tipo_ejemplar, tamano, sexo,* **dni_captura**, *od_mariposa)*
EjemplarLiberado	*(***nombre_especie, nombre_zona, fecha_captura, hora_captura**, *tiempo_estudio, od_liberado)*
EjemplarColeccionado	*(***nombre_especie, nombre_zona, fecha_captura, hora_captura**, *precio_ejemplar,* **dni_coleccion**, *od_coleccionado)*

9.5.1. Definición sintáctica de las tablas

```
   /* Inicialmente se borran las tablas */
DROP TABLE EjemplarLiberado;
DROP TABLE EjemplarColeccionado;
DROP TABLE EjemplarMariposa;
DROP TABLE Coleccion;
DROP TABLE EspZonCom;
DROP TABLE Especie;
DROP TABLE Genero;
DROP TABLE Familia;
DROP TABLE ZonLocpro;
DROP TABLE Aficionado;
DROP VIEW Datos_coleccion;

   /* Se crean las tablas, del esquema propuesto */
```



```
CREATE TABLE Familia  (
   nombre_familia VARCHAR2(30) NOT NULL,
   od_familia LONG,
    CONSTRAINT pk_fam
      PRIMARY KEY (nombre_familia),
    CONSTRAINT ck_fam
      CHECK (nombre_familia = INITCAP(nombre_familia)) );
CREATE TABLE Genero  (
   nombre_genero VARCHAR2(30) NOT NULL,
   nombre_familia VARCHAR2(30) NOT NULL,
   od_genero LONG,
    CONSTRAINT pk_gen
      PRIMARY KEY (nombre_genero),
    CONSTRAINT ck_gen
      CHECK (nombre_genero = INITCAP(nombre_genero)),
    CONSTRAINT fk_gen_fam
      FOREIGN KEY (nombre_familia)
      REFERENCES Familia(nombre_familia) );
CREATE TABLE Especie  (
   nombre_especie VARCHAR2(30) NOT NULL,
   nombre_cientifico VARCHAR2(30) NOT NULL,
   nombre_genero VARCHAR2(30) NOT NULL,
   od_especie LONG,
    CONSTRAINT pk_esp
      PRIMARY KEY (nombre_especie),
    CONSTRAINT ak_esp
      UNIQUE (nombre_cientifico),
    CONSTRAINT ck_esp
      CHECK (nombre_especie = INITCAP(nombre_especie)),
    CONSTRAINT ck_cie
      CHECK (nombre_cientifico = INITCAP(nombre_cientifico)),
    CONSTRAINT fk_esp_gen
      FOREIGN KEY (nombre_genero)
      REFERENCES Genero(nombre_genero) );
CREATE TABLE Aficionado  (
   dni NUMBER(8) NOT NULL,
   nombre VARCHAR2(20) NOT NULL,
   apellidos VARCHAR2(40) NOT NULL,
   actividad VARCHAR2(30),
   od_persona LONG,
    CONSTRAINT pk_aficIo
      PRIMARY KEY (dni) );
CREATE TABLE Coleccion  (
   dni NUMBER(8) NOT NULL,
   fecha_inicio DATE,
   precio_estimado NUMBER(10),
   cuidado VARCHAR2(30),
   od_coleccion LONG,
```

```
    CONSTRAINT pk_col
      PRIMARY KEY (dni),
    CONSTRAINT fk_col_per
      FOREIGN KEY (dni)
      REFERENCES Aficionado(dni)
      ON DELETE CASCADE );
CREATE TABLE ZonLocPro  (
   nombre_zona VARCHAR2(20) NOT NULL,
   localidad VARCHAR2(20),
   provincia VARCHAR2(20),
    CONSTRAINT pk_zlp
      PRIMARY KEY (nombre_zona),
    CONSTRAINT ck_zlp
      CHECK (nombre_zona = INITCAP(nombre_zona)) );
CREATE TABLE EspZonCom  (
   nombre_especie VARCHAR2(30) NOT NULL,
   nombre_zona VARCHAR2(20) NOT NULL,
   nombre_comun VARCHAR2(40),
    CONSTRAINT pk_esz
      PRIMARY KEY (nombre_especie, nombre_zona),
      CONSTRAINT fk_esz_esp
    FOREIGN KEY (nombre_especie)
      REFERENCES Especie(nombre_especie),
    CONSTRAINT fk_esz_zon
      FOREIGN KEY (nombre_zona)
      REFERENCES ZonLocPro(nombre_zona) );
CREATE TABLE EjemplarMariposa  (
   nombre_especie VARCHAR2(30) NOT NULL,
   nombre_zona VARCHAR2(20) NOT NULL,
   fecha_captura DATE NOT NULL,
   hora_captura DATE NOT NULL,
   tipo_ejemplar CHAR(1),
   tamano NUMBER(4),
   sexo CHAR(1),
   dni_captura NUMBER(8) NOT NULL,
   od_mariposa LONG,
    CONSTRAINT pk_ejm
      PRIMARY KEY (nombre_especie, nombre_zona,
        fecha_captura, hora_captura),
    CONSTRAINT fk_ejm_ezc
      FOREIGN KEY (nombre_especie, nombre_zona)
      REFERENCES EspZonCom(nombre_especie, nombre_zona),
    CONSTRAINT fk_ejm_per
      FOREIGN KEY (dni_captura)
      REFERENCES Aficionado(dni),
    CONSTRAINT ck_ejm_sexo
      CHECK (sexo IN ('M', 'H', 'I')),
    CONSTRAINT ck_ejm_tipo
      CHECK (tipo_ejemplar IN ('L','C')) );
```



```
CREATE TABLE EjemplarLiberado  (
   nombre_especie VARCHAR2(30) NOT NULL,
   nombre_zona VARCHAR2(20) NOT NULL,
   fecha_captura DATE NOT NULL,
   hora_captura DATE NOT NULL,
   tiempo_estudio NUMBER(4),
   od_liberado LONG,
    CONSTRAINT pk_ejl
      PRIMARY KEY (nombre_especie, nombre_zona,
        fecha_captura, hora_captura),
    CONSTRAINT fk_ejl_ejm
      FOREIGN KEY (nombre_especie, nombre_zona,
        fecha_captura, hora_captura)
      REFERENCES EjemplarMariposa(nombre_especie,
        nombre_zona, fecha_captura, hora_captura)
      ON DELETE CASCADE );
CREATE TABLE EjemplarColeccionado  (
   nombre_especie VARCHAR2(30) NOT NULL,
   nombre_zona VARCHAR2(20) NOT NULL,
   fecha_captura DATE NOT NULL,
   hora_captura DATE NOT NULL,
   precio_ejemplar NUMBER(6),
   dni_coleccion NUMBER(8) NOT NULL,
   od_coleccionado LONG,
    CONSTRAINT pk_ejc
      PRIMARY KEY (nombre_especie, nombre_zona,
        fecha_captura, hora_captura),
    CONSTRAINT fk_ejc_ejm
      FOREIGN KEY (nombre_especie, nombre_zona,
        fecha_captura, hora_captura)
      REFERENCES EjemplarMariposa(nombre_especie,
        nombre_zona, fecha_captura, hora_captura)
      ON DELETE CASCADE,
    CONSTRAINT fk_ejc_col
      FOREIGN KEY (dni_coleccion)
      REFERENCES Coleccion(dni) );
```

9.5.1.1. ANÁLISIS DEL ESQUEMA RELACIONAL

Le proponemos al lector, un pequeño ejercicio de análisis del esquema relacional propuesto. Para ello, repase con detenimiento el mismo y podrá observar que en algunas definiciones de tablas, se ha utilizado en la declaración de las claves foráneas la cláusula *ON DELETE CASCADE* y en otras no.

Recuerde que esta cláusula tiene como finalidad el que cuando se borre de la tabla a la que se hace referencia una tupla, en la tabla en la que se encuentra definida la referencia (con esta cláusula) se borrarán todas las tuplas en las que el valor de la clave foránea tenga el mismo valor que el del atributo clave de la tupla que ha sido borrada.

CUESTIÓN: *¿Cuál es la razón por la cual no se ha definido la clave foránea dni_coleccion con esta cláusula en la tabla EjemplarColeccionado?*

La respuesta es simple. En el caso en que se hubiera declarado esta cláusula en la definición de la clave foránea *dni_coleccion* en la tabla *EjemplarColeccionado*, implicaría que cuando se borrara una tupla de la tabla *Colección*, se borrarían automáticamente las tuplas correspondientes a todos los ejemplares de esa colección en la tabla *EjemplarColeccionado*, originando una incorrecta representación del problema.

La inconsistencia se basa en que se ha considerado una especialización total y exclusiva del tipo de entidad *EjemplarMariposa*, y en este caso ocurriría que existirían tuplas en la tabla *EjemplarMariposa* que no aparecerían en ninguna de las tablas que representan su especialización, a no ser que el procedimiento que borrara la tupla de la tabla *Colección* insertara previamente en la tabla *EjemplarLiberado* todas las tuplas de la tabla *EjemplarColeccionado* que se borrarían automáticamente si esta cláusula se hubiera declarado en la definición de la clave foránea *dni_coleccion*.

Razonamientos similares pueden presentarse para otros casos, tanto de este ejemplo, como de otros propuestos en la obra. La cláusula *ON DELETE CASCADE*, aunque es muy potente, su uso es delicado y requiere de un análisis profundo del problema a tratar

9.5.2. Manipulación de la Base de Datos

Cap09ej01.sql

Obtener la información correspondiente a las especies de mariposas existentes en la base de datos, las cuales han sido capturadas después de abril de 1999.

De la tabla Especie se obtienen los atributos nombre_especie y nombre_genero, y de la tabla Genero el atributo nombre_familia. El resultado de ambas consultas se combina en una reunión bajo la condición de que el atributo nombre_genero de ambas tablas tenga el mismo valor y además se cumpla que en la tabla EjemplarMariposa la fecha de captura sea posterior al 30/04/1999, ordenándose la salida alfabéticamente.

```
COLUMN Especie FORMAT A25
COLUMN Genero FORMAT A15
COLUMN Familia FORMAT A25
SELECT nombre_especie "Especie", Esp.nombre_genero "Genero",
       nombre_familia "Familia"
  FROM Especie Esp, Genero Gen
  WHERE Esp.nombre_genero = Gen.nombre_genero AND
        nombre_especie IN
     (SELECT DISTINCT nombre_especie
        FROM EjemplarMariposa
        WHERE fecha_captura >
              TO_DATE('30/04/1999','DD/MM/YYYY'))
  ORDER BY nombre_especie ASC;
```


Resultado

```
Especie                     Genero      Familia
--------------------------- ----------- ----------
Anthocharis Cardamines      Anthocharis Pieridae
Philosamia Cynthia          Cecropia    Saturnidae
```

Cap09ej02.sql

Obtener los diferentes nombres comunes y los nombres científicos de cada una de las especies cuyos ejemplares hayan sido capturados entre los años 1996 y 1997, todos ellos ordenados por el nombre de las zonas.

Se realiza una consulta de los atributos nombre_zona y nombre_comun de la tabla EspZonCom y otra sobre el atributo nombre_cientifico de la tabla Especie; se reúne el resultado de ambas y se restringe dicho resultado a que el nombre_especie debe corresponderse a ejemplares que hayan sido capturados dentro del período de tiempo comprendido entre 01/01/1996 y 31/12/1997. El resultado se ordena por nombre_zona y nombre_científico.

```
COLUMN nombre_zona FORMAT A16
COLUMN nombre_cientifico FORMAT A30
COLUMN nombre_comun FORMAT A20
SELECT nombre_zona, nombre_cientifico, nombre_comun
   FROM EspZonCom Ezc, Especie E
   WHERE E.nombre_especie = Ezc.nombre_especie AND
         E.nombre_especie IN
      (SELECT DISTINCT nombre_especie
         FROM EjemplarMariposa
         WHERE fecha_captura BETWEEN TO_DATE('01/01/1996',
               'DD/MM/YYYY') AND TO_DATE('31/12/1997',
               'DD/MM/YYYY'))
   ORDER BY nombre_zona, nombre_cientifico ASC;
```

Resultado

```
NOMBRE_ZONA  NOMBRE_CIENTIFICO        NOMBRE_COMUN
------------ ---------------------- ------------
Andalucía1   Philosamia Cynthia     Cynthia
Andalucía2   Philosamia Cynthia     Cynthia
Canarias3    Philosamia Cynthia     Cynthia
Extremadura1 Apatura Iris           Iris
```

Cap09ej03.sql

Obtener la mariposa más cara y la más barata de todas las capturadas, la persona que la capturó, y a qué colección pertenece.

De la tabla EjemplarColeccionado se obtiene toda la información del ejemplar de mariposa más caro, y el más barato, con excepción de la persona que capturó el ejemplar, para lo cual se realiza una reunión sobre la tabla EjemplarMariposa. Es necesario indicar la necesidad del uso de paréntesis que engloban las condiciones SELECT MAX ... OR SELECT MIN... para la correcta ejecución de la instrucción, debido a la mayor precedencia del operador AND sobre el OR, lo que generaría un resultado sin sentido sino se utilizan los paréntesis indicados.

```
COLUMN dni_captura FORMAT 99999999
COLUMN dni_coleccion FORMAT 99999999
COLUMN tamano FORMAT 999999
COLUMN sexo FORMAT A1
```

```
COLUMN nombre_especie FORMAT A20
COLUMN precio_ejemplar FORMAT 999999
SELECT dni_captura, dni_coleccion, tamano, sexo,
       EjMa.nombre_especie  precio_ejemplar
   FROM EjemplarMariposa EjMa, EjemplarColeccionado EjCo
   WHERE EjMa.nombre_especie = EjCo.nombre_especie
      AND EjMa.nombre_zona = EjCo.nombre_zona
      AND EjMa.fecha_captura = EjCo.fecha_captura
      AND EjMa.hora_captura = EjCo.hora_captura
      AND(precio_ejemplar =
            (SELECT MAX(precio_ejemplar)
                FROM EjemplarColeccionado)
          OR precio_ejemplar =
            (SELECT MIN(precio_ejemplar)
                FROM EjemplarColeccionado));
```

Resultado

DNI_CAPTURA	DNI_COLECCION	TAMANO	S	NOMBRE_ESPECIE	PRECIO_EJEMPLAR
73637486	67896790	122	M	Lymantria Dispar	4000
30405544	30405544	122	H	Hyloicus Pinastri	20500

Cap09ej04.sql

Obtener las familias de las mariposas cuyos ejemplares han sido liberados.

De la tabla EjemplarMariposa se obtiene el atributo nombre_especie de todos los ejemplares de mariposas cuyo atributo tipo_ejemplar tenga el valor 'L'; es decir, se han liberado. El resultado de esta subconsulta se utiliza para obtener de la tabla Especie el atributo nombre_genero de las mariposas de esa especie, y, de nuevo, el resultado de esta subconsulta se utiliza para obtener de la tabla Genero el atributo nombre_familia.

```
SELECT DISTINCT nombre_familia
   FROM Genero
   WHERE nombre_genero IN
      (SELECT DISTINCT nombre_genero
          FROM Especie
          WHERE nombre_especie IN
             (SELECT DISTINCT nombre_especie
                 FROM EjemplarMariposa
                 WHERE tipo_ejemplar = 'L' ));
```

Resultado

NOMBRE_FAMILIA
Lymantriidae
Ornecdidae
Saturnidae

Cap09ej05.sql

Obtener información sobre las familias, géneros y especies de mariposas que nunca hayan sido capturados.

De la tabla EjemplarMariposa se obtienen todas las especies de mariposas que han sido capturadas alguna vez para, negando el resultado de esta consulta gracias al operador NOT, conocer aquellas que no han sido capturadas. Gracias al resultado de esta subconsulta y a la reunión del resultado de la consulta sobre las tablas Genero y Especie se obtiene toda la información solicitada.


```
COLUMN nombre_familia FORMAT A20
COLUMN nombre_genero FORMAT A20
COLUMN nombre_especie FORMAT A25
SELECT nombre_familia, Genero.nombre_genero, nombre_especie
   FROM Genero, Especie
   WHERE Genero.nombre_genero = Especie.nombre_genero
         AND Especie.nombre_especie NOT IN
      (SELECT DISTINCT nombre_especie
          FROM EjemplarMariposa );
```

Resultado

NOMBRE_FAMILIA	NOMBRE_GENERO	NOMBRE_ESPECIE
Saturnidae	Saturdae	Saturnia Cal
Saturnidae	Cropidae	Saturnia Perosa
Nymphalis	Nympha	Mimas Tiliae

Cap09ej06.sql

Obtener, ordenadas por el número de capturas, información de las zonas geográficas en las que se hayan realizado más de 6 capturas.

De la tabla ZonLocPro se obtiene la información de las zonas geográficas y, a partir de la tabla EjemplarMariposa, se obtiene la información de la cantidad de ejemplares capturados. Dicha cantidad va a ser comparada en la clausula HAVING para sólo quedarse con aquellas zonas en las que se hayan capturado más de 6 ejemplares.

```
SELECT EjemplarMariposa.nombre_zona, localidad, provincia,
       COUNT(*) "CAPTURAS"
   FROM EjemplarMariposa, ZonLocPro
   WHERE EjemplarMariposa.nombre_zona = ZonLocPro.nombre_zona
   GROUP BY EjemplarMariposa.nombre_zona, localidad, provincia
     HAVING count(*)>6
   ORDER BY COUNT(*);
```

Resultado

NOMBRE_ZONA	LOCALIDAD	PROVINCIA	CAPTURAS
Extremadura2	Acehuche	Cáceres	7
Andalucía2	Cardeña	Córdoba	8

Cap09ej07.sql

Crear una vista para todas las colecciones iniciadas a partir de 1996, y que permita consultar simultáneamente los atributos dni, nombre, apellidos y precio estimado.

Para crear una vista, se recurre la instrucción DDL, CREATE VIEW, el cual define un nombre para la consulta encerrada dentro de su estructura. Para resolver la cuestión se proyectará sobre los campos dni, nombre, apellidos de la tabla Aficionado, y de la tabla Coleccion sobre el campo precio_estimado, realizando una reunión de las tablas implicadas y seleccionando bajo la restricción de la fecha impuesta.

```
CREATE OR REPLACE VIEW datos_coleccion AS
   SELECT Aficionado.dni, nombre, apellidos, precio_estimado
     FROM Aficionado, Coleccion
```

```
    WHERE Aficionado.dni = Coleccion.dni AND fecha_inicio >
          TO_DATE ('31/12/1995','DD/MM/YYYY')
  WITH CHECK OPTION
    CONSTRAINT ck_datos_coleccion;
```

Resultado

```
Vista creada.
```

Cap09ej08.sql

Realizar una consulta sobre la vista datos_coleccion.

Una vista es una tabla lógica, lo que implica que una consulta sobre una vista se realiza exactamente igual que sobre una tabla. De este modo, el obtener la información que encierra la vista es tan simple como realizar una operación SELECT sobre ella.

```
COLUMN dni FORMAT 99999999
COLUMN nombre FORMAT A10
COLUMN apellidos FORMAT A20
COLUMN precio_estimado FORMAT 9999999
SELECT *
  FROM datos_coleccion;
```

Resultado

```
       DNI NOMBRE     APELLIDOS            PRECIO_ESTIMADO
---------- ---------- -------------------- ---------------
  67896786 Pedro      Lara Bellido                   50000
```

Cap09ej09.sql

Obtener información de las personas que han capturado ejemplares antes del 5 de noviembre del 1980, pero que no tienen ninguna colección.

De la tabla Aficionado se obtiene la información correspondiente a los datos de la persona que cumple la condición de que el valor de su atributo dni no aparezca en ninguna tupla de la tabla Coleccion, con lo cual se garantiza que ha realizado alguna captura, pero que no es propietario de ninguna colección.

```
SELECT DISTINCT A.dni, A.nombre, A.apellidos
  FROM Aficionado A, EjemplarMariposa E
  WHERE A.dni=E.dni_captura  AND E.fecha_captura <
        TO_DATE('05/11/1980','DD/MM/YYYY')
        AND A.dni NOT IN
    (SELECT dni
     FROM Coleccion );
```

Resultado

```
       DNI NOMBRE     APELLIDOS
---------- ---------- --------------------
  34985032 Rafael     Pérez de la Torre
  45673487 Pedro José López Morilla
  67874647 Juan       López López
  73637486 Ángel Luis Pérez Casero
  89376376 José Juan  Romero Pérez
```


Cap09ej10.sql

Actualizar el valor estimado de las colecciones al valor real de la suma de los ejemplares que la componen.

De la tabla EjemplarColeccionado se obtiene la suma de los valores de los ejemplares y, con esta información, se actualiza el campo precio_estimado correspondiente a la tabla Coleccion.

```
UPDATE Coleccion
  SET precio_estimado =
    (SELECT SUM(precio_ejemplar)
     FROM EjemplarColeccionado
     WHERE EjemplarColeccionado.dni_coleccion =
           Coleccion.dni );
```

Resultado

```
15 filas actualizadas.
```

Cap09ejP1.sql

Actualizar el precio de los ejemplares de las colecciones de nuestra base de datos en un tanto por ciento dado y según el precio de los mismos, considerando la existencia de tres intervalos: menor de 7.000, de 7.000 a 10.000 y mayor de 10.000.

La actualización de los precios de los ejemplares se realizará recorriendo cada fila de la tabla Coleccion a través de un cursor y el uso de un bucle FOR en el cual se declara una variable de tipo ROWTYPE denominada fila_cursor que almacenará en cada iteración una fila de la tabla. Para acceder al campo precio_ejemplar, se hará igual que si se accede a un registro fila_cursor.precio_ejemplar. De esta forma se podrá comprobar en qué intervalo se encuentra este precio, y qué tipo de porcentaje se le aplica al actualizarlo.

```
CREATE OR REPLACE PROCEDURE act_ejemplar
  /* Se definen las variables de la interfaz */
  (p_primer_porcentaje NUMBER,
  p_segundo_porcentaje NUMBER,
  p_tercer_porcentaje NUMBER)
AS
  /* Se definen las variables internas de trabajo */
  incremento NUMBER;
  CURSOR eje_col IS
    SELECT *
      FROM EjemplarColeccionado;
  fila_cursor eje_col%ROWTYPE;
BEGIN
  /* Se recorren todas las tuplas de la tabla */
  FOR fila_cursor IN eje_col LOOP
  /* Se calcula el incremento dependiendo del precio de la
  mariposa a la que se ha accedido */
    IF fila_cursor.precio_ejemplar<7000 THEN
      incremento := (p_primer_porcentaje)/100;
    ELSIF fila_cursor.precio_ejemplar>=7000 AND
          fila_cursor.precio_ejemplar<10000 THEN
      incremento := (p_segundo_porcentaje)/100;
```

```
      ELSE
        incremento := (p_tercer_porcentaje)/100;
      END IF;
    /* Se actualiza el precio del ejemplar */
      UPDATE EjemplarColeccionado
        SET precio_ejemplar = fila_cursor.precio_ejemplar +
        ROUND(fila_cursor.precio_ejemplar*incremento)
          WHERE dni_coleccion=fila_cursor.dni_coleccion;
    END LOOP;
    /* Se actualiza el precio estimado de cada colección a la
    suma de los precios de cada ejemplar */
    UPDATE Coleccion
      SET precio_estimado =
        (SELECT SUM(precio_ejemplar)
          FROM EjemplarColeccionado
          WHERE EjemplarColeccionado.dni_coleccion =
                Coleccion.dni);
COMMIT;
    /* Se controlan los posibles errores de operación */
EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
    ROLLBACK;
END;
/

Prueba de ejecución
-- Se comprueba el estado inicial
SELECT nombre_especie, nombre_zona, fecha_captura,
        precio_ejemplar
    FROM EjemplarColeccionado;

-- Se ejecuta el procedimiento
       EXEC act_ejemplar (10,25,30);

-- Se comprueba el estado final
SELECT nombre_especie, nombre_zona, fecha_captura,
        precio_ejemplar
    FROM EjemplarColeccionado;
```

CAPÍTULO 10

VENTA Y CONSUMO DE CIGARRILLOS

10.1. ENUNCIADO DEL PROBLEMA

Se desea mantener información sobre las ventas de tabaco que son realizadas por las diferentes expendedurías autorizadas. Es importante para el distribuidor conocer información sobre las expendedurías, los pedidos que realizan y las ventas de las mismas. Asimismo, es importante conocer datos sobre las ventas detalladas para cada una de las clases de tabaco.

Los procesos de consulta más usuales que se realizarán harán referencia a las ventas y distribución de tabacos y el consumo que se realiza, bajo cualquier tipo de agrupación, de los tabacos vendidos por las expendedurías.

Otros supuestos semánticos que caracterizan el problema que va a ser tratado son:

SUPUESTO 1: Los estancos son abastecidos con un número de cigarrillos diferente que sólo depende de las órdenes de pedido y de la existencia de esos tipos. Por lo tanto, los estancos podrán vender cualquier tipo de cigarrillos de los que tengan existencias.

SUPUESTO 2: Los estancos tienen asignado un número de expendeduría que se puede repetir de una localidad a otra. Además, cada estanco tiene asignado un número de identificación fiscal que corresponde a la empresa como tal (o responsable de la

misma), así como un nombre, el cual puede ser el del responsable o no, que también puede repetirse incluso en la misma localidad.

SUPUESTO 3: Los fabricantes de cigarrillos tienen su sede principal en un país, aunque en un mismo país se pueden encontrar sedes de varios fabricantes.

SUPUESTO 4: Cada fabricante puede fabricar un número variable de marcas de cigarrillos, si bien una marca de cigarrillos, independientemente de su tipo, sólo puede ser fabricada por un único fabricante.

SUPUESTO 5: Para cada marca de cigarrillos se fabrican distintos tipos de ellos, según la existencia o no de filtro, el color de la hoja de tabaco y la cantidad de nicotina y alquitrán existente en los mismos.

SUPUESTO 6: De una misma marca pueden existir cigarrillos con filtro y sin filtro.

SUPUESTO 7: De una misma marca pueden existir cigarrillos de hoja negra y rubia.

SUPUESTO 8: De una misma marca pueden existir cigarrillos con contenido de contaminantes (nicotina y alquitrán) catalogados, según el fabricante, en: *Light*, *SuperLight* y *UltraLight*. Cuando un tipo no está catalogado en alguno de estos grupos (no ha tenido un tratamiento especial para eliminar parte de los contaminantes) se considera que es *Normal*. Los cigarrillos sin filtro son siempre catalogados en el tipo *Normal*[22].

SUPUESTO 9: De una misma marca de cigarrillos pueden existir cigarrillos mentolados y no mentolados. Los cigarrillos mentolados tienen siempre un contenido de contaminantes *Normal*.

SUPUESTO 10: No interesa conocer los clientes a los que se les venden los cigarrillos en los estancos.

SUPUESTO 11: Tanto para las compras como para las ventas sólo interesa conocer el total de ellas, de cada tipo de cigarrillos, realizadas por día.

10.2. MODELO CONCEPTUAL

A continuación vamos a pasar a extraer del enunciado y los supuestos introducidos en el problema los tipos de entidad e interrelaciones existentes en el mismo. Introduciremos en primer lugar los elementos del problema directamente deducibles del enunciado y las relaciones existentes entre los mismos. En un segundo paso, se discutirá la representación más apropiada para aquellos elementos del problema más complejos de representar.

[22] En la mayoría de las ocasiones el responsable de la eliminación en los cigarrillos de gran parte de los contaminantes es el filtro.

10.2.1. Los estancos, fabricantes y cigarrillos

Está claro que los tipos de entidad más significativos son aquellos que representan a los distintos tipos de cigarrillos y sus fabricantes, así como los estancos encargados de las compras y ventas de los mismos. Así, se tiene:

Tipo de entidad *Estancos*: el cual representa a ***"cada una de las expendedurías autorizadas para la venta y distribución de cigarrillos"***. Este tipo de entidad viene determinado, además de por el enunciado del problema, por los SUPUESTOS 1 y 2.

Se van a considerar los siguientes atributos para este tipo de entidad: *expendeduria#*, representando al número de expendeduría o licencia, ***nif_estanco***, ***nombre_estanco***, ***direccion_estanco***, ***localidad_estanco***, ***cp_estanco*** y ***provincia_estanco***, representando el número de identificación fiscal, el nombre de la expendeduría o de la empresa o del responsable y la dirección completa del estanco, respectivamente.

Sobre la base del SUPUESTO 2 es conveniente elegir el atributo ***nif_estanco*** como identificador principal de este tipo de entidad, siendo posible elegir los siguientes identificadores alternos:

- La agregación de los atributos ***expendeduria# + localidad_estanco***, debido a que en diferentes localidades pueden existir los mismos números de licencia. Esta clave sería válida siempre y cuando se consideren que no existen nombres de localidades iguales en distintas provincias. Como pueden existir, y de hecho existen, localidades con el mismo nombre en distintas provincias, es más adecuado elegir como clave alterna la agregación de los atributos ***expendeduria# + cp_estanco***, puesto que el código postal de la localidad será diferente para cada localidad.
- La agregación de los atributos ***expendeduria# + nombre_estanco***, siempre que se considere que no se dará el caso en que estancos con el mismo nombre (adviértase que puede ser el nombre del propietario del mismo) tengan asignados el mismo número de licencia.

Tipo de entidad *Fabricantes*: el cual representa a ***"cada una de las empresas dedicadas a la fabricación de cigarrillos en sus diferentes tipos"***. Este tipo de entidad está perfectamente definido en los SUPUESTOS 3 y 4 y, por tanto, sólo interesa considerar para el mismo los atributos ***nombre_fabricante*** y ***pais_fabricante***, representando el nombre de la empresa fabricante de marcas de cigarrillos y el país en el cual se encuentra su sede central respectivamente.

El atributo ***nombre_fabricante*** es elegido como identificador de este tipo de entidad, al considerarse que no existen dos fabricantes con el mismo nombre.

Tipo de entidad *Cigarrillos*: representando a ***"cada uno de los tipos de cigarrillos que son fabricados para su venta y consumo"***. Para una mejor representación del problema, este tipo de entidad va a representar a cada una de las ***cajetillas*** o unidades de venta de los cigarrillos. Así, como es sabido, los cigarrillos se venden

en paquetes, generalmente de 20 unidades, aunque existen marcas y tipos dentro de las marcas cuyo número de unidades de cigarrillos por cajetilla es diferente.

Son los SUPUESTOS 5-9 los que determinan la mayor parte de los atributos de este tipo de entidad, los cuales son:

- *marca*: identificando al nombre o marca que se le asigna a un tipo de mezcla de tabaco que se comercializa con independencia del formato y tipo de cigarrillo que se presenta al consumidor (SUPUESTO 5).
- *filtro*: identificando si los cigarrillos se presentan con filtro o sin él (SUPUESTO 6).
- *color:* identificando el tipo de hoja predominante que lleva la liga de tabacos de un tipo de cigarrillos, este atributo puede tomar los valores de *Rubio* o *Negro* (SUPUESTO 7).
- *clase*: representando el tratamiento especial que ha sufrido la liga de tabacos para alterar la cantidad de contaminantes que contiene la propia. Este atributo puede tomar los valores de: *Normal, Light,* y *UltraLight* (SUPUESTO 8).
- *mentol*: identificando si a la liga de tabacos de un determinado tipo de cigarrillos le es añadido mentol o cualquier otro producto que dé la característica de mentolado al cigarrillo.

Sobre la base de los mismos supuestos semánticos, está claro que el identificador de este tipo de entidad debe ser la agregación de todos y cada uno de los atributos antes mencionados. Otros atributos que son necesarios considerar para este tipo de entidad son:

- *nicotina*: representando la cantidad de nicotina por cigarrillo.
- *alquitran*: representando la cantidad de alquitrán por cigarrillo.
- *precio_costo*: el precio de costo de las unidades de venta (cajetillas) de los diferentes tipos de cigarrillos a las expendedurías.
- *precio_venta*: el precio de venta de las cajetillas de cigarrillos a los consumidores.

Los dos últimos atributos son impuestos en base al enunciado y a los SUPUESTOS 10 y 11 en los que se hace referencia a la intención del control económico de las compras y ventas de cigarrillos por los estancos.

Pero hay que tener en cuenta que al igual que los cigarrillos son vendidos por los estancos en unas unidades mínimas (las cajetillas), los estancos realizan sus compras al distribuidor, generalmente, en otras unidades mínimas denominadas *cartones*. Un cartón está formado por un conjunto de cajetillas, generalmente 10, y es en estas unidades, y sólo en estas, en las que los estancos pueden realizar los pedidos y compras de cigarrillos a los distribuidores.

Por ello es necesario considerar dos nuevos atributos en el tipo de entidad *Cigarrillos*:

- *carton*: representando el número de cajetillas que entran en un cartón de un tipo de cigarrillos determinado.
- *embalaje*: representando el número de cigarrillos que se empaquetan en cada cajetilla.

Para lo cual es necesario introducir un nuevo supuesto.

SUPUESTO 12: los estancos realizan las compras de cigarrillos por unidades de los mismos denominadas cartones. Un cartón está formado por un conjunto de cajetillas de cigarrillos, generalmente 10. Y una cajetilla de cigarrillos está formada por un conjunto de cigarrillos, generalmente 20.

De los tipos de entidad descritos se puede derivar fácilmente la relación existente entre los cigarrillos y los fabricantes, de la forma.

Tipo de interrelación *Cigarrillos/Fabricantes* **(C-F)**: que como se muestra en la figura 10.1 representa la relación existente entre los distintos tipos de cigarrillos y los fabricantes que los manufacturan. En este tipo de interrelación, el tipo de entidad *Fabricantes* participa con las cardinalidades *(1,1)* puesto que un tipo de cigarrillo es fabricado por un solo fabricante, mientras que el tipo de entidad *Cigarrillos* participa con las cardinalidades *(1,n)* puesto que un fabricante puede fabricar varios tipos. La cardinalidad mínima es uno en ambos casos debido a:

- Un tipo de cigarrillo siempre tiene asignado un fabricante (SUPUESTO 4).
- No interesa considerar aquellos fabricantes de los cuales, en el sistema, no se va a tener conocimiento de los cigarrillos que fabrica, puesto que los estancos no los van a comprar ni, por lo tanto, vender.

10.2.2. Las compras y las ventas

Las otras relaciones existentes en el problema son algo más difíciles de representar. Comenzaremos viendo en dónde está el problema para pasar, basándonos en las deficiencias de una primera representación, a proponer la más conveniente.

Como se indica en el enunciado y en algunos supuestos semánticos es interesante tener en cuenta las compras y las ventas de cigarrillos que son realizadas por los estancos. Existen, por tanto, dos clases de relaciones entre los estancos y los cigarrillos: una relación que representa las compras que realizan los estancos, y otra que representa las ventas de cigarrillos que realizan los estancos. Se trata en ambos casos de relaciones ternarias en las que interviene un tercer tipo de entidad, el tipo de entidad *Fecha*, para representar que las compras y ventas se realizan diariamente (como así indica el SUPUESTO 11).

Es conveniente descomponer estos dos tipos de relaciones ternarias en tipos de entidad débiles que representen las compras y ventas que realizan los estancos cada día de los diferentes tipos de cigarrillos. Aplicando, entonces, este proceso de refinamiento que permite clarificar el modelo, se tiene:

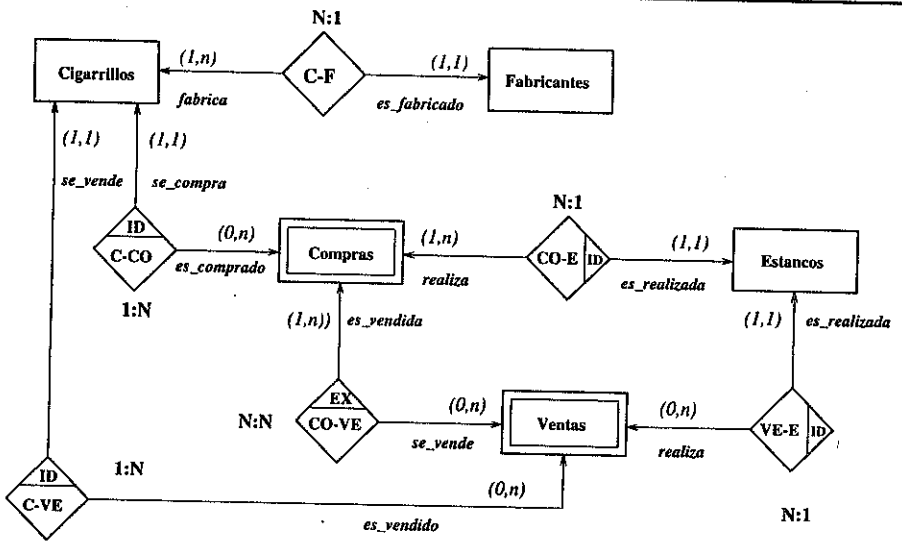


Figura 10.1 Las compras, las ventas y sus relaciones

Tipo de entidad *Compras*: representando al "*conjunto de pedidos o compras de los diferentes tipos de cigarrillos que realiza cada uno de los estancos al distribuidor*". También es obvio que se trata de un tipo de entidad débil:

- Por identificación con respecto al tipo de entidad *Cigarrillos*, con el fin de poder conocer el artículo que es comprado por el estanco.
- Por identificación con respecto al tipo de entidad *Estancos*, para identificar el estanco que realiza la compra.

Este hecho se ha representado en la figura 10.1 en la que se presentan dos tipos de interrelación en las que participa el tipo de entidad *Compras*:

Tipo de interrelación *Cigarrillos/Compras* **(C-CO)**: representando que en una compra se suministra uno y sólo un tipo de cigarrillos. El tipo de entidad *Cigarrillos* participa con cardinalidades *(1,1)*, mientras que un cigarrillo es objeto de cero o muchas compras en este tipo de interrelación.

Tipo de interrelación *Compras/Estanco* **(CO-E)**: representando que un estanco realiza una o varias compras. El tipo de entidad *Compras* participa con cardinalidades *(1,n)*, si no hubiera compras no habría ventas y, por lo tanto, no interesaría el estanco, mientras que una compra es realizada por un único estanco.

Se puede observar que el tipo de entidad *Compras* está representando cada una de las líneas que conforman las órdenes de compra de los estancos al distribuidor y, por tanto, vendrá caracterizado por los siguientes atributos:

- Los atributos identificadores del tipo de entidad *Cigarrillos* con el cual mantiene una relación de debilidad por identificación. Es decir, los atributos: *marca, filtro, color, clase* y *mentol*.
- El atributo identificador del tipo de entidad *Estancos*, con la cual mantiene una relación de debilidad por identificación. Es decir, el atributo: *nif_estanco*.

Además, será necesario considerar los siguientes atributos:

- *fecha_compra*: representando el día, mes y año en el cual se realiza la compra, pues es necesario conocer el total de compras por día de cada tipo de cigarrillos (SUPUESTO 11).
- *cantidad_compra*: el número de unidades de cigarrillos que se compran, es decir, el número de cartones de cajetillas que se compran al distribuidor, como así introduce el SUPUESTO 12.
- *precio_compra*: si van a considerarse las compras y ventas de cada estanco, se va a mantener información sobre los beneficios económicos de los mismos. Es usual en el mercado que en el transcurso de tiempo en que se realiza una compra de un artículo y la venta del mismo se produzca una modificación de los precios de coste del mismo. Por ello, es conveniente para cada compra realizada mantener información correspondiente al coste de la misma, con independencia de que se tenga información del coste actual (véase el tipo de entidad *Cigarrillos*) de ese artículo.

El identificador principal de este tipo de entidad será, por tanto, la agregación de los atributos: *marca, filtro, color, clase* y *mentol* (heredados del tipo de entidad *Cigarrillos*), *nif_estanco* (heredado del tipo de entidad *Estancos*), y *fecha_compra*, que identifica el día en que un estanco determinado realizó la compra de un determinado tipo de cigarrillos.

Tipo de entidad *Ventas*: representando al "*total de las ventas de cada tipo de cigarrillos que realiza cada uno de los estancos diariamente*", como así introduce el SUPUESTO 11.

Este tipo de entidad presenta algunos problemas en cuanto a su representación en el modelo conceptual. Veamos por qué:

- Es evidente que para identificar una determinada venta es necesario conocer: el estanco que la realiza, la fecha en que se realiza la venta y el tipo de cigarrillo a que corresponde la venta, al igual que ha ocurrido con el tipo de entidad *Compras*, y como se apunta en el SUPUESTO 11.
- Pero también es evidente que, para que un estanco pueda realizar una venta de un cigarrillo, ha tenido que comprarlo previamente y, es matemáticamente obligado, que debe haber existencias del mismo.

Se tiene, entonces, una serie de alternativas:

1. Si se considera que el tipo de entidad *Ventas* es débil por identificación con respecto a los tipos de entidad *Cigarrillos* y *Estancos*, será débil por existencia con respecto al tipo de entidad *Compras*.

   En este caso, al no ser débil por identificación con respecto al tipo de entidad *Compras*, no hay una herencia redundante de los identificadores de los cigarrillos y los estancos, pero sí representa que debe existir una compra para que se pueda realizar una venta, aunque se sigue teniendo el problema de no poder representar la necesidad de un stock de cigarrillos para poder realizar su venta (ver figura 10.1).

2. Si se considera que el tipo de entidad *Ventas* es débil por identificación con respecto al tipo de entidad *Compras* y, por tanto, con respecto a los tipos de entidad *Cigarrillos* y *Estancos*, puesto que el tipo de entidad *Compras* es débil por identificación con respecto a estos tipos de entidad.

   En este caso se siguen presentado los mismos problemas que en el anterior y, además, se está representando que una venta debe corresponder a una compra, mientras que una compra puede originar cero, una o más ventas.

La primera solución parece ser la más acertada y por ello se ha representado en la figura 10.1, pero sigue presentando serios problemas como son:

- Sigue sin poder representarse adecuadamente el hecho de que para que se realice una venta debe existir stock de la clase de cigarrillos que se vende.
- Como se observa en la figura 10.1, el tipo de interrelación *(CO-VE)* entre los tipos de entidad *Compras* y *Ventas* es *muchos a muchos*, para representar que no toda la cantidad de un determinado tipo de cigarrillos que se compra en *una compra* tiene que venderse en *una venta*, pues, como es lógico, una venta puede corresponder a una cantidad de cigarrillos de una clase correspondiente a una, menos de una o más de una compra.

Estas interpretaciones introducen confusión en el modelo propuesto. Por ello, es interesante refinar aún más el modelo conceptual introduciendo nuevos tipos de entidad.

10.2.3. Los almacenes de artículos de los estancos

Se ha descrito en la sección anterior los problemas que presenta el modelo inicialmente propuesto. Esto es debido a la no consideración de un tipo de entidad que debe estar presente en cualquier problema (a no ser que los requisitos del mismo digan lo contrario) que trate productos, de cualquier tipo, que sean adquiridos y vendidos por una empresa (los estancos en el caso que nos ocupa). Se trata de un tipo de entidad que represente a los artículos almacenados por la empresa; artículos que van a ser comprados y, posteriormente, vendidos. Así, se puede introducir este tipo de entidad de la forma siguiente:

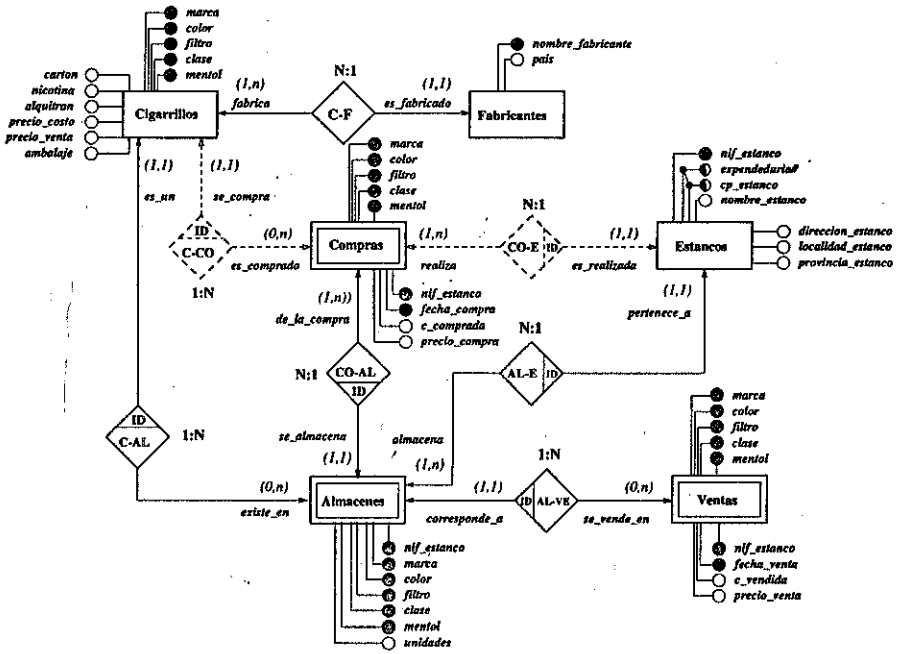


Figura 10.2 Tipo de entidad Almacenes *y sus relaciones*

Tipo de entidad *Almacenes*: el cual representa al "*conjunto de distintos tipos de cigarrillos que los estancos pueden adquirir, o han adquirido, y los cuales pueden vender a los clientes de los mismos*". Es decir, los artículos con que cada uno de los estancos realizan las operaciones de compra y venta. Las relaciones que mantiene este tipo de entidad con el resto de los tipos descritos se pueden observar en la figura 10.2 y describen lo siguiente:

Tipo de interrelación *Cigarrillos/Almacenes* **(C-AL)**: mediante el cual se representan los distintos tipos de cigarrillos que cada estanco puede tener almacenados y listos para su venta. Se trata de un tipo de interrelación *1:N* en el que el tipo de entidad *Almacenes* participa como un tipo de entidad débil por identificación.

Este tipo de interrelación permite representar los distintos tipos de cigarrillos que un estanco puede adquirir a su proveedor con independencia de que se haya realizado alguna compra de esos cigarrillos hasta la fecha, lo cual está permitiendo introducir una buena ampliación con respecto al modelo presentado en la figura 10.1.

En este sentido, el tipo de entidad *Almacenes* representa a un subconjunto del total de cigarrillos que el suministrador puede proporcionar y con los cuales cada estanco trabaja (compra y vende), con independencia de que en un momento dado tenga existencias de ellos.

Tipo de interrelación *Almacenes/Estancos* **(AL-E)**: mediante el cual se representan los diferentes tipos de cigarrillos con los que trabaja cada estanco. Como se puede observar en la figura 10.2, también aquí el tipo de entidad *Almacenes* es débil por identificación con respecto *Estancos*, pues se pueden tener los mismos tipos de cigarrillos en más de un estanco.

Sobre la base de estos dos tipos de interrelación, el tipo de entidad *Almacenes* tendrá como identificador la agregación de los atributos identificadores de los tipos de entidad *Cigarrillos* y *Estancos*. Además tendrá como atributo *unidades* para representar las unidades de cajetillas de cigarrillos que tienen en existencia para cada uno de los tipos de cigarrillos en cada uno de los estancos.

Es, en estos momentos, importante realizar algunas consideraciones con respecto al tipo de entidad *Compras*. Se pueden tomar los siguientes criterios:

1. Los estancos realizan compras de cigarrillos con independencia de que éstos hayan sido considerados en su almacén.

   Esta consideración determina la independencia entre los tipos de entidad *Compras* y *Almacenes*, dando lugar a que el primero participe con los tipos de entidad *Cigarrillos* y *Estancos* en tipos de interrelaciones débiles por identificación (véase figura 10.2, en líneas de trazos).

2. Los estancos realizan compras de cigarrillos en base al conocimiento de los mismos en su almacén.

   Esta consideración determina la existencia de un tipo de interrelación entre los tipos de entidad *Almacenes* y *Compras*, comportándose este último como débil por identificación (véase figura 10.2, en líneas continuas).

Como se puede apreciar fácilmente es este último criterio el más adecuado (ha sido el considerado en este ejemplo) y determina la dependencia de una compra con el conocimiento del almacén de la existencia del cigarrillo que se compra[23].

Tipo de interrelación *Almacenes/Ventas* **(AL-VE)**: representando las ventas que realizan los diferentes estancos de los productos existentes en su almacén. Es obvio que el tipo de entidad *Ventas* es débil por identificación con respecto al tipo de entidad *Almacenes*, pues son los artículos del almacén los que identifican a cada una de las ventas que se realizan cada día por cada estanco, y por transitividad con los tipos de entidad *Cigarrillos* y *Estancos*.

[23] Naturalmente, será necesario controlar en el proceso de compra que existe el cigarrillo que se desea comprar (no se ha borrado previamente) aunque todavía queden existencias en el almacén, y en el proceso de borrado que aunque se borre un cigarrillo, no se borre ni su entrada en el almacén ni sus compras y ventas, pues en caso contrario se perdería la información histórica correspondiente a las compras y ventas realizadas por los estancos.

10.3. MODELO RELACIONAL

Del esquema conceptual propuesto anteriormente y representado en la figura 10.2 puede derivarse el siguiente esquema relacional:

BASE DE DATOS DE TABACOS

Fabricantes	*(nombre_fabricante, pais)*
Estancos	*(nif_estanco, expendeduria#, cp_estanco, nombre_estanco, direccion_estanco, localidad_estanco, provincia_estanco)*
Cigarrillos	*(marca, filtro, color, clase, mentol, nicotina, alquitran,* **nombre_fabricante**, *precio_venta, precio_costo, carton, embalaje)*
Almacenes	**(nif_estanco, marca, filtro, color, clase, mentol**, *unidades)*
Compras	**(nif_estanco, marca, filtro, color, clase, mentol**, *fecha_compra, c_comprada, precio_compra)*
Ventas	**(nif_estanco, marca, filtro, color, clase, mentol**, *fecha_venta, c_vendida, precio_venta)*

El esquema relacional presentado se ha obtenido de la forma siguiente:

Por aplicación de la regla *RTECAR-1*: son obtenidas las relaciones *Fabricantes* y *Estancos*. Derivadas de los tipos de entidad del mismo nombre, los cuales participan con cardinalidades máximas uno en todos los tipos de interrelación en los que intervienen.

Por aplicación de la regla *RTECAR-3.1*: son obtenidas las tablas *Cigarrillos, Compras, Almacenes* y *Ventas*, derivadas de los tipos de interrelación *(C-F), (C-AL)* y *(AL-E), (CO-AL)* y *(AL-VE)*, respectivamente.

10.4. NORMALIZACIÓN DEL MODELO

Si no consideramos la dependencia funcional existente entre los atributos *cp_estanco, localidad_estanco* y *provincia_estanco*, todas las relaciones propuestas en el esquema relacional se encuentran normalizadas en *FNBC* a excepción de la relación *Cigarrillos*.

En la relación *Cigarrillos* se puede observar lo siguiente:

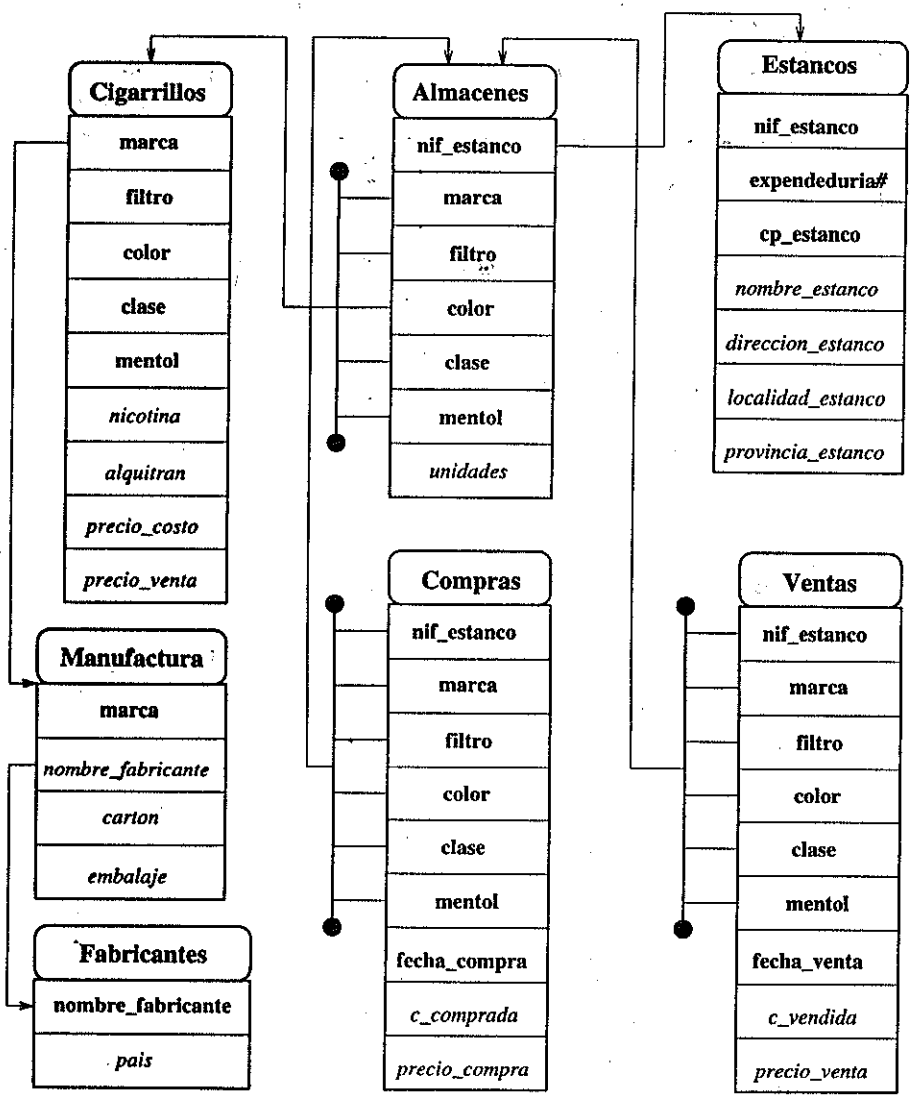


Figura 10.3 Diagrama relacional de la base de datos del problema

- Que existe una dependencia funcional completa entre la clave de esta relación y los atributos no primos *nicotina, alquitran, precio_venta* y *precio_costo*, como se muestra gráficamente a continuación:

marca, filtro, color, clase, mentol → *nicotina, alquitran, precio_costo, precio_venta*

- Es evidente considerar que todos los tipos de tabacos de una misma marca son manufacturados por un mismo fabricante, luego existe una dependencia funcional de la forma *Cigarrillos.marca* → *Cigarrillos.nombre_fabricante* que origina que la relación *Cigarrillos* no se encuentre en *FN2*. Por tanto, es necesario descomponer esta relación de la forma siguiente:

Manufactura (*marca*, **nombre_fabricante**)

Cigarrillos (**marca**, *filtro, color, clase, mentol, nicotina, alquitran, precio_venta, precio_costo, carton, embalaje*)

- Ahora es necesario realizar algunas consideraciones:

1. Se puede considerar que para una marca determinada los fabricantes manufacturan todos los tipos de cigarrillos de esa marca de la misma forma. Es decir, en todas las cajetillas existe el mismo número de cigarrillos, y todos los cartones contienen el mismo número de cajetillas, con independencia del tipo de cigarrillo de esa marca.

   En este caso, existe de nuevo una dependencia funcional incompleta de la forma:

   Cigarrillos.marca → *Cigarrillos.carton*

   Cigarrillos.marca → *Cigarrillos.embalaje*

   que es necesario evitar y, que por lo tanto, obliga a redefinir las relaciones anteriores quedando de la forma:

   Manufactura (*marca*, **nombre_fabricante**, *carton, embalaje*)

   Cigarrillos (**marca**, *filtro, color, clase, mentol, nicotina, alquitran, precio_venta, precio_costo*)

2. Que exista una independencia entre el número de cajetillas que contiene un cartón y el número de cigarrillos por cajetilla con respecto a la marca. En tal caso el fabricante para cada tipo de cigarrillos que manufactura puede comercializar éstos con distinto número de unidades por cajetilla y por cartón.

3. Se podría considerar cualquier otra dependencia entre estos atributos (*carton* y *cajetilla*) para realizar el estudio de las dependencias funcionales en esta relación. Dejamos al lector, como práctica, el planteamiento de otros tipos de dependencias.

Tomaremos el primero de los casos para el desarrollo del problema, quedando de esta manera el esquema relacional como se presenta en la figura 10.3.

10.5. CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

Bajo esta nueva visión más refinada del problema, el esquema relacional propuesto quedaría de la forma siguiente (ver figura 10.3):

BASE DE DATOS DE TABACOS	
Fabricantes	*(nombre_fabricante, pais)*
Manufactura	*(marca,* **nombre_fabricante**, *carton, embalaje)*
Estancos	*(nif_estanco, expendeduria#, cp_estanco, nombre_estanco, direccion_estanco, localidad_estanco, provincia_estanco)*
Cigarrillos	*(***marca**, *filtro, color, clase, mentol, nicotina, alquitran,* **nombre_fabricante**, *precio_venta, precio_costo)*
Almacenes	*(***nif_estanco, marca, filtro, color, clase, mentol**, *unidades)*
Compras	*(***nif_estanco, marca, filtro, color, clase, mentol**, *fecha_compra, c_comprada, precio_compra)*
Ventas	*(***nif_estanco, marca, filtro, color, clase, mentol**, *fecha_venta, c_vendida, precio_venta)*

10.5.1. Definición sintáctica de las tablas

```
   /* Se borran los objetos definidos en la base de datos */
DROP TABLE Compras;
DROP TABLE Ventas;
DROP TABLE Almacenes;
DROP TABLE Estancos;
DROP TABLE Cigarrillos;
DROP TABLE Manufactura;
DROP TABLE Fabricantes;
DROP VIEW ventas_del_7;
DROP SEQUENCE numero_del_estanco;

   /* Se crean las tablas del esquema propuesto */
CREATE TABLE Fabricantes (
   nombre_fabricante VARCHAR2(30) NOT NULL,
   pais VARCHAR2(15),
   CONSTRAINT pk_fab
     PRIMARY KEY (nombre_fabricante),
   CONSTRAINT ck_nfab
     CHECK (nombre_fabricante = INITCAP(nombre_fabricante)) );
```



```
CREATE TABLE Manufactura (
   marca VARCHAR2(20) NOT NULL,
   nombre_fabricante VARCHAR2(30),
   carton NUMBER(2) DEFAULT 10,
   embalaje NUMBER(2) DEFAULT 20,
   CONSTRAINT pk_man
     PRIMARY KEY (marca),
   CONSTRAINT fk_man_fab
     FOREIGN KEY (nombre_fabricante)
     REFERENCES Fabricantes(nombre_fabricante)
     ON DELETE CASCADE,
   CONSTRAINT ck_carton
     CHECK (carton > 0),
   CONSTRAINT ck_embalaje
     CHECK (embalaje > 0) );
CREATE TABLE Cigarrillos (
   marca VARCHAR2(20) NOT NULL,
   filtro CHAR(1) NOT NULL,
   color CHAR(1) NOT NULL,
   clase VARCHAR2(10) NOT NULL,
   mentol CHAR(1) NOT NULL,
   nicotina NUMBER(2,1),
   alquitran NUMBER(4,2),
   precio_venta NUMBER(3),
   precio_costo NUMBER(3),
   CONSTRAINT pk_cig
     PRIMARY KEY (marca, filtro, color, clase, mentol),
   CONSTRAINT fk_cig_man
     FOREIGN KEY (marca)
     REFERENCES Manufactura(marca)
     ON DELETE CASCADE,
   CONSTRAINT ck_filtro
     CHECK (filtro IN ('S', 'N')),
   CONSTRAINT ck_color
     CHECK (color IN ('R', 'N')),
   CONSTRAINT ck_clase
     CHECK (clase IN ('Light', 'SuperLight', 'UltraLight',
            'Normal')),
   CONSTRAINT ck_mentol
     CHECK (mentol IN ('S', 'N')),
   CONSTRAINT ck_pv
     CHECK (precio_venta > 0),
   CONSTRAINT ck_pc
     CHECK (precio_costo > 0) );
CREATE TABLE Estancos (
   nif_estanco VARCHAR2(12) NOT NULL,
   expendeduria NUMBER(4) NOT NULL,
   cp_estanco NUMBER(5) NOT NULL,
   nombre_estanco VARCHAR2(30) NOT NULL,
```

```
    direccion_estanco VARCHAR2(30),
    localidad_estanco VARCHAR2(15),
    provincia_estanco VARCHAR2(15),
    CONSTRAINT pk_est
      PRIMARY KEY (nif_estanco),
    CONSTRAINT sk_est
      UNIQUE (expendeduria, cp_estanco),
    CONSTRAINT ck_nest
      CHECK (nombre_estanco = INITCAP(nombre_estanco)),
    CONSTRAINT ck_nif
      CHECK (nif_estanco = UPPER(nif_estanco)) );
CREATE TABLE Almacenes (
    nif_estanco VARCHAR2(12) NOT NULL,
    marca VARCHAR2(20) NOT NULL,
    filtro CHAR(1) NOT NULL,
    color CHAR(1) NOT NULL,
    clase VARCHAR2(10) NOT NULL,
    mentol CHAR(1) NOT NULL,
    unidades NUMBER(4),
    CONSTRAINT pk_Alm
      PRIMARY KEY (nif_estanco, marca, filtro, color,
                   clase, mentol),
    CONSTRAINT fk_Alm_est
      FOREIGN KEY (nif_estanco)
      REFERENCES Estancos(nif_estanco)
      ON DELETE CASCADE,
    CONSTRAINT fk_alm_cig
      FOREIGN KEY (marca, filtro, color, clase, mentol)
      REFERENCES Cigarrillos(marca, filtro, color, clase, mentol)
      ON DELETE CASCADE,
    CONSTRAINT ck_uni
      CHECK (unidades > = 0) );
CREATE TABLE Compras (
    nif_estanco VARCHAR2(12) NOT NULL,
    marca VARCHAR2(20) NOT NULL,
    filtro CHAR(1) NOT NULL,
    color CHAR(1) NOT NULL,
    clase VARCHAR2(10) NOT NULL,
    mentol CHAR(1) NOT NULL,
    fecha_compra DATE NOT NULL,
    c_comprada NUMBER(3),
    precio_compra NUMBER(6),
    CONSTRAINT pk_com
      PRIMARY KEY (nif_estanco, marca, filtro, color,
                   clase, mentol, fecha_compra),
    CONSTRAINT fk_com_alm
      FOREIGN KEY (nif_estanco, marca, filtro, color,
                   clase, mentol)
      REFERENCES Almacenes(nif_estanco, marca, filtro,
```



```
                   color, clase, mentol)
      ON DELETE CASCADE,
    CONSTRAINT ck_c_comprada
      CHECK (c_comprada > 0),
    CONSTRAINT ck_precio_compra
      CHECK (precio_compra > 0) );
CREATE TABLE Ventas (
    nif_estanco VARCHAR2(12) NOT NULL,
    marca VARCHAR2(20) NOT NULL,
    filtro CHAR(1) NOT NULL,
    color CHAR(1) NOT NULL,
    clase VARCHAR2(10) NOT NULL,
    mentol CHAR(1) NOT NULL,
    fecha_venta DATE NOT NULL,
    c_vendida NUMBER(3),
    precio_venta NUMBER(6),
    CONSTRAINT pk_ven
      PRIMARY KEY (nif_estanco, marca, filtro, color, clase,
                   mentol, fecha_venta),
    CONSTRAINT fk_ven_alm
      FOREIGN KEY (nif_estanco, marca, filtro, color,
                   clase, mentol)
      REFERENCES Almacenes(nif_estanco, marca, filtro,
                   color, clase, mentol)
      ON DELETE CASCADE,
    CONSTRAINT ck_c_vendida
      CHECK (c_vendida > 0),
    CONSTRAINT ck_precio_venta
      CHECK (precio_venta > 0) );
```

10.5.2. Manipulación de la Base de Datos

Cap10ej01.sql

Obtener todas las marcas de cigarrillos extranjeros.

De la tabla Fabricantes se obtiene el atributo nombre_fabricante de todos los fabricantes cuyo atributo país tiene un valor distinto de 'España'. El resultado de esta subconsulta se utiliza para obtener el atributo marca de la tabla Manufactura de todos los cigarrillos manufacturados por las empresas anteriores. El resultado de la consulta aparece bajo la etiqueta 'Cigarrillos extranjeros' y ordenado alfabéticamante por el atributo marca.

```
SELECT DISTINCT marca "Cigarrillos extranjeros"
    FROM Manufactura
    WHERE nombre_fabricante IN
      (SELECT nombre_fabricante
        FROM Fabricantes
        WHERE pais != 'España')
    ORDER BY marca;
```

Resultado

```
Cigarrillos extranjeros
-----------------------
Camel
Marlboro
Chesterfield
Winston
```

Cap10ej02.sql

Obtener el valor de todas las compras realizadas por el estanco '11111' de la marca 'Camel' desde el año 1996 hasta la fecha.

*Mediante el operador SUM se obtiene el sumatorio del producto precio_compra * c_comprada de la tabla Compras en aquellas tuplas que cumplen la condición de que el atributo nif_estanco tenga el valor '11111', el atributo marca tenga el valor 'Camel', y el atributo fecha_compra almacene una fecha posterior o igual a la cadena '01/01/1996' convertida al tipo de dato fecha por la función TO_DATE.*

```
SELECT SUM (precio_compra * c_comprada)
       "Importe de las compras"
   FROM Compras
   WHERE nif_estanco = '11111' AND marca = 'Camel'
       AND fecha_compra >= TO_DATE('01/01/1996','DD/MM/YYYY');
```

Resultado

```
Importe de las compras
----------------------
                114610
```

Cap10ej03.sql

Obtener el valor de todas las ventas de la marca 'Ducados' que han realizado los estancos de la provincia de Madrid.

*De la tabla Estancos se obtiene el atributo nif_estanco de todos aquellos estancos cuyo atributo provincia_estanco tiene el valor 'Madrid'. El resultado de esta subconsulta se utiliza para obtener de la tabla Ventas el sumatorio del producto precio_venta * c_vendida de la marca 'Ducados'.*

```
SELECT SUM (precio_venta * c_vendida)
       "Ventas en Madrid de Ducados"
   FROM Ventas
   WHERE marca = 'Ducados' AND nif_estanco IN
     (SELECT nif_estanco
      FROM Estancos
      WHERE provincia_estanco = 'Madrid');
```

Resultado

```
Ventas en Madrid de Ducados
---------------------------
                     291500
```

Cap10ej04.sql

Obtener la marca de cigarrillos estadounidense que vende más cigarrillos.

De la tabla Fabricantes se obtiene el atributo nombre_fabricante de todas aquellas tuplas que cumplen la condición de que el país tiene el valor 'USA'. El resultado de esta subconsulta se utiliza para obtener la marca de aquellas tuplas de la tabla Manufactura que cumplen la condición de que el atributo nombre_fabricante tiene el valor anteriormente obtenido. Este resultado es utilizado a su vez para obtener el máximo de las cantidades vendidas por marca de la tabla Ventas que cumplen la condición de que su clave es igual al resultado devuelto por la subconsulta anterior. Por último, este valor numérico resultante se utiliza para obtener de la tabla Ventas los atributos marca y la cantidad vendida.


```
SELECT SUM(c_vendida) "Total Ventas",
       marca "Americano más vendido"
   FROM Ventas
   GROUP BY marca
   HAVING SUM(c_vendida) =
     (SELECT MAX(SUM(c_vendida))
      FROM Ventas
      WHERE marca IN
        (SELECT marca
           FROM Manufactura
           WHERE nombre_fabricante IN
              (SELECT nombre_fabricante
                 FROM Fabricantes
                 WHERE pais = 'USA'))
      GROUP BY marca);
```

Resultado

```
Total Ventas Americano más vendido
----------- --------------------
       6500 Camel
```

Cap10ej05.sql

Obtener los ingresos por ventas de la marca 'Winston' el 22 de Agosto de 1995.

*De la tabla Ventas se obtiene el sumatorio del producto precio_venta * c_vendida de todas las tuplas que cumplen la condición de que el atributo marca tenga el valor 'Winston' y fecha_venta tenga el valor equivalente a convertir la cadena '22/08/1995' al tipo de dato DATE.*

```
SELECT SUM (precio_venta * c_vendida) "Ingresos"
   FROM Ventas
   WHERE marca = 'Winston' AND fecha_venta =
         TO_DATE('22/08/1995','DD/MM/YYYY');
```

Resultado

```
Ingresos
---------
   820000
```

Cap10ej06.sql

Modificar la tabla Cigarrillos, de modo que se pueda almacenar en ella una medida de su calidad.

Se procede a añadir a la estructura de la tabla Cigarrillos un campo que se denominará calidad, que será de tipo NUMBER, compuesto por una parte entera de un dígito de longitud.

```
ALTER TABLE Cigarrillos
   ADD (calidad NUMBER(1));
```

Resultado

```
Tabla modificada.
```

Cap10ej07.sql

Modificar el tamaño del atributo unidades de la tabla Almacenes de 4 a 5 dígitos.

Se procede a alterar la estructura de la tabla Almacenes, modificando las características del campo unidades que pasa a ser de 5 dígitos de longitud.

```
ALTER TABLE Almacenes
   MODIFY (unidades NUMBER(5));
```

Resultado

```
Tabla modificada.
```

Cap10ej08.sql

Acelerar los accesos a las localidades y las provincias donde se encuentran los estancos.

Se procede a crear un índice sobre la tabla Estancos, concretamente sobre sus atributos localidad_estanco y provincia_estanco, el cual se denominará i_Estancos, y que garantizará un mayor desempeño en los accesos en los que aparezcan esos atributos en la cláusula de búsqueda.

```
   /* Se crea el índice. Se borra primero por si existe */
DROP Index i_Estancos;
CREATE INDEX i_Estancos
   ON Estancos (localidad_estanco, provincia_estanco)
   PCTFREE 5;
```

Resultado

```
Índice creado.
```

```
   /* Consultamos el índice creado */
SELECT TABLE_NAME "Tabla", PCT_FREE "PCT", INITIAL_EXTENT
"Inicial", NEXT_EXTENT "Siguiente",
       MAX_EXTENTS "Máximo", PCT_INCREASE "Incremento"
   FROM User_Indexes
   WHERE index_name = 'I_ESTANCOS';
```

Resultado

```
Tabla          PCT  Inicial  Siguiente  Máximo  Incremento
-------------- ---  -------  ---------  ------  ----------
ESTANCOS         5    57344      57344     121           1.
```

Cap10ej09.sql

Impedir que en la tabla Estancos se puedan almacenar estancos cuyo NIF tenga menos de doce caracteres.

Se procede a alterar el esquema de la tabla Estancos mediante un nuevo CONSTRAINT encargado de representar la nueva restricción impuesta.

```
ALTER TABLE Estancos
   ADD (CONSTRAINT ck_nif1
     CHECK(LENGTH(nif_estanco)=12));
```

Resultado

```
  CHECK(LENGTH(nif_estanco)=12))
                             *
  ERROR at line 3:
  ORA-02293: no se ha podido validar (TABACOS.CK_NIFL) - compruebe la
             restricción violada
```



```
   /* Como se puede observar, se produce un error debido a que
   el atributo nif_estanco está referenciado por otras tablas
   como clave foránea produciéndose una inconsistencia. */
```

Cap10ej10.sql

Obtener el nombre y el NIF de todos los estancos de Madrid que no venden cigarrillos Winston con mentol, pero sí Celtas sin filtro.

En dos consultas distintas de la tabla Almacenes se obtiene el atributo nombre_fabricante de todas aquellas tuplas en las que:

- *marca tiene el valor 'Winston' y mentol tiene el valor 'S'.*
- *marca tiene el valor 'Celtas' y filtro tiene el valor 'N'.*

El resultado de la primera consulta se niega mediante el operador NOT, con lo cual se ha obtenido el nif_estanco de todos los estancos que no venden Winston con mentol, pero sí Celtas sin filtro. Este resultado es utilizado por una consulta sobre la tabla Estancos, en la que se obtiene la proyección solicitada para los estancos de Madrid.

```
SELECT nif_estanco, nombre_estanco
   FROM Estancos
WHERE provincia_estanco='Madrid'AND nif_estanco NOT IN
          (SELECT DISTINCT nif_estanco
              FROM Almacenes
              WHERE marca='Winston'AND mentol='S')
      AND nif_estanco IN
          (SELECT DISTINCT nif_estanco
              FROM Almacenes
              WHERE marca='Celtas'AND filtro='N');
```

Resultado

```
NIF_ESTANCO  NOMBRE_ESTANCO
-----------  ----------------
11111        La Pajarita
```

Cap10ejP1.sql

Crear un procedimiento PL/SQL que permita obtener las marcas de cigarrillos para los cuales se hayan vendido menos de una determinada cantidad (primer parámetro del procedimiento) eliminando todas estas marcas en el almacén de un determinado estanco, cuyo NIF constituye el segundo parámetro del procedimiento.

Se obtiene la cantidad de ventas de cada una de las marcas de cigarrillos realizada por cada estanco, recorriendo la tabla Ventas a través de un cursor y un bucle FOR en el cual se utiliza la variable "fila_del_cursor" de tipo ROWTYPE. Si el total de ventas de una determinada marca para el estanco seleccionado (parámetro "nif_del_estanco") es inferior al parámetro "minimo", esta marca es eliminada en la tabla almacén para dicho estanco.

```
CREATE OR REPLACE PROCEDURE menos_vendidas
   /* Se definen las variables de la interfaz */
   (nif_del_estanco estancos.nif_estanco%TYPE,
    minimo NUMBER )
AS
   CURSOR ventas_menores IS
```

```
    SELECT  marca, SUM(c_vendida) total_ventas
       FROM Ventas
       WHERE nif_estanco = nif_del_estanco
       GROUP BY marca;
BEGIN
    /* Se recorre el cursor declarado */
    FOR fila_del_cursor IN ventas_menores LOOP
    /* Se eliminan las tuplas que cumplen las condiciones
    declaradas en los parámetros de la interfaz */
      IF fila_del_cursor.total_ventas < minimo THEN
        DELETE FROM Almacenes
          WHERE nif_estanco = nif_del_estanco
                AND marca = fila_del_cursor.marca;
      END IF;
    END LOOP;
COMMIT;
    /* Se controlan los  errores */
EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
    ROLLBACK;
END;
/
```

Prueba de ejecución

```
-- Se comprueba el estado actual
SELECT nif_estanco,marca, AVG(c_vendida) promedio_ventas
   FROM Ventas Where nif_estanco LIKE '11112'
   GROUP BY marca,nif_estanco;

-- Se ejecuta el procedimiento
     EXEC menos_vendidas (11112,300);

-- Se comprueba el estado final
SELECT  nif_estanco,marca, AVG(c_vendida) promedio_ventas
   FROM Ventas Where nif_estanco LIKE '11112'
   GROUP BY marca,nif_estanco;
```

CAPÍTULO 11

PESCA DEPORTIVA

11.1. ENUNCIADO DEL PROBLEMA

Una asociación deportiva dedicada a la práctica de la pesca en agua dulce desea informatizar su gestión de socios y eventos o concursos. La asociación, denominada "Fishermen Team", cuenta con un conjunto de afiliados que participan en competiciones deportivas de pesca que pueden ser organizadas por ella misma, por otras organizaciones del mismo tipo, o por organizaciones privadas o públicas.

A Fishermen Team le interesa conocer información de sus afiliados en cada una de las competiciones en las que participan, las capturas realizadas, posición que obtuvo en la competición, premios obtenidos, etc. Además, también le interesa conocer otras capturas que pueden realizar sus afiliados fuera de las competiciones, así como información general sobre los peces que existen en los cauces de agua dulce de nuestro territorio nacional para proporcionársela a sus afiliados.

En este problema es necesario considerar, además, los siguientes supuestos semánticos:

SUPUESTO 1: *Fishermen Team* sólo considerará información de los afiliados a la asociación, los cuales tienen asignado un número de socio, y sobre los cuales se desea mantener toda su información personal.

SUPUESTO 2: A la asociación le interesa mantener información sobre todos los eventos deportivos relacionados con la captura de peces en los que participen sus asociados o no.

SUPUESTO 3: Cada evento deportivo tiene un nombre único, se realiza en un lugar y se puede capturar un número de peces correspondientes a una serie de especies predeterminado a priori.

SUPUESTO 4: A la asociación le interesa mantener información sobre los lugares donde sus afiliados pueden practicar o han practicado la pesca.

SUPUESTO 5: Cada lugar se encuentra ubicado en un cauce fluvial, lago o pantano.

SUPUESTO 6: Para cada lugar existen unas vedas en cuanto a los peces que se pueden capturar, talla mínima, número máximo de ejemplares, coste adicional del uso del lugar, licencia necesaria, etc. Además, interesa mantener información de las mejores capturas obtenidas en ese lugar y la media de capturas por tipo de pez.

SUPUESTO 7: Cada lugar pertenece a una única comunidad autónoma y, por tanto, es necesario el permiso de pesca de la misma para poder practicar la pesca en ese lugar.

SUPUESTO 8: A la asociación le interesa mantener información sobre las capturas realizadas por sus afiliados. Las capturas pueden ser realizadas en competiciones o no.

SUPUESTO 9: Para cada captura se requiere información correspondiente a las características del pez capturado, la fecha, hora y lugar de la misma, y a ser posible una foto en la que se pueda comprobar, mediante referencias a objetos de tamaño conocido, el tamaño de la pieza capturada.

SUPUESTO 10: Si una captura no se realiza en un evento deportivo, para que ésta sea considerada tiene que estar avalada por dos afiliados de la asociación.

SUPUESTO 11: En las participaciones de los afiliados en los eventos deportivos, a la asociación le interesa mantener información del resultado de las mismas: posición que ocupó, capturas realizadas, puntos obtenidos, trofeos, etc.

11.2. MODELO CONCEPTUAL

Los aficionados al deporte de la pesca están muy familiarizados con los objetos presentes en el problema y, para aquellos lectores que hasta ahora no han conocido las virtudes de este deporte, podrán extraer de igual forma los objetos más importantes en el problema, y comprender sin dificultad la representación del mismo propuesta en la figura 11.1.

11.2.1. Los cauces fluviales, lugares de pesca y peces

El tipo de pesca que interesa a los afiliados de esta asociación es el de agua dulce, con independencia de su modalidad, por lo cual y como introducen los SUPUESTOS 4-7, a la asociación le interesa mantener información de los cauces fluviales, los lugares de pesca en los mismos y los peces que pueden ser capturados, o ya han sido capturados, en estos territorios piscícolas. Surgen, por tanto, los siguientes tipos de entidad:

Tipo de entidad *Cauces*: representando a "*las corrientes fluviales existentes en el territorio nacional y en las cuales existen lugares en los que se puede practicar la pesca deportiva*" (SUPUESTO 5).

Para este tipo de entidad se van a considerar los atributos: *cauce*, como identificador del tipo de entidad y representando el nombre del cauce fluvial, y *od_cauce* para representar cualquier otra información de interés acerca del mismo (mucha seguramente para una asociación de pesca, así como para sus pescadores).

Tipo de Entidad *Lugares*: representando a "*las diferentes zonas de pesca correspondientes a un cauce fluvial*". Un lugar representa las distintas zonas acotadas, o no, en las que se puede practicar la pesca (porque esté permitido o, simplemente, porque existan peces para ello) sobre la cual desea estar informada la asociación, o simplemente porque en él se realizan competiciones deportivas (SUPUESTO 3).

Para este tipo de entidad se van a considerar los atributos: *lugar*, como identificador del mismo y representando el nombre de ese lugar, *comunidad*, representando el nombre de la comunidad autónoma a la que pertenece el lugar (SUPUESTO 7), y *od_lugar*, representando a cualquier otra información de interés.

Para considerar al atributo *lugar* como identificador de este tipo de entidad es necesario introducir el siguiente supuesto:

SUPUESTO 12: Cada lugar de pesca tiene un nombre único, independientemente del cauce fluvial al cual pertenezca.

Tipo de entidad *Peces*: representando a "*los distintos tipos de peces que están presentes en los lugares de pesca y que son, o pueden ser, capturados por los afiliados de la asociación*". Para este tipo de entidad se van a considerar los siguientes atributos[24]:

- *pez*: representando el nombre o tipo de pez. Este atributo podrá tomar el valor del nombre común, nombre científico, o simplemente el nombre por el cual los afiliados de la asociación reconocen a una determinada especie

[24] Naturalmente, en un caso real el número de atributos a considerar en éste y otros tipos de entidad que existen en el problema sería mucho mayor. Nuestro objetivo es considerar, tanto en éste como en otros ejercicios, un número de atributos mínimo que sin introducir una complejidad excesiva, permita clarificar el proceso de análisis y diseño de la base de datos.

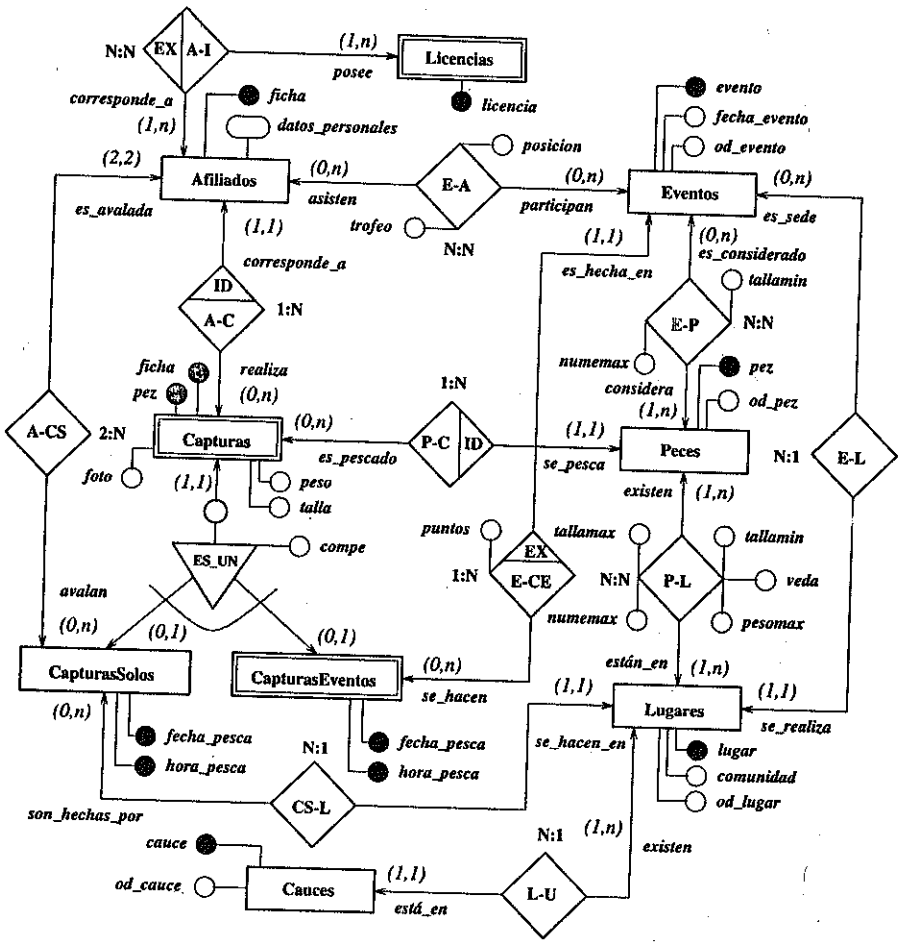


Figura 11.1 Esquema E-R del problema de la asociación de pesca

piscícola. Consideraremos este atributo como identificador del tipo de entidad *Peces*.

- *od_pez*: para representar cualquier otra información de interés referente a este tipo de entidad.

Estos tres tipos de entidad se encuentran relacionados por medio de dos tipos de interrelación. El tipo de interrelación *(L-U)* entre los tipos de entidad *Lugares* y *Cauces*, que representa el hecho de que un lugar pertenece a un único cauce fluvial (SUPUESTO 5) —el tipo de entidad *Cauces* participa con las cardinalidades *(1,1)*—, y

que en un cauce pueden existir al menos uno, y posiblemente muchos lugares de pesca —el tipo de entidad *Lugares* participa con las cardinalidades *(1,n)*[25]—.

Por otro lado, los tipos de entidad *Lugares* y *Peces* participan en un tipo de interrelación *N:N*, denominado *(P-L)*, de forma que en un lugar pueden existir uno o más tipos de peces, y un tipo de pez puede estar presente en uno o muchos lugares.

Este tipo de interrelación viene caracterizado por los siguientes atributos según introduce el SUPUESTO 6:

- *tallamax*: representando el tamaño más grande de cada clase de pez capturado en ese lugar (por afiliados o no).
- *pesomax*: representando el peso máximo de cada clase de pez capturado en ese lugar.
- *numemax*: representando el número máximo de ejemplares que pueden ser capturados en cada lugar de cada clase de pez.
- *tallamin*: representando el tamaño mínimo de cada tipo de pez que está permitido capturar en cada lugar.
- *veda*: representando la veda existente en cada lugar para cada tipo de pez. Consideraremos, para simplificar el problema, que es un atributo informativo, puesto que en realidad la información correspondiente a las vedas en cada lugar es bastante más compleja de representar pues, entre otras razones, es dependiente de las fechas del año.

11.2.2. Los afiliados y las competiciones deportivas

Los afiliados son los objetos principales del problema planteado, así como los eventos deportivos en los cuales ellos pueden participar. Entonces, se puede definir los siguientes tipos de entidad:

Tipo de entidad *Afiliados*: representando a *"los diferentes socios de Fishermen Team"*. Como se observa en la figura 11.1, para este tipo de entidad se han considerado los atributos: *ficha*, como identificador y representando el número de socio, y *datos_personales*, un atributo compuesto y formado por un conjunto de atributos que representan todos los datos personales de los afiliados (nombre, apellidos, dirección, etc.). Este atributo será descompuesto directamente en el proceso de derivación del esquema relacional.

Tipo de entidad *Eventos*: representando a *"las diferentes competiciones deportivas de pesca sobre las cuales desea mantener información la asociación"*. Este tipo de entidad viene caracterizado por los atributos: *evento*, representando el nombre del evento deportivo y, según el SUPUESTO 3, sirviendo de identificador, *fecha_evento*, y *od_evento*, representando la fecha en que se celebra el mismo, y

[25] La cardinalidad mínima es uno, pues no se consideran los cauces en los que no existen lugares de pesca. A la asociación le interesan los lugares de pesca y no los cauces fluviales.

cualquier otra información de interés, como podría ser el precio de inscripción, número de participantes, su carácter nacional o internacional, etc.

Estos dos tipos de entidad están relacionados mediante el tipo de interrelación *(E-A)*, de forma que un afiliado puede participar en *(0,n)* eventos, mientras que en un evento pueden no participar afiliados o participar varios de ellos.

Se trata, por tanto, de un tipo de interrelación *N:N* que, como muestra la figura 11.1, viene caracterizado por los atributos *posicion* y *trofeo* que representan el resultado de las participaciones de los afiliados en los eventos, como así introduce el SUPUESTO 11.

Por otro lado, el tipo de entidad *Eventos* participa en un tipo de interrelación *N:1* con el tipo de entidad *Lugares*, representando (SUPUESTO 3) que un evento deportivo se realiza únicamente en un lugar, mientras que en un lugar se pueden realizar varios eventos deportivos de pesca.

Además, el tipo de entidad *Afiliados* incorpora el atributo múltiple *licencia*, representando las clases de licencias deportivas que posee cada uno de los afiliados. En este momento es necesario recordar que, según se describió en el Capítulo 5, los atributos múltiples deben ser eliminados, sobre la base de la aplicación de la regla *PRTECAR-1*, en tipos de entidad débiles por existencia o identificación para poder realizar la derivación del esquema conceptual.

Como se muestra en la figura 11.1 este proceso se ha realizado directamente considerándose un tipo de entidad débil por existencia con respecto al tipo de entidad *Afiliados*. Este tipo de entidad se ha denominado *Licencias* y viene caracterizado por un único atributo, denominado *licencia*, que representa a los distintos tipos de licencias que poseen los afiliados de la asociación.

11.2.3. Las capturas de peces

Los SUPUESTOS 8-10 introducen la necesidad de representar las capturas que realizan los afiliados de la asociación tanto en competiciones como fuera de ellas. Se puede considerar un tipo de entidad *Capturas*, y especializarlo de forma total y exclusiva, como se muestra en la figura 11.1, en dos subtipos que representen las diferentes clases de capturas que deben considerarse.

El tipo de entidad *Capturas* representa a cada uno de los peces capturados por los afiliados y:

- como los peces vienen identificados por el nombre del pez (el atributo *pez*),
- un afiliado puede capturar más de un pez del mismo tipo, y
- un mismo tipo de pez puede ser capturado por más de un afiliado.

se trata de un tipo de entidad débil por identificación con respecto a los tipos de entidad *Afiliados* y *Peces*, heredando, por tanto, sus atributos identificadores.

Además, este tipo de entidad incorpora los atributos *peso, talla* y *foto*, como así introduce el SUPUESTO 9, representando las características de los peces capturados.

Pero estos atributos no son todavía suficientes para servir de identificadores para este tipo de entidad, debido a que un afiliado puede capturar más de un pez de un mismo tipo con la misma talla y peso (es usual en algunas zonas). Por ello, como se muestra en la misma figura, los subtipos de entidad incorporan otros atributos. Así, se tiene:

Tipo de entidad *CapturasSolos*: representando a "*las capturas que realizan los afiliados fuera de las competiciones deportivas*". Este tipo de entidad incorpora los atributos *fecha_pesca* y *hora_pesca* que, agregados con los atributos identificadores del tipo de entidad *Capturas*, desempeñan el papel de identificadores, puesto que se considera que no se realiza, al mismo tiempo, una captura igual por un mismo afiliado.

Tipo de entidad *CapturasEventos*: representando a "*las capturas realizadas en las competiciones deportivas de pesca*". Este tipo de entidad tiene el mismo comportamiento que el anterior. Aunque existen dos posibilidades a considerar para este tipo de entidad:

1. La considerada; es decir, que el tipo de entidad *CapturasEventos* es similar al tipo de entidad *CapturasSolos* y, por tanto, incorpora los atributos *fecha_pesca* y *hora_pesca*, aunque participa en un tipo de interrelación débil por existencia con el tipo de entidad *Eventos*, puesto que si no existe un evento no se pueden realizar este tipo de capturas[26].
2. Considerar que el tipo de entidad *CapturasEventos* es débil por identificación con respecto al tipo de entidad *Eventos* y, por tanto, hereda su identificador, además de los identificadores del tipo de entidad *Capturas* y el atributo *hora_pesca*, formando también parte del identificador de este subtipo de entidad.

El no considerar la segunda opción es debido a que entonces se estaría representando que todas las capturas que se realizan en un evento se han realizado en la fecha en que se realiza el mismo (el atributo *fecha_evento*, puesto que la única información acerca de la fecha vendría determinada a partir del tipo de interrelación *(E-CE)*), mientras que con la opción primera (la considerada) se pueden realizar capturas en eventos deportivos en otras fechas diferentes, y esto es lo que ocurre realmente puesto que los eventos deportivos pueden durar más de un día.

[26] Los atributos *fecha_pesca* y *hora_pesca* se han considerado en los subtipos de entidad y no en el supertipo *Capturas* únicamente para poder realizar este análisis, en realidad, estos atributos deberían propagarse al supertipo de entidad.

Naturalmente, será conveniente controlar que el valor del atributo *fecha_pesca* del tipo de entidad *CapturasEventos* no tiene un valor inconsistente con el del atributo *fecha_evento* en el correspondiente esquema relacional que se derive.

Por otra parte, el tipo de entidad *CapturasSolos* participa en un tipo de interrelación *2:N* con el tipo de entidad *Afiliados* para representar la imposición del SUPUESTO 10, referente al aval necesario de dos afiliados para considerar este tipo de capturas.

El tipo de entidad *CapturasSolos* está también relacionado con el tipo de entidad *Lugares* mediante el tipo de interrelación *(CS-L)*, para representar el lugar en el que se realiza la captura.

Obsérvese que no es necesario relacionar al tipo de entidad *CapturasEventos* con el tipo de entidad *Lugares*, puesto que esta información se conoce por la relación existente entre el primero y el tipo de entidad *Eventos*, el cual sí está relacionado con el tipo de entidad *Lugares*.

Se ha considerado que el tipo de interrelación *(E-CE)* entre los tipos de entidad *CapturasEventos* y *Eventos* está caracterizado por el atributo *puntos*, representando los puntos asignados en la competición a esa captura (SUPUESTO 11).

11.3. MODELO RELACIONAL

Se puede observar que no se trata de un problema muy complejo y, por lo tanto, la derivación del siguiente esquema relacional es medianamente simple:

BASE DE DATOS DE LA ASOCIACIÓN DE PESCA DEPORTIVA

Afiliados	*(ficha, nombre_afiliado, apellidos_afiliado, direccion_afiliado, telf_afiliado, sexo_afiliado, nacimiento_afiliado, od_afiliado)*
Licencias	*(licencia)*
Permisos	**(ficha, licencia)**
Cauces	*(cauce, od_cauce)*
Lugares	*(lugar, comunidad, od_lugar,* **cauce***)*
Eventos	*(evento, fecha_evento,* **lugar**, *od_evento)*
Peces	*(pez, od_pez)*
Participaciones	**(ficha, evento**, *posicion, trofeo)*
CapturasSolos	**(ficha, pez**, *fecha_pesca, hora_pesca, peso, talla, foto,* **lugar, aval1, aval2***)*

CapturasEventos	**(ficha, pez**, *fecha_pesca, hora_pesca, peso, talla, foto,* **evento**, *puntos)*
Faunas	**(lugar, pez**, *tallamax, pesomax, numemax, tallamin, veda)*
Concursos	**(evento, pez**, *tallamin, numemax)*

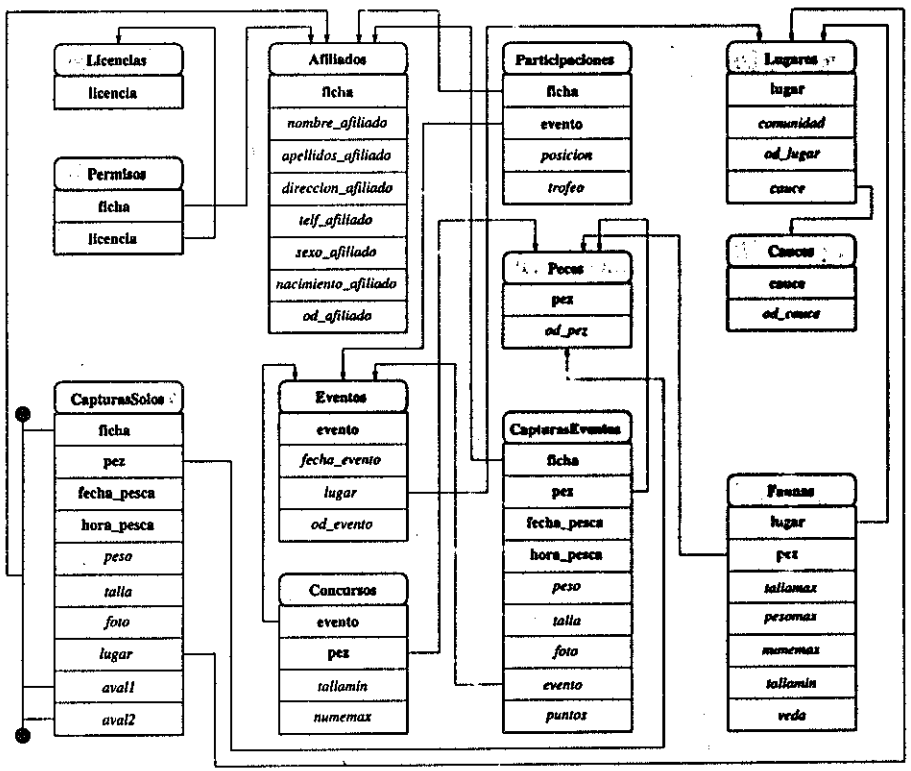


Figura 11.2 Diagrama relacional de la base de datos del problema

El esquema relacional propuesto ha sido derivado de la forma siguiente:

- Por aplicación de la regla *RTECAR-1* se derivan las tablas *Afiliados*, *Cauces* y *Peces* de los tipos de entidad con el mismo nombre, respectivamente.

  Como el tipo de entidad *Licencias* no tiene más atributos que su clave (recuérdese que se obtuvo por aplicación de la regla *PRTECAR-1*), la tabla correspondiente que se deriva podría eliminarse del esquema generado.

- Por aplicación de la regla *RTECAR-3.1*,
  - Al tipo de interrelación *(L-U)*, la tabla *Lugares*.

- Al tipo de interrelación (*E-L*), la tabla *Eventos*.
- Al tipo de interrelación (*A-CS*), la tabla *CapturasSolos*.
- Al tipo de interrelación (*E-CE*), la tabla *CapturasEventos*.
- Al tipo de interrelación (*E-P*), la tabla *Concursos*.

- Por aplicación de la regla *PRTECAR-3*, se elimina el supertipo *Capturas*, y se generan los subtipos *CapturasSolos* y *CapturasEventos*.
- Por aplicación de la regla *RTECAR-4*,
  - Al tipo de interrelación (*A-I*), la tabla *Permisos*.
  - Al tipo de interrelación (*E-A*), la tabla *Participaciones*.
  - Al tipo de interrelación (*P-L*), la tabla *Faunas*.

Todas las tablas se encuentran normalizadas en la forma *FNBC*, por lo que se puede generar el esquema relacional correspondiente y cuyo diagrama se presenta en la figura 11.2[27].

11.4. CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

11.4.1. Definición sintáctica de las tablas

Para este ejemplo se ha utilizado un nuevo método de representación del esquema relacional que permite las nuevas versiones de *Oracle* (8i y sucesivas). Este método de representación está basado en el paradigma de objetos mediante el cual las tablas son consideradas representaciones de unos tipos de datos estructurados.

Como se observa, inicialmente se definen los tipos de objetos que representan los elementos de información que intervienen en el dominio del problema, y posteriormente se definen las tablas del esquema relacional a partir de los tipos de objetos previamente declarados. En el ejemplo, se han definido únicamente tipos para las estructuras de las tablas, las cuales posteriormente serán definidas en función de estos tipos, si bien una tabla puede ser definida posteriormente haciendo uso de uno o más tipos definidos.

El lector observará que la definición del esquema, además de la definición de las tablas en función de tipos previamente definidos, incorpora otras modificaciones con respecto a los casos prácticos propuestos en los capítulos anteriores; éstas son:

1. La definición de las tablas se hace de forma independiente a las restricciones. Inicialmente se definen las tablas en función de tipos existentes y,

[27] El lector puede apreciar fácilmente que en el hipotético sistema que manejara esta base de datos se debería controlar que la tabla *Faunas* se fuera actualizando a medida que se fueran considerando nuevas capturas de peces en aquellos lugares en los cuales no se tuviera constancia hasta el momento de la existencia de esos peces capturados.

posteriormente se definen las restricciones de clave, integridad y de dominio. Este método de definición del esquema, que es opcional, permite la separación conceptual de los elementos que intervienen en la representación intencional de un problema.

2. Cuando se define la estructura de los objetos sobre la cual van a ser definidas las tablas, aquellos atributos que deben ser definidos en el esquema relacional como claves foráneas pueden ser definidos directamente como referencias a objetos a través de la cláusula *REF*. Este método garantiza la integridad de referencia en la tabla en la cual el tipo de objeto definido de esta forma esté presente sin necesidad de definir posteriormente una restricción de referencia para garantizar la misma a través de una cláusula *CONSTRAINT*. Para introducir al lector a este método de definición del esquema sólo se ha hecho uso de esta técnica para dos objetos *Lugares_Type* y *Eventos_Type*, a través de los cuales serán definidas las tablas *Lugares* y *Eventos*, respectivamente.
3. Los tipos de datos *LONG* no se pueden utilizar en la definición de los tipos de objetos, en su lugar se deben utilizar los tipos de datos *CLOB*.
4. Los tipos de datos *LONG RAW* tampoco pueden ser utilizados se deben utilizar los tipos de datos *BLOB*.

```
   /* Se borran las tablas existentes */
DROP TABLE Faunas;
DROP TABLE CapturasEventos;
DROP TABLE Concursos;
DROP TABLE Participaciones;
DROP TABLE Permisos;
DROP TABLE Licencias;
DROP TABLE CapturasSolos;
DROP TABLE Afiliados;
DROP TABLE Eventos;
DROP TABLE Peces;
DROP TABLE Lugares;
DROP TABLE Cauces;

   /* Se borran los tipos de objetos existentes */
DROP TYPE Faunas_Type;
DROP TYPE CapturasEventos_Type;
DROP TYPE Concursos_Type;
DROP TYPE Participaciones_Type;
DROP TYPE Permisos_Type;
DROP TYPE Licencias_Type;
DROP TYPE CapturasSolos_Type;
DROP TYPE Afiliados_Type;
DROP TYPE Eventos_Type;
DROP TYPE Peces_Type;
DROP TYPE Lugares_Type;
DROP TYPE Cauces_Type;
```

```
   /* Se definen los tipos de objetos */
CREATE OR REPLACE TYPE Afiliados_Type AS OBJECT (
   ficha NUMBER(4),
   nombre_afiliado VARCHAR2(20),
   apellidos_afiliado VARCHAR2(30),
   direccion_afiliado VARCHAR2(30),
   telf_afiliado NUMBER(9),
   sexo_afiliado CHAR(1),
   nacimiento_afiliado DATE,
   od_afiliado CLOB )
   /
CREATE OR REPLACE TYPE Licencias_Type AS OBJECT (
   licencia VARCHAR2(9) )
   /
CREATE OR REPLACE TYPE Permisos_Type AS OBJECT (
   ficha NUMBER(4),
   licencia VARCHAR2(9) )
   /
CREATE OR REPLACE TYPE Cauces_Type AS OBJECT (
   cauce VARCHAR2(50),
   od_cauce CLOB )
   /
   /* En este tipo de objeto se define el atributo cauce como
   una referencia al tipo de objeto que lo representa */
CREATE OR REPLACE TYPE Lugares_Type AS OBJECT (
   lugar VARCHAR2(50),
   comunidad VARCHAR2(20),
   od_lugar CLOB,
   cauce REF Cauces_Type )
   /
   /* En este tipo de objeto se define el atributo lugar como
   una referencia al tipo de objeto que lo representa */
CREATE OR REPLACE TYPE Eventos_Type AS OBJECT (
   evento VARCHAR2(60),
   fecha_evento DATE,
   lugar REF Lugares_Type,
   od_evento CLOB )
   /
CREATE OR REPLACE TYPE Peces_Type AS OBJECT (
   pez VARCHAR2(30),
   od_pez CLOB )
   /
CREATE OR REPLACE TYPE Participaciones_Type AS OBJECT (
   ficha NUMBER(4),
   evento VARCHAR2(50),
   posicion NUMBER(3),
   trofeo VARCHAR2(30) )
   /
```



```
CREATE OR REPLACE TYPE CapturasSolos_Type AS OBJECT (
   ficha NUMBER(4),
   pez VARCHAR2(30),
   fecha_pesca DATE,
   hora_pesca DATE,
   peso NUMBER(5,3),
   talla NUMBER(3),
   foto BLOB,
   lugar VARCHAR2(40),
   aval1 NUMBER(4),
   aval2 NUMBER(4) )
   /
CREATE OR REPLACE TYPE CapturasEventos_Type AS OBJECT (
   ficha NUMBER(4),
   pez VARCHAR2(30),
   fecha_pesca DATE,
   hora_pesca DATE,
   peso NUMBER(2),
   talla NUMBER(3),
   foto BLOB,
   evento VARCHAR2(60),
   puntos NUMBER(4) )
   /
CREATE OR REPLACE TYPE Faunas_Type AS OBJECT (
   lugar VARCHAR2(40),
   pez VARCHAR2(30),
   tallamax NUMBER(3),
   pesomax NUMBER(5,3),
   numemax NUMBER(2),
   tallamin NUMBER(3),
   veda CLOB )
   /
CREATE OR REPLACE TYPE Concursos_Type AS OBJECT(
   evento VARCHAR2(50),
   pez VARCHAR2(30),
   tallamin NUMBER(3),
   numemax NUMBER(2) )
   /
   /* Se definen las tablas del esquema relacional a partir de
   los tipos de objetos previamente declarados y se asignan
   características a los atributos de las mismas */
CREATE TABLE Afiliados OF Afiliados_Type (
   ficha NOT NULL,
   nombre_afiliado NOT NULL,
   apellidos_afiliado NOT NULL )
   /
```

```
CREATE TABLE Licencias OF Licencias_Type (
   licencia NOT NULL )
   /
CREATE TABLE Permisos OF Permisos_Type (
   ficha NOT NULL,
   licencia NOT NULL )
   /
CREATE TABLE Cauces OF Cauces_Type (
   cauce NOT NULL )
   /
CREATE TABLE Lugares OF Lugares_Type (
   lugar NOT NULL,
   cauce NOT NULL )
   /
CREATE TABLE Eventos OF Eventos_Type (
   evento NOT NULL,
   fecha_evento NOT NULL )
   /
CREATE TABLE Peces OF Peces_Type (
   pez NOT NULL )
   /
CREATE TABLE Participaciones OF Participaciones_Type (
   ficha NOT NULL,
   evento NOT NULL )
   /
CREATE TABLE CapturasSolos OF CapturasSolos_Type (
   ficha NOT NULL,
   pez NOT NULL,
   fecha_pesca NOT NULL,
   hora_pesca NOT NULL )
   /
CREATE TABLE CapturasEventos OF CapturasEventos_Type (
   ficha NOT NULL,
   pez NOT NULL,
   fecha_pesca NOT NULL,
   hora_pesca NOT NULL,
   evento NOT NULL )
   /
CREATE TABLE Faunas OF Faunas_Type (
   lugar NOT NULL,
   pez NOT NULL )
   /
CREATE TABLE Concursos OF Concursos_Type (
   evento NOT NULL,
   pez NOT NULL )
   /
```



```
   /* Se definen las claves de las tablas */
ALTER TABLE Afiliados
   ADD CONSTRAINT pk_Afiliados
    PRIMARY KEY (ficha);
ALTER TABLE Licencias
   ADD CONSTRAINT pk_lic
    PRIMARY KEY (licencia);
ALTER TABLE Permisos
   ADD CONSTRAINT pk_Permisos
    PRIMARY KEY (ficha, licencia);
ALTER TABLE Cauces
   ADD CONSTRAINT pk_Cauces
    PRIMARY KEY (cauce);
ALTER TABLE Lugares
   ADD CONSTRAINT pk_lugar
    PRIMARY KEY (lugar);
ALTER TABLE Eventos
   ADD CONSTRAINT pk_evento
    PRIMARY KEY (evento);
ALTER TABLE Peces
   ADD CONSTRAINT pk_Peces
    PRIMARY KEY (pez);
ALTER TABLE Participaciones
   ADD CONSTRAINT pk_Participaciones
    PRIMARY KEY (ficha,evento);
ALTER TABLE CapturasSolos
   ADD CONSTRAINT pk_CapturasSolos
    PRIMARY KEY (ficha, pez, fecha_pesca, hora_pesca);
ALTER TABLE CapturasEventos
   ADD CONSTRAINT pk_CapturasEventos
    PRIMARY KEY (ficha, pez, fecha_pesca, hora_pesca);
ALTER TABLE Faunas
   ADD CONSTRAINT pk_Faunas
    PRIMARY KEY (lugar, pez);
ALTER TABLE Concursos
   ADD CONSTRAINT pk_Permiso
    PRIMARY KEY (evento, pez);
/* Se definen las restricciones de integridad de referencia */
ALTER TABLE Permisos
   ADD CONSTRAINT fk_per_afi
    FOREIGN KEY (ficha)
    REFERENCES Afiliados(ficha)
    ON DELETE CASCADE;
ALTER TABLE Permisos
   ADD CONSTRAINT fk_per_lic
    FOREIGN KEY (licencia)
```

```
    REFERENCES Licencias(licencia)
    ON DELETE CASCADE;
ALTER TABLE Participaciones
   ADD CONSTRAINT fk_par_aso
    FOREIGN KEY (ficha)
    REFERENCES Afiliados(ficha)
    ON DELETE CASCADE;
ALTER TABLE Participaciones
   ADD CONSTRAINT fk_par_eve
    FOREIGN KEY (evento)
    REFERENCES Eventos(evento)
    ON DELETE CASCADE;
ALTER TABLE CapturasSolos
   ADD CONSTRAINT fk_cps_af1
    FOREIGN KEY (ficha)
    REFERENCES Afiliados(ficha)
    ON DELETE CASCADE;
ALTER TABLE CapturasSolos
   ADD CONSTRAINT fk_cps_pez
    FOREIGN KEY (pez)
    REFERENCES Peces(pez)
    ON DELETE CASCADE;
ALTER TABLE CapturasSolos
   ADD CONSTRAINT fk_cps_lug
    FOREIGN KEY (lugar)
    REFERENCES Lugares(lugar)
    ON DELETE CASCADE;
ALTER TABLE CapturasSolos
   ADD CONSTRAINT fk_cps_af2
    FOREIGN KEY (aval1)
    REFERENCES Afiliados(ficha);
ALTER TABLE CapturasSolos
   ADD CONSTRAINT fk_cps_af3
    FOREIGN KEY (aval2)
    REFERENCES Afiliados(ficha);
ALTER TABLE CapturasEventos
   ADD CONSTRAINT fk_cpe_afi
    FOREIGN KEY (ficha)
    REFERENCES Afiliados(ficha)
    ON DELETE CASCADE;
ALTER TABLE CapturasEventos
   ADD CONSTRAINT fk_cpe_pez
    FOREIGN KEY (pez)
    REFERENCES Peces(pez)
    ON DELETE CASCADE;
ALTER TABLE CapturasEventos
   ADD CONSTRAINT fk_cpe_eve
    FOREIGN KEY (evento)
```



```
    REFERENCES Eventos(evento)
    ON DELETE CASCADE;
ALTER TABLE Faunas
   ADD CONSTRAINT fk_fau_lug
    FOREIGN KEY (lugar)
    REFERENCES Lugares(lugar)
    ON DELETE CASCADE;
ALTER TABLE Faunas
   ADD CONSTRAINT fk_fau_pez
    FOREIGN KEY (pez)
    REFERENCES Peces(pez)
    ON DELETE CASCADE;
ALTER TABLE Concursos
   ADD CONSTRAINT fk_per_eve
    FOREIGN KEY (evento)
    REFERENCES Eventos(evento)
    ON DELETE CASCADE;
ALTER TABLE Concursos
   ADD CONSTRAINT fk_per_pez
    FOREIGN KEY (pez)
    REFERENCES Peces(pez)
    ON DELETE CASCADE;
   /* Se definen las restricciones de dominio */
ALTER TABLE Licencias
   ADD CONSTRAINT ck_lic
    CHECK (licencia = UPPER(licencia));
ALTER TABLE Cauces
   ADD CONSTRAINT ck_cau
    CHECK (cauce = INITCAP(cauce));
ALTER TABLE Lugares
   ADD CONSTRAINT ck_lug
    CHECK (lugar = INITCAP(lugar));
ALTER TABLE Lugares
   ADD CONSTRAINT ck_comun
    CHECK (comunidad = UPPER(comunidad));
ALTER TABLE Afiliados
   ADD CONSTRAINT ck_nap
    CHECK (apellidos_afiliado = INITCAP(apellidos_afiliado));
ALTER TABLE Afiliados
   ADD CONSTRAINT ck_saf
    CHECK (sexo_afiliado IN ('H','M'));
ALTER TABLE Afiliados
   ADD CONSTRAINT ck_naf
    CHECK (nombre_afiliado = INITCAP(nombre_afiliado));
ALTER TABLE Peces
   ADD CONSTRAINT ck_pez
```

```
    CHECK (pez = INITCAP(pez));
ALTER TABLE CapturasSolos
   ADD CONSTRAINT ck_pso
    CHECK (peso > 0);
ALTER TABLE CapturasSolos
   ADD CONSTRAINT ck_tal
    CHECK (talla > 0);
ALTER TABLE CapturasEventos
   ADD CONSTRAINT ck_ps2
    CHECK (peso > 0);
ALTER TABLE CapturasEventos
   ADD CONSTRAINT ck_ta2
    CHECK (talla > 0);
ALTER TABLE Faunas
   ADD CONSTRAINT ck_pma
    CHECK (pesomax > = 0);
ALTER TABLE Faunas
   ADD CONSTRAINT ck_nma
    CHECK (numemax > = 0);
ALTER TABLE Faunas
   ADD CONSTRAINT ck_tmi
    CHECK (tallamin > = 0);
ALTER TABLE Faunas
   ADD CONSTRAINT ck_tma
    CHECK (tallamax > = 0);
ALTER TABLE Concursos
   ADD CONSTRAINT ck_tan
    CHECK (tallamin > 0);
ALTER TABLE Concursos
   ADD CONSTRAINT ck_maz
    CHECK (numemax > = 0);
```

11.4.2. Manipulación de la Base de Datos

Cap11ej01.sql

Obtener la relación de las licencias de los miembros de la familia 'Lozano Conde'.

De la tabla Afiliados se obtiene el atributo ficha de todas las tuplas que cumplan la condición de que apellidos_afiliado tenga el valor 'Lozano Conde'. El resultado de esta subconsulta se utiliza para obtener el atributo licencia de todas las tuplas de la tabla Permisos cuyo atributo ficha tenga el valor que fue devuelto anteriormente.

```
SELECT licencia "Licencias"
   FROM Permisos
   WHERE ficha IN
     (SELECT ficha
        FROM Afiliados
        WHERE apellidos_afiliado = 'Lozano Conde');
```


Resultado

```
Licencias
---------
A-1111111
B-1111111
D-1111111
C-1111111
E-1111111
```

Cap11ej02.sql

Obtener la mejor posición alcanzada en competición a lo largo de su vida por el afiliado 3456.

De la tabla Participaciones se obtienen los atributos evento, posición y trofeo de todas las tuplas que cumplan la condición de que ficha tenga el valor 3456 y posicion tenga el valor resultante de una nueva subconsulta sobre la tabla Participaciones en la que se obtiene la mejor posición alcanzada por el participante anteriormente indicado.

```
   /* Se formatea la salida */
COLUMN evento FORMAT A50
COLUMN posicion FORMAT 999
COLUMN trofeo FORMAT A14
SELECT evento, posicion, trofeo
   FROM Participaciones
   WHERE ficha = 3456 AND posicion =
     (SELECT MIN(posicion)
        FROM Participaciones
        WHERE ficha = 3456);
```

Resultado

```
EVENTO                                             POSICION TROFEO
-------------------------------------------------- -------- ------
1º Concurso de Pesca Ciudad de Granada                    4
```

Cap11ej03.sql

Obtener los cauces y en qué lugar de ellos se pueden encontrar Tencas.

De la tabla Lugares se obtienen los atributos cauce y comunidad de todas las tuplas que cumplan la condición de que lugar tiene el valor devuelto por una subconsulta sobre la tabla Faunas en la que se obtienen todas las tuplas que cumplen la condición de que el atributo pez tiene el valor 'Tencas'.

```
   /* En este ejemplo se puede observar cómo se referencia un
   atributo que ha sido definido haciendo uso de una
   referencia a un tipo de objeto. La forma de hacerse es
   mediante la referencia al nombre del objeto, seguida por
   un punto y seguida por el nombre del atributo al cual se
   desea acceder.
   Se observa además que, en este caso, es necesario definir
   un alias para la tabla Lugares. La función de este alias
   es instanciar un objeto de la clase Lugares para que sea
   utilizado por la función de consulta */
   /* Se formatea la salida */
COLUMN lugar FORMAT A30
COLUMN cauce FORMAT A23
COLUMN comunidad FORMAT A10
```

```
SELECT L.lugar, L.cauce.cauce cauce, comunidad
   FROM Lugares L
   WHERE L.lugar IN
      (SELECT lugar
       FROM Faunas
       WHERE pez = 'Tenca');
```

Resultado

LUGAR	CAUCE	COMUNIDAD
Coto De Dílar	Río Dílar	ANDALUCÍA
Coto De El Bosque	Río Mojaceite	ANDALUCÍA
Coto De Pinillos	Río Genil	ANDALUCÍA
Coto Del Embalse De Colomera	Embalse De Colomera	ANDALUCÍA

Cap11ej04.sql

Obtener todos los eventos en los que participó el afiliado 3796 en 1995 y en los que no consiguió trofeo.

De la tabla Eventos se obtienen los atributos evento y fecha_evento de todas las tuplas que cumplen la condición de que fecha_evento está comprendida entre el 1 de enero de 1995 y el 31 de diciembre de 1995, y evento tenga como valor el resultado de una subconsulta realizada sobre la tabla Participaciones en la que se obtiene el valor del atributo evento de todas las tuplas que cumplan la condición de que ficha tiene el valor 3796 y trofeo sea NULL.

```
   /* Se formatea la salida */
COLUMN evento FORMAT A50
SELECT evento, TO_CHAR(fecha_evento,'DD/MM/YYYY') "FECHA"
   FROM Eventos
   WHERE TO_CHAR(fecha_evento,'YYYY') = '1995' AND evento IN
      (SELECT evento
          FROM Participaciones
          WHERE ficha = 3796 AND trofeo IS NULL);
```

Resultado

EVENTO	FECHA
1º Encuentro Provincial	28/06/1995
2º Encuentro Lures And Pikes	15/03/1995

Cap11ej05.sql

Obtener la ficha, nombre, apellidos, posición y trofeo de todos los participantes del evento 'Super Barbo' clasificados entre los tres primeros.

De la tabla Afiliados se obtienen los atributos ficha, nombre_afiliado y apellidos_afiliado, y de la tabla Participaciones los atributos posicion y trofeo de todas las tuplas que cumplen la condición de que el atributo evento tenga el valor 'Super Barbo'. El resultado de ambas consultas se reúne bajo la condición de que el atributo ficha generado por ambas consultas tenga el mismo valor. Los valores resultantes se restringen a que el valor del atributo posicion sea menor de 4.

```
   /* Se formatea la salida */
COLUMN nombre FORMAT A9
COLUMN apellidos FORMAT A18
COLUMN trofeo FORMAT A14
```



```
SELECT Afiliados.ficha, nombre_afiliado nombre,
       apellidos_afiliado apellidos ,posicion, trofeo
   FROM Afiliados, Participaciones
   WHERE evento = 'Super Barbo' AND posicion < 4
         AND Afiliados.ficha = Participaciones.ficha
   ORDER BY apellidos_afiliado;
```

Resultado

FICHA	NOMBRE	APELLIDOS	POSICION	TROFEO
2009	Jesús	Fernández Tamayo	3	
1000	Enrique	Lozano Conde	2	
1002	Alfonsa	Lozano Conde	1	Copa y 20000

Cap11ej06.sql

Obtener el nombre y apellidos de todas las personas que en alguno de los eventos '1er Encuentro Lures and Pikes' hayan realizado capturas de un peso superior a la media que se registró en dicho evento.

De la tabla Afiliados se obtienen los atributos nombre_afiliado y apellidos_afiliado de todas las tuplas que cumplan la condición de que el atributo ficha tenga el valor devuelto por una subconsulta sobre la tabla CapturasEventos de aquellas tuplas que cumplen la condición de que:

1. *El valor del atributo peso sea superior al devuelto por una subconsulta realizada sobre la tabla CapturasEventos en la que se obtiene el valor medio (AVG) de peso para todas las tuplas que cumplen la condición de que evento contiene el valor '1er Encuentro Lures and Pikes'.*
2. *El valor del atributo evento contenga '1er Encuentro Lures and Pikes'.*

```
   /* Se formatea la salida */
COLUMN nombre_afiliado FORMAT A20
COLUMN apellidos_afiliado FORMAT A40
SELECT nombre_afiliado, apellidos_afiliado
   FROM Afiliados
   WHERE ficha IN
     (SELECT ficha
        FROM CapturasEventos
        WHERE evento='1er Encuentro Lures And Pikes'
              AND peso >
           (SELECT AVG(peso)
              FROM CapturasEventos
              WHERE evento='1er Encuentro Lures And Pikes'));
```

Resultado

NOMBRE_AFILIADO	APELLIDOS_AFILIADO
Enrique	Lozano Conde
Antonio	Fernández Lorente
Antonio	López Sillero
Bernardo	Fernández González

Cap11ej07.sql

Obtener todas las personas que han realizado una captura en solitario pero que nunca han avalado la captura de otro.

De la tabla Afiliados se obtienen los atributos ficha, nombre_afiliado y apellidos_afiliado de todas las tuplas que cumplan las condiciones de que:

1. *EXISTE alguna tupla en la tabla CapturasSolos en la que el atributo ficha tenga el valor del mismo atributo de la tabla Afiliados.*
2. *NO EXISTE ninguna tupla en la tabla CapturasSolos en la que el atributo aval1 tenga el valor del atributo ficha de la tabla Afiliados.*
3. *NO EXISTE ninguna tupla en la tabla CapturasSolos en la que el atributo aval2 tenga el valor del atributo ficha de la tabla Afiliados.*

```
/* Se formatea la salida */
COLUMN ficha FORMAT 99999
COLUMN nombre_afiliado FORMAT A20
COLUMN apellidos_afiliado FORMAT A40
SELECT ficha, nombre_afiliado, apellidos_afiliado
   FROM Afiliados
   WHERE EXISTS
            (SELECT ficha
               FROM CapturasSolos
               WHERE ficha = Afiliados.ficha)
         AND NOT EXISTS
            (SELECT aval1
               FROM CapturasSolos
               WHERE aval1 = Afiliados.ficha)
         AND NOT EXISTS
            (SELECT aval2
               FROM CapturasSolos
               WHERE aval2 = Afiliados.ficha);
```

Resultado

```
 FICHA NOMBRE_AFILIADO      APELLIDOS_AFILIADO
------ -------------------- --------------------------
  1000 Enrique              Lozano Conde
```

Cap11ej08.sql

Confirmar si alguien ha capturado más peces de los permitidos en el evento 'Super Barbo'.

Se contabiliza el número de tuplas de la tabla CapturasEventos que cumplen la condición de que dicho número es mayor que el valor del atributo numemax de la tabla Concursos de aquella tupla que cumple la condición de que evento tiene el valor 'Super Barbo'. Si el resultado de la consulta es 'ninguna fila' entonces nadie habrá infringido las normas.

```
SELECT ficha, pez, COUNT (*) "CAPTURAS"
   FROM CapturasEventos
   WHERE evento = 'Super Barbo'
   GROUP BY ficha, pez
     HAVING COUNT (*) >
       (SELECT numemax
          FROM Concursos
          WHERE evento = 'Super Barbo' AND
                Concursos.pez = CapturasEventos.pez);
```


Resultado

```
 FICHA PEZ         CAPTURAS
------ ---------- ---------
  1002 Barbo              6
```

Cap11ej09.sql

Actualizar el peso máximo de una trucha, para todos los lugares de la comunidad Andaluza, al de la trucha más pesada pescada en el evento 'La Gran Trucha'.

Se actualiza el valor del atributo pesomax para todas las tuplas de la tabla Faunas que cumplen la condición de que su atributo pez tenga el valor 'Trucha' y el atributo lugar tenga el valor resultante de una subconsulta sobre la tabla Lugares en la que se obtiene el valor de lugar de aquellas tuplas en las que el atributo comunidad tiene el valor 'ANDALUCÍA'. El valor de pesomax es actualizado al resultado de obtener el valor peso máximo existente entre todas las tuplas de CapturasEventos que cumplen la condición de que pez tiene el valor 'Trucha' y evento el valor 'La Gran Trucha'.

```
UPDATE Faunas
   SET pesomax =
     (SELECT MAX(peso)
      FROM CapturasEventos
      WHERE evento = 'La Gran Trucha' AND pez = 'Trucha')
   WHERE pez = 'Trucha' AND lugar IN
     (SELECT lugar
      FROM Lugares
      WHERE comunidad = 'ANDALUCÍA');
```

Resultado

```
10 filas actualizadas.
```

Cap11ej10.sql

Obtener quién ha pescado el pez récord de cada lugar y si esta captura fue realizada en un evento o no a partir del año 1998.

Para realizar esta consulta es necesario realizar una unión de dos subconsultas, una sobre la tabla CapturasSolos y la otra sobre la reunión de las tablas CapturasEventos y Eventos. En estas subconsultas se proyecta sobre la información solicitada haciendo uso de los operadores MAX.

```
/* En este ejercicio se aprecia de nuevo la forma en la
cual se debe referenciar a un atributo de un objeto al
cual se ha hecho referencia mediante la cláusula REF en la
definición de un tipo de objeto */
/* Se formatea la salida */
COLUMN pez FORMAT A12
COLUMN talla FORMAT 999
COLUMN lugar.lugar FORMAT A30
COLUMN ficha FORMAT 99999
SELECT DISTINCT pez, talla, e.lugar.lugar, 'Evento', ficha
   FROM CapturasEventos c, Eventos e
   WHERE e.evento = c.evento
         AND fecha_pesca>TO_DATE('1998', 'YYYY') AND talla=
     (SELECT MAX(talla)
```

```
        FROM CapturasEventos cint,Eventos eint
        WHERE eint.lugar.lugar=e.lugar.lugar AND c.pez=cint.pez
             AND fecha_pesca>TO_DATE('1998', 'YYYY')
             AND cint.evento=eint.evento)
UNION
SELECT pez, talla, lugar, 'Solitario', ficha
   FROM CapturasSolos c
   WHERE talla=
     (SELECT MAX(talla)
        FROM CapturasSolos cint
        WHERE cint.lugar=c.lugar AND c.pez=cint.pez
             AND fecha_pesca>TO_DATE('31/12/1998','DD/MM/YYYY'))
             AND fecha_pesca>TO_DATE('31/12/1998','DD/MM/YYYY');
```

Resultado

PEZ	TALLA	LUGAR.LUGAR	EVENTO	FICHA
Barbo	67	Coto De Borosa	Solitario	2000
Barbo	68	Coto De Embalse De Béznar	Evento	2008
Barbo	68	Coto De Genazar	Solitario	1009
Barbo	68	Coto Del Colomera	Evento	1000
Barbo	69	Coto Del Embalse Quéntar	Evento	2001
Black-Bass	81	Coto De Fardes	Evento	1010
Black-Bass	90	Coto De San Rafael	Evento	1004
Carpa	97	Coto De Genazar	Evento	1000
Trucha	80	Coto De Alhama	Evento	1006
Trucha	89	Coto De Riofrío Intensivo	Solitario	2003

Cap11ejP1.sql

Construir un procedimiento PL/SQL que borre toda la información relacionada con un determinado cauce.

Debido a que el campo cauce en la tabla Lugares está definido como una referencia a otra tabla, cuando se realiza el borrado de un determinado cauce, esta referencia se queda incompleta, lo cual implica que los lugares y toda la información relacionada con ellos puede que quede en un estado no consistente. Para ello se creará este bloque que permitirá borrar toda la información relacionada con el cauce borrado. Habría primero que analizar qué campos de las tablas de la base de datos hacen referencia con el operador REF a alguna de las tablas que están relacionadas con la tabla CAUCES.

```
/* En este procedimiento el usuario podrá apreciar cómo
son manejados los objetos que forman parte de la
definición de otros objetos haciendo uso de la cláusula
REF */
CREATE OR REPLACE PROCEDURE borra_cauce
   /* Se definen las variables de la interfaz */
   (p_cauce VARCHAR2)
AS
   /* Se definen las variables internas de trabajo y el cursor
   para recorrer la tabla */
   cauce_inexistente EXCEPTION;
   contador NUMBER;
```



```
BEGIN
   SELECT count(*) INTO contador
    FROM Cauces
    WHERE UPPER(cauce)=UPPER(p_cauce);
   IF contador<1 THEN
    RAISE cauce_inexistente;
   END IF;
   DELETE FROM Eventos e
    WHERE e.lugar=
     (SELECT REF(l)
      FROM Lugares l
      WHERE l.cauce=
         (SELECT REF(c)
            FROM Cauces c
            WHERE UPPER(cauce)=UPPER(p_cauce)));
   DELETE FROM Lugares l
    WHERE l.cauce=
     (SELECT REF(c)
         FROM Cauces c
         WHERE UPPER(cauce)=UPPER(p_cauce));
COMMIT;
   /* Se controlan los posibles errores de operación */
EXCEPTION
   WHEN cauce_inexistente THEN
    DBMS_OUTPUT.PUT_LINE('NOMBRE DE CAUCE NO EXISTENTE');
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
   ROLLBACK;
END;
/
```

Prueba de ejecución

```
-- Se ejecuta una prueba del procedimiento
    EXEC borra_cauce ('río fardes');
```

CAPÍTULO 12

PROYECCIONES DE PELÍCULAS

12.1. ENUNCIADO DEL PROBLEMA

Los españoles cada vez acuden con más frecuencia al cine, y ello se aprecia en la recaudación de las taquillas donde se proyectan tanto películas extranjeras como españolas, experimentando, estas últimas, un gran incremento en el número de espectadores en los últimos años.

Es interesante el mantener la información correspondiente a las películas estrenadas en los cines españoles en los últimos años. Sobre cada película interesa conocer información sobre el director, actores y productores que participan en la misma, así como los premios en los grandes festivales (Hollywood, Cannes, Berlín, etc.) que han podido recibir tanto las películas como los directores, actores y productores a lo largo de la historia de estos festivales. Además, y con fines puramente informativos y no contables, interesa considerar el número de espectadores y la recaudación que ha tenido cada película en cada cine donde ha sido estrenada, —teniendo en cuenta posibles reestrenos—.

En este problema que se intenta abordar es importante conocer además los siguientes supuestos semánticos:

SUPUESTO 1: El título de una película no es único. Sobre una película (nos referimos al guión de la misma, naturalmente) pueden realizarse distintas versiones y éstas pueden tener el mismo o distinto título. De igual forma, dos películas diferentes (nos referimos de nuevo a los guiones de las mismas) pueden tener también el mismo título.

SUPUESTO 2: Aunque dos películas tengan el mismo título, éstas no pueden haberse producido en el mismo año, ni en ellas intervienen los mismos actores, ni son dirigidas por los mismos directores, etc. Además, cada película puede ser identificada por un código que identifica la propiedad intelectual de la misma.

SUPUESTO 3: Una película puede ser producida por más de un productor, y viceversa.

SUPUESTO 4: Una película puede ser dirigida por más de un director, y viceversa.

SUPUESTO 5: En una película pueden intervenir uno o varios actores, y viceversa[28].

SUPUESTO 6: No interesa conocer a los directores, productores ni actores a no ser que participen en alguna de las películas sobre las que se desea mantener información.

SUPUESTO 7: Cada actor, en una película, interpreta un papel que tiene asignado un determinado nivel de importancia en la misma. Los niveles de importancia con que pueden participar los actores en las películas están predefinidos de antemano (actor principal, actriz principal, actor secundario, etc.).

SUPUESTO 8: Existe una serie de festivales de cine a los que se presentan las películas con la finalidad de optar a algún premio. Los festivales organizan certámenes anuales para premiar los trabajos cinematográficos realizados, generalmente, en ese año.

SUPUESTO 9: Los premios son característicos de cada festival y pueden ser otorgados (o no) en cada certamen, pudiendo, por tanto, quedar premios desiertos en los mismos.

SUPUESTO 10: Por otra parte, en ciertos certámenes se conceden premios a actores y directores basándose en la labor realizada en su carrera profesional, independientemente de que en el certamen participen en alguna película que se presente a concurso. Es de interés conocer también este tipo de premios concedidos a las personas sobre las cuales se tiene información.

SUPUESTO 11: Las películas son proyectadas en los cines de toda España y, para cada cine, en una o varias de sus salas de proyección. Una película puede ser proyectada más de una vez en el mismo cine en la misma o distinta sala. Se considera una proyección a la serie de días en que una película se proyecta al público desde el primer pase (estreno o no) hasta que se retira de la sala.

SUPUESTO 12: Se va a considerar que los nombres de los cines son únicos con independencia de la zona geográfica de los mismos. Este supuesto (no muy cierto en el mundo real) se introduce únicamente para simplificar la identificación de los cines[29].

[28] Existen películas en las que no intervienen actores, como pueden ser los documentales, dibujos animados, etc., en tal caso se considerará un actor ficticio o simbólico en la misma.

12.2. MODELO CONCEPTUAL

El enunciado y el conjunto de supuestos semánticos expuestos anteriormente aportan la información necesaria para extraer del dominio de discurso el siguiente conjunto de tipos de entidad e interrelación:

12.2.1. Un primer análisis del problema

Se va, a continuación, a realizar una descripción inicial de los tipos de entidad e interrelación que más fácilmente pueden extraerse del problema, los cuales, una vez expuestos, nos permitirán avanzar en el proceso de análisis y extraer del problema otros tipos de entidad (e interrelación), más complejos y difíciles de advertir inicialmente, hasta obtener el modelo conceptual perseguido.

Tipo de entidad *Pelicula*: el cual representa al "***conjunto de películas que son proyectadas en los cines españoles y sobre las cuales interesa mantener información***". Es el tipo de entidad más importante en el problema, siendo introducido por la mayor parte de los supuestos semánticos, así como por el propio enunciado del problema.

Se van a considerar los siguientes atributos para este tipo de entidad: ***titulo_p, titulo_s, ano_produccion, cip, presupuesto, duracion, nacionalidad***. El atributo *titulo_p* representa el título de la película en castellano y el atributo *titulo_s* representa el título de la película en otro idioma (si fuera una película extranjera y/o un posible segundo título de la misma). El atributo *cip* representa el código de identificación de la película (código de la propiedad de la misma, SUPUESTO 2), representando el resto de los atributos, como su nombre indica, el año en que se produjo, el presupuesto, la duración en minutos y la nacionalidad de la misma, respectivamente.

El identificador de este tipo de entidad es, según el SUPUESTO 2, el atributo *cip*, pudiéndose considerar como identificador alterno el formado por la agregación de los atributos *titulo_p* y *ano_produccion* (SUPUESTO 1), pero no se pueden considerar como identificadores alternos ni al formado por la agregación de los atributos *titulo_s* y *ano_produccion* ni la agregación de los atributos *titulo_p* y *titulo_s*, debido a que puede haber películas españolas que no tengan un segundo título, violándose en ese caso la integridad de la clave.

Tipo de entidad *Cine*: representando al "***conjunto de cines en los cuales se realizan las proyecciones de las películas***" (SUPUESTOS 11 y 12).

[29] Se podría introducir como identificador de los cines el número de identificación fiscal de la empresa de igual forma.

Para este tipo de entidad se han considerado los atributos: *cine*, para identificar al nombre de los cines, *ciudad_cine* y *direccion_cine* y, según el SUPUESTO 12, se ha elegido como identificador de este tipo de entidad al primero de ellos.

Tipo de entidad *Sala*: representando al "*conjunto de salas de proyección de los cines*" (SUPUESTO 11). Está claro que se trata de un tipo de entidad débil por identificación con respecto al tipo de entidad *Cine*, pues no existirá ni se podrá identificar una sala de proyección si no se conoce el cine al cual pertenece.

Este tipo de entidad, por tanto, vendrá caracterizado por los atributos: *cine*, heredado del tipo de entidad *Cine*, *sala*, el cual identifica las salas de cada cine y *aforo* que representa la capacidad de la sala. El identificador de este tipo de entidad es la agregación de los atributos *cine* y *sala*.

Tipo de entidad *Festival*: representando a "*aquellas organizaciones que periódicamente, o no, organizan certámenes para juzgar y premiar, en su caso, la calidad de las películas y sus participantes*". Este tipo de entidad viene impuesto por el propio enunciado del problema y por otros supuestos como los SUPUESTOS 8-10.

Este tipo de entidad caracteriza a la organización y no al certamen en el cual se conceden los premios a las películas, como se verá más adelante. Sólo se considerarán como atributos a *festival*, como identificador y representando a los nombres de los festivales, y a *fundacion* representando el año en que se fundó el mismo.

Tipo de entidad *Certamen*: los certámenes representan a "*cada uno de los concursos en los cuales se juzgan y premian las películas*" (SUPUESTO 8). Se trata, por tanto, de un tipo de entidad débil por identificación con respecto al tipo de entidad *Festival*.

Por ello, este tipo de entidad vendrá caracterizado por los atributos: *festival*, heredado del tipo de entidad *Festival*, *certamen*, que representa al año en que se convoca ese festival, y *organizador*, que representa a la persona encargada de coordinar el evento. El identificador de este tipo de entidad es la agregación de los atributos *festival* y *certamen*.

Tipo de entidad *Persona*: el cual representa a "*cualquier persona que puede participar en la realización de una película, independientemente de su cometido*", según se detalla en los SUPUESTOS 3-5.

Este tipo de entidad, según el SUPUESTO 6, es débil por existencia con respecto al tipo de entidad *Pelicula*, pues no existirán en el dominio del problema a no ser que se considere alguna de las películas en las que intervienen.

Si ahora introducimos el siguiente supuesto:

SUPUESTO 12+1: En una misma película una misma persona puede desempeñar una o más de las siguientes tareas: director, productor y/o actor.

podemos clarificar los SUPUESTOS 3-7 considerando que en una película una misma persona puede realizar el trabajo de director y actor, por ejemplo.

El tipo de entidad *Persona* vendrá caracterizado por los atributos: *nombre_persona*, *nacionalidad_persona*, *sexo_persona*, siendo el identificador único y principal el atributo *nombre_persona*, con lo que se está considerando que no existen dos personas diferentes que puedan tener el mismo nombre (una simplificación admisible introducida en el problema para no complicar su desarrollo).

12.2.2. Avanzando en el proceso de análisis

Se podría, a priori, realizar dos consideraciones diferentes, como se ha representado en la figura 12.1:

1. Considerar tres nuevos tipos de entidad *Actor*, *Director* y *Productor* para representar a aquellas personas que realizan las funciones de actor, director y productor en las películas en las que participan. Estos tres tipos de entidad serían considerados como subtipos del tipo de entidad *Persona*, heredando los atributos de este último.
2. No considerar estos subtipos de entidad sino, en su lugar, considerar que *actor*, *director* y *productor* son funciones, tareas o trabajos que realizan las personas que intervienen en las películas consideradas.

CONSIDERACIÓN 1: *Tipos de entidades especializadas*

En este caso existirá una relación jerárquica entre el tipo de entidad *Persona* y los subtipos *Actor*, *Director* y *Productor*, la cual será total (si sólo se desea considerar estas tareas, o parcial en caso contrario) e inclusiva, puesto que una persona puede realizar varias de estas tareas en el conjunto de las películas que sean consideradas en el dominio del problema. En algunas películas una persona sólo realizará la tarea de actor, por ejemplo, pero en otra puede ser director y en otra director, productor y actor al mismo tiempo.

En este caso será necesario considerar que existen tres tipos de interrelaciones entre el tipo de entidad *Pelicula* y cada uno de los subtipos de entidad considerados. Estos tipos de interrelación no requieren ningún atributo que cualifique la tarea desarrollada por las personas en las películas, puesto que esta tarea está definida en base al subtipo de entidad que participa en cada uno de los tipos de interrelación.

Una de las ventajas que proporcionaría esta línea de análisis es la de poder considerar diferentes atributos para cada uno de los subtipos de entidad; atributos que no serían considerados en el tipo de entidad *Persona* sino sólo en los subtipos en los cuales fuera necesario, eliminando, así, redundancia en la representación conceptual y, por tanto, en la futura representación lógica.

Si no se consideran los subtipos (como se describe a continuación) todos los atributos (iguales y diferentes) de los subtipos de entidad deberán considerarse en el tipo de entidad *Persona*, lo que dará lugar a que puedan tomar, algunos de ellos, valores nulos para muchas entidades de este tipo.

CONSIDERACIÓN 2: *Tipo de entidad Tarea*

Si se tiene en cuenta este principio de análisis, será necesaria la consideración de un nuevo tipo de entidad:

Tipo de entidad *Tarea*: el cual representa a *"los diferentes trabajos, funciones o tareas que una persona puede desempeñar en una película"*. Este tipo de entidad permite representar los trabajos de actor, director y productor, así como cualquier otro tipo de trabajo que se desee y, por tanto, ampliar fácilmente, en el dominio del problema, el número de actividades o tareas que puede realizar una persona.

El tipo de entidad *Tarea* estará caracterizado por los atributos *tarea*, representando a las funciones que pueden realizar las personas en las películas, y es considerado como identificador único y principal de este tipo de entidad, y el atributo *sexo_tarea*.

Este atributo, *sexo_tarea*, es necesario considerarlo debido a que existen algunas tareas que pueden llevarlas a cabo las personas en las películas y que son dependientes del sexo de las mismas, lo que se aprecia claramente en la diferencia en el sexo que existe entre las tareas de actor principal y actriz principal. Por otro lado, existen tareas que pueden desempeñarlas las personas con independencia de su sexo, por ejemplo, la de director. Por ello, el atributo *sexo_tarea* para este tipo de entidad podrá tomar valores nulos o un valor que indique la indiferencia del sexo en la realización de una determinada tarea.

De esta forma, y como se ha considerado el atributo *sexo_persona* en el tipo de entidad *Persona*, y el atributo *sexo_tarea* en el tipo de entidad *Tarea*, se puede controlar fácilmente que las personas que realizan una determinada tarea en una película tienen un sexo compatible con el sexo de las personas que pueden realizar dicha tarea.

Bajo este punto de vista existirá un tipo de interrelación ternaria entre los tipos de entidad *Persona, Pelicula* y *Tarea* mediante la cual se representa a las personas que intervienen en una película y la tarea o tareas que desarrollan en la misma.

Se nos plantean, ahora, algunas preguntas: ¿Cuál de las dos consideraciones es la correcta? ¿Son las dos correctas? (Como se aprecia no consideramos la posibilidad de que ninguna de las dos no sea correcta). No vamos, en este momento, a decidir la línea de análisis a seguir a lo largo del ejercicio, ni a defender la validez o invalidez de una u otra consideración. Seguiremos avanzando en el análisis del problema planteado y, en este proceso, el lector podrá apreciar cuál de las dos consideraciones es la que debe ser adoptada. Por ahora, vamos a considerar que ambas son correctas y que cualquiera de las dos podría ser adoptada.

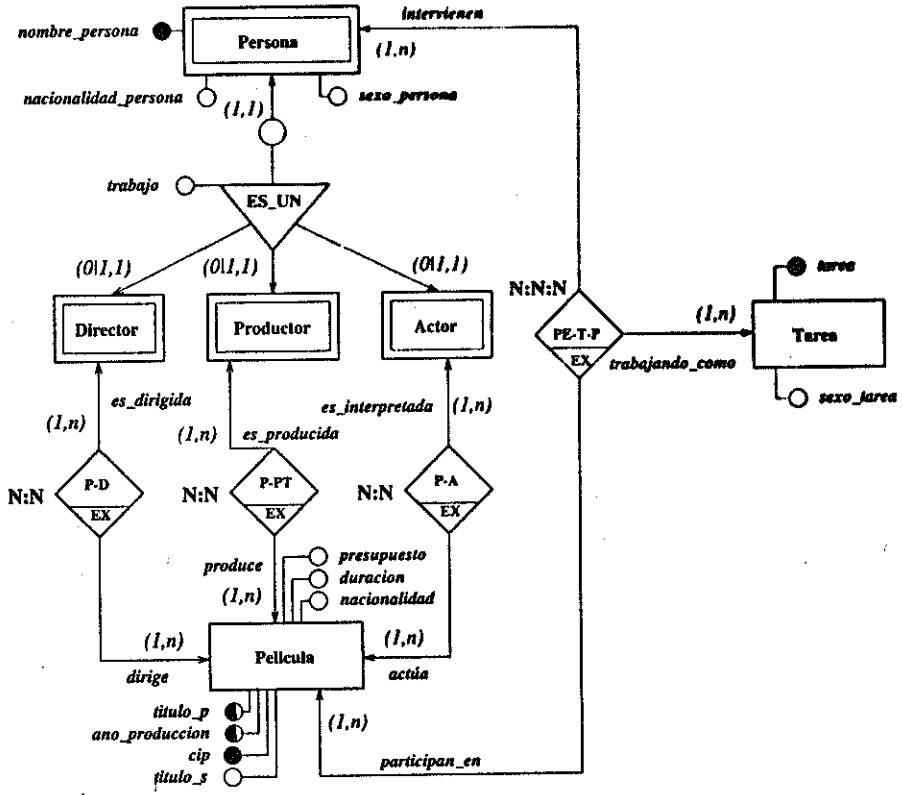


Figura 12.1 Las películas y las personas que intervienen

En la figura 12.1 se muestran, en un mismo diagrama, las dos consideraciones descritas anteriormente, mostrándose los tipos de entidades así como los tipos de interrelación en los que participarían.

Se puede observar en la figura 12.1 que si bien existe un tipo de interrelación entre los tipos de entidad *Actor* y *Pelicula* para conocer el *papel* de importancia que un actor realiza en una película es necesario considerar alguna de las dos alternativas siguientes:

1. Cualificar el tipo de interrelación *(P-A)* con un atributo (por ejemplo, denominado *papel*) que cualifique el papel que desarrolla el actor en la película, debido a que una persona puede, desarrollando el trabajo de actor, realizar un papel de importancia de entre un conjunto bien definido: actor principal, actor secundario, actriz principal, etc.
2. Considerar que el subtipo de entidad *Actor* viene caracterizado por un nuevo atributo (el atributo *papel*), que forma parte de su identificador y que representa

los diferentes papeles que una persona ha realizado como actor en las diferentes películas en las que ha intervenido.

Esta segunda consideración origina una duplicación en cuanto a la consideración de tantas entidades actores como una misma persona haya realizado diferentes papeles de importancia en las diferentes películas consideradas en el dominio del problema. Por otra parte, la primera opción dará lugar, como se verá a continuación, a algunos problemas en la representación conceptual y, sobre todo, en la representación relacional, para conseguir una integridad de la información, debido a la existencia de otros objetos existentes en el problema y que participan en tipos de interrelación tanto con el tipo de entidad *Persona* y sus subtipos (*Actor, Director*, y *Productor*) como con el tipo de entidad *Pelicula*, y en algunas de estas relaciones es necesario considerar el *papel* de los actores.

12.2.3. Los festivales, los certámenes y los premios

El enunciado del problema y los SUPUESTOS 8-10 aportan la información suficiente para reconocer la existencia de otros tipos de entidad que representan el hecho de que las películas se presentan a ciertos eventos para intentar alcanzar un reconocimiento en los mismos.

Como es necesario considerar los premios que obtienen las películas y, por tanto, también las personas que realizan alguna función en las mismas, es preciso considerar qué tareas son reconocidas por los festivales para ser premiadas en sus certámenes correspondientes. Por ello, es necesario considerar:

Tipo de interrelación *Festival/Tarea* **(F-T)**: en el que se representan qué tipo de trabajos o aportaciones que realizan las personas en las películas son consideradas por los diferentes festivales para su reconocimiento y premio. Se trata de un tipo de interrelación *N:N* que viene caracterizado por el atributo *galardon*, el cual representa el tipo de premio que se otorga en los certámenes del festival a la persona premiada (por ejemplo: león de oro, león de plata, óscar, etc.).

Pero como en los certámenes de los festivales se otorgan premios, además de a la labor de determinadas personas en las películas, por otras causas, con independencia de las personas que intervienen en las mismas (por ejemplo: premio a la mejor película, mejor fotografía, mejor ambientación, etc.) o, mejor dicho, con independencia de que las personas premiadas sean consideradas en el dominio del problema puesto que sólo son consideradas, en principio, aquellas personas que realizan las tareas de actor, director y productor. Entonces, estos otros premios que se dan a las películas deben ser también considerados, necesitándose definir el siguiente tipo de entidad:

Tipo de entidad *Premio*: el cual representa a "*los diferentes aspectos de una película por el cual, en los certámenes de los festivales, se reconoce la calidad de la misma*".

Este tipo de entidad es necesario especializarlo de forma total y exclusiva en dos subtipos que consideren los diferentes tipos de premios:

- Aquellos que se otorgan por la tarea realizada en las películas de las personas que se consideran en el problema; es decir, actores, directores y productores (si el lector no desea ampliar a otros tipos como diseñadores, escritores, etc.).
- Aquellos que se otorgan a otras personas o a las películas en sí.

Para lo cual, como se muestra en la figura 12.2, se definen dos subtipos de entidad:

Tipo de entidad *PremioTarea*: representando a "*aquellos premios que son considerados en los festivales para premiar la labor desempeñada por las personas, consideradas en el problema, que participan en las películas*".

Como se puede apreciar, se trata de un tipo de entidad idéntico al tipo de entidad considerado anteriormente, el tipo de entidad *Tarea*, en el sentido en que ambos tipos de entidad participarán en un tipo de interrelación *1:1*, de la forma siguiente:

- Si se consideran sólo aquellas tareas por las cuales pueden ser premiadas las personas en los festivales y aquellos premios correspondientes a las tareas consideradas, ambos tipos de entidad participarán en el tipo de interrelación con cardinalidades máxima y mínima igual a uno.

  Bajo este punto de vista, los tipos de entidad *Tarea* y *PremioTarea* serían el mismo tipo de entidad y, por tanto, sólo sería necesario considerar a uno de ellos. Sin embargo, y únicamente para que el lector pueda seguir más fácilmente el desarrollo del problema seguiremos considerando dos tipos de entidad distintos.

- Si se consideran otras tareas que las personas pueden realizar en las películas con independencia de que los festivales las consideren para ser premiadas, el tipo de entidad *PremioTarea* participará con cardinalidad mínima igual a cero.

  En este caso, los tipos de entidad *Tarea* y *PremioTarea* serían distintos y esta última sería débil con respecto al tipo de entidad *Tarea*, participando el tipo de entidad *PremioTarea* con cardinalidad mínima cero y máxima uno.

Independientemente de la consideración que se adoptara es necesario precisar que los atributos *premio* (identificador del tipo de entidad *PremioTarea*) y *tarea* (identificador del tipo de entidad *Tarea*) están definidos en el mismo dominio de datos, tomando, para el primer caso, los mismos valores.

Tipo de entidad *OtroPremio*: representando a "*cualquier otro premio que se considera en los festivales para premiar a las películas que se presentan a concurso en los certámenes de los mismos*".

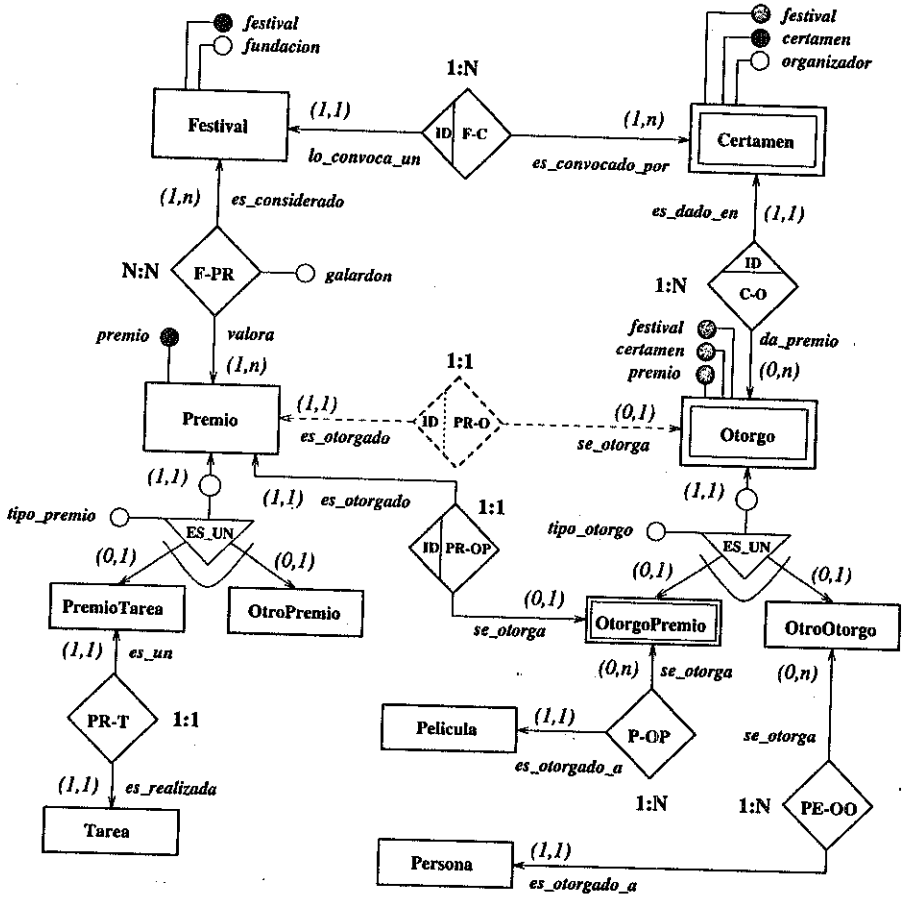


Figura 12.2 Los festivales, los certámenes y los premios

Se ha estado detallando los premios que son considerados en los festivales de cine a cuyos certámenes se presentan las películas para ser consideradas, por tanto, es necesario considerar la existencia de un tipo de interrelación entre estos dos tipos de entidades:

Tipo de interrelación *Festival/Certamen* **(F-C)**: en el cual el tipo de entidad *Festival* participa con cardinalidades *(1,1)* (un certamen está relacionado con uno y sólo un festival), y el tipo de entidad *Certamen* participa con cardinalidades *(1,n)* (un festival organiza diferentes certámenes, al menos uno para ser considerado en el problema).

Se puede observar que el tipo de entidad *Certamen* es débil por identificación con respecto al tipo de entidad *Festival*, puesto que no existirán certámenes de un festival que no exista y, en base a los atributos considerados en el tipo de entidad

Certamen, es necesaria para la identificación de una entidad de este tipo la identificación del festival al cual pertenece.

Los premios reconocidos por los festivales pueden o no ser otorgados a las películas y sus participantes que se presentan a los certámenes correspondientes. Por ello, es necesario considerar un nuevo tipo de entidad que represente los premios otorgados en cada uno de los certámenes de los festivales.

Tipo de entidad *Otorgo*: representando a "*los diferentes premios, de los reconocidos por los festivales, que se imparten en los diferentes certámenes de los mismos*".

El tipo de entidad *Otorgo* será:

- Un tipo de entidad débil por identificación con respecto al tipo de entidad *Certamen*, puesto que es necesario identificar el certamen en el que se entrega el premio ya que un mismo premio puede ser entregado en varios certámenes diferentes.
- Un tipo de entidad débil por identificación con respecto al tipo de entidad *Premio*, puesto que es necesario conocer el premio que se entrega en cada uno de los certámenes, en los cuales se pueden impartir varios de ellos.

Por otra parte, y como se describe en el SUPUESTO 10, en algunos certámenes se imparten premios a personas con independencia de que participen en alguna de las películas que se presentan a los mismos. Por ello, es necesario especializar el tipo de entidad *Otorgo* en dos subtipos, de la forma siguiente:

Tipo de entidad *OtorgoPremio*: representando a "*aquellos premios que se imparten en los certámenes, y que corresponden a los premios considerados en los festivales de los mismos; premios que se conceden tanto a las películas que se presentan como a las personas que participan en las mismas*".

Este tipo de entidad participa en un tipo de interrelación *1:1* con el tipo de entidad *Premio*, de forma que el tipo de entidad *Premio* participa con cardinalidades máxima y mínima igual a 1, y el tipo de entidad *OtorgoPremio* con cardinalidad mínima cero y máxima uno, debido a que pueden existir certámenes en los cuales quede un premio desierto o no se considere. Además, este tipo de entidad participa en un tipo de interrelación *N:1* con el tipo de entidad *Pelicula*, representando a la película a la que se le entrega el reconocimiento.

Tipo de entidad *OtroOtorgo*: representando a "*aquellos reconocimientos que se imparten en los certámenes a las personas por su labor profesional con independencia de que participen en alguna de las películas que se presentan a concurso en los mismos*".

Este tipo de entidad participará en un tipo de interrelación *N:1* con el tipo de entidad *Persona*, puesto que un otorgo de este tipo puede ser asignado a una persona (si no, no sería considerado) y una persona puede recibir cero o muchos otorgos de este tipo a lo largo de su vida.

De esta forma queda representado correctamente el SUPUESTO 10, por el cual es necesario considerar otros tipos de reconocimientos que en los certámenes se les otorgan a las personas por su labor profesional.

Por otra parte, queda también validado el SUPUESTO 9, por el cual se define que los premios que se otorgan en los certámenes son característicos de los festivales y no siempre todos los premios que se reconocen en los festivales son entregados en los certámenes; véase los tipos de interrelación *(F-PR)*, *(PR-O)* y *(PR-OP)*.

El tipo de interrelación *(P-OP)* permite representar aquellas películas que son premiadas en los certámenes. Estos premios podrán ser otorgados a las personas que se consideran que intervienen en las películas o por otras causas, y se tiene:

- Que a través del tipo de interrelación ternaria *(PE-T-P)* se representa a las personas que intervienen en las películas y la tarea que realizan,
- que a través del tipo de interrelación *(PR-T)* las tareas y los premios están directamente relacionados, y
- que a través del tipo de interrelación *(PR-OP)* también los premios y los otorgos están directamente relacionados

entonces, se está representando a las personas que intervienen en las películas y que, por tanto, reciben el premio concedido a éstas en los certámenes por la labor desarrollada.

12.2.4. Los cines, salas y las proyecciones

Sobre la base de varios supuestos del problema (SUPUESTOS 11 y 12), interesa conocer cómo ha respondido el público a las proyecciones de las películas en los cines. Por ello, interesa considerar el siguiente tipo de entidad:

Tipo de entidad *Proyeccion*: el cual representa *"las distintas proyecciones que se realizan de las películas en las salas de cine españolas"*. Como según el SUPUESTO 11 una misma película puede ser proyectada en varias salas del mismo o distinto cine en la misma o distinta fecha y, además, puede ser reestrenada en el mismo cine, es necesario considerar a este tipo de entidad como débil por identificación con respecto:

- Al tipo de entidad *Sala*, para identificar la sala del cine en la que se proyecta.
- Al tipo de entidad *Pelicula* para identificar la película que es proyectada.

Por lo que heredará los atributos identificadores de estos tipos de entidad y además incorpora los atributos: *fecha_estreno* —que representa la fecha del primer pase en la sala, sea estreno o reestreno—, *dias_estreno* —representando los días continuos que se proyectó—, *espectadores* —representando el número de espectadores que acudieron a verla— y, por último, *recaudacion* —representando la recaudación en los días de proyección—.

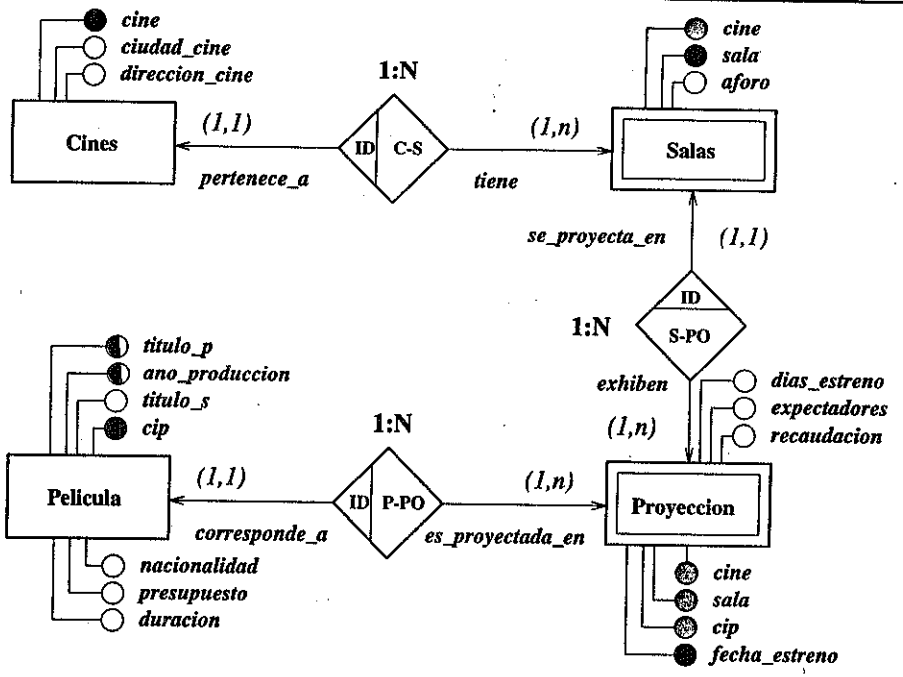


Figura 12.3 Relaciones entre los festivales, certámenes y premios

El tipo de entidad *Proyeccion* mantiene un conjunto de relaciones como se muestra en la figura 12.3, las cuales describimos a continuación:

Tipo de interrelación *Sala/Proyeccion* **(S-PO)**: en el que el tipo de entidad *Sala* participa con cardinalidades *(1,1)* y el tipo de entidad *Proyección* con cardinalidades *(1,n)*.

Tipo de interrelación *Pelicula/Proyeccion* **(P-PO)**: en el que el tipo de entidad *Pelicula* participa con cardinalidades *(1,1)* y el tipo de entidad *Proyeccion* con cardinalidades *(1,n)*.

En ambos casos, y debido a la debilidad por identificación del tipo de entidad *Proyeccion* con los tipos de entidad *Sala* y *Pelicula*, las cardinalidades con las que participan en los tipos de interrelación *(S-PO)* y *(P-PO)* son las mismas (ver figura 12.3).

12.3. MODELO RELACIONAL

Vamos a describir a continuación las tablas resultantes de la traducción del modelo conceptual propuesto, indicando para los casos más significativos las reglas aplicadas para ello.

BASE DE DATOS PELICULAS	
Pelicula	*(cip, titulo_p, ano_produccion, titulo_s, nacionalidad, presupuesto, duracion)*
Personaje	*(nombre_persona, nacionalidad_persona, sexo_persona)*
Tarea	*(tarea, sexo_tarea)*
Trabajo	**(cip, nombre_persona, tarea)**
Festival	*(festival, fundacion)*
Certamen	**(festival**, *certamen, organizador)*
Premio	*(premio,* **tarea)**
Festival_Premio	**(festival, premio**, *galardon)*
Otorgo	**(festival, certamen, premio, cip)**
Reconocimiento	**(festival, certamen, premio, nombre_persona)**
Cine	*(cine, ciudad_cine, direccion_cine)*
Sala	**(cine**, *sala, aforo)*
Proyeccion	**(cine, sala, cip**, *fecha_estreno, dias_estreno, espectadores, recaudacion)*

Este esquema ha sido derivado de la forma siguiente:

- La tabla *Pelicula*: por aplicación de la regla *RTECAR-1* sobre el tipo de entidad de igual nombre.
- La tabla *Personaje*: por aplicación de la *RTECAR-1* sobre el tipo de entidad de *Persona*.
- La tabla *Tarea*: por aplicación de la *RTECAR-1* sobre el tipo de entidad de igual nombre.
- La tabla *Trabajo*: por aplicación de la regla *RTECAR-4* sobre el tipo de interrelación (*PE-T-P*).
- La tabla *Festival*: por aplicación de la regla *RTECAR-1* sobre el tipo de entidad de igual nombre.
- La tabla *Certamen*: por aplicación de la regla *RTECAR-3.1* sobre el tipo de interrelación (*F-C*) y, por tanto, la regla *RTECAR-1* sobre el tipo de entidad de igual nombre.
- La tabla *Premio*: por aplicación de la regla *PRTECAR-4*, por la cual se eliminan los subtipos de entidad *PremioTarea* y *OtroPremio*. Al realizarse esta eliminación, en el tipo de interrelación (*PR-T*) intervienen ahora los tipos de entidad *Tarea* con cardinalidades (0,1) y *Premio* con cardinalidades (1,1).

- La tabla *Festival_Premio*: por aplicación de la regla *RTECAR-4* al tipo de interrelación (*F-PR*).
- La tabla *Otorgo*: por aplicación de las siguientes reglas:
  1. La regla *PRTECAR-3*, por la cual se elimina el supertipo de entidad *Otorgo* considerándose sólo los subtipos *OtorgoPremio* y *OtroOtorgo*. Por ello, los tipos de interrelación *(C-O)* y *(PR-O)* se transfieren a los subtipos como participantes.
  2. La regla *RTECAR-2.2* al tipo de interrelación *(PR-OP)*.
  3. La regla *RTECAR-3.1* al tipo de interrelación *(P-OP)*.

  Como se observa, no se genera una tabla para el tipo de entidad *Otorgo* (o sus subtipos) puesto que los otorgos son premios que se dan, o no, en los certámenes y la información de los mismos ya está considerada en la tabla *Premio*.
- La tabla *Reconocimiento*: se deriva de forma similar a la anterior, con la diferencia de que en este caso se aplica la regla *RTECAR-3.1* sobre el tipo de interrelación (*PE-OO*).
- La tabla *Cine*: por aplicación de la regla *RTECAR-1* sobre el tipo de entidad del mismo nombre.
- La tabla *Sala*: por aplicación de la regla *RTECAR-3.1* sobre el tipo de interrelación (*C-S*) y, por tanto, por aplicación de la regla *RTECAR-1* sobre el tipo de entidad del mismo nombre.
- La tabla *Proyeccion*: por aplicación de la regla *RTECAR-3.*1 sobre los tipos de interrelación (*S-PO*) y (*P-PO*) y, por tanto, por aplicación de la regla *RTECAR-1* sobre el tipo de entidad del mismo nombre.

12.4. CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

El modelo relacional derivado se encuentra normalizado, representándose en la figura 12.4 el diagrama relacional correspondiente al mismo.

12.4.1. Definición sintáctica de las tablas

```
   /* Inicialmente se borran las tablas */
DROP TABLE Trabajo;
DROP TABLE Proyeccion;
DROP TABLE Sala;
DROP TABLE Cine;
DROP TABLE Otorgo;
DROP TABLE Reconocimiento;
DROP TABLE Festival_Premio;
DROP TABLE Pelicula;
DROP TABLE Premio;
DROP TABLE Tarea;
```

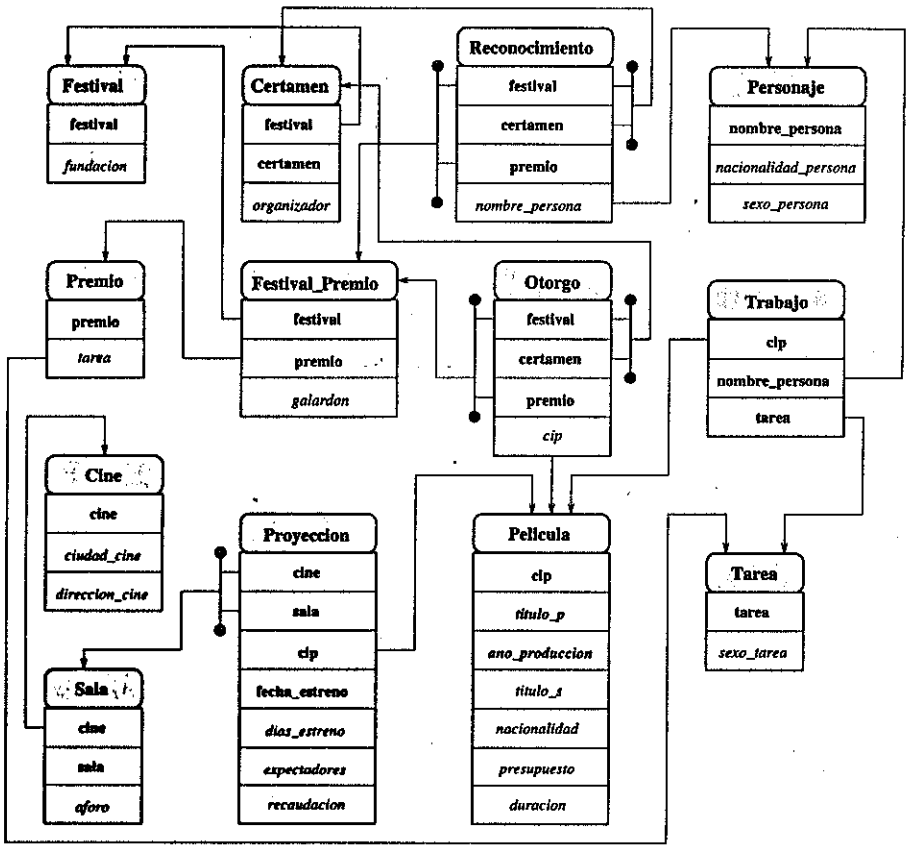


Figura 12.4 Diagrama relacional del problema de las películas

```
DROP TABLE Personaje;
DROP TABLE Certamen;
DROP TABLE Festival;

   /* Se crean las tablas del esquema relacional propuesto */
CREATE TABLE Pelicula  (
   cip VARCHAR2(10) NOT NULL,
   titulo_p VARCHAR2(45) NOT NULL,
   ano_produccion NUMBER(4) NOT NULL,
   titulo_s VARCHAR2(45),
   nacionalidad VARCHAR2(15),
   presupuesto NUMBER(11),
   duracion NUMBER(3),
     CONSTRAINT pk_pelicula
       PRIMARY KEY (cip),
     CONSTRAINT ck_cip
       UNIQUE (titulo_p,ano_produccion),
```



```
     CONSTRAINT ck_pre
     CHECK (presupuesto > 0),
     CONSTRAINT ck_dur
       CHECK (duracion > 0) );
CREATE TABLE Personaje  (
   nombre_persona VARCHAR2(25) NOT NULL,
   nacionalidad_persona VARCHAR2(15),
   sexo_persona CHAR(1),
     CONSTRAINT pk_persona
       PRIMARY KEY(nombre_persona),
     CONSTRAINT ck_npe
       CHECK (nombre_persona=INITCAP(nombre_persona)),
     CONSTRAINT ck_sep
       CHECK (sexo_persona IN ('H','M')) );
CREATE TABLE Tarea  (
   tarea VARCHAR2(30) NOT NULL,
   sexo_tarea CHAR(1),
     CONSTRAINT pk_tarea
       PRIMARY KEY (tarea),
     CONSTRAINT ck_set
       CHECK (sexo_tarea IN ('H','M','N')) );
CREATE TABLE Trabajo  (
   cip VARCHAR2(10) NOT NULL,
   nombre_persona VARCHAR2(25) NOT NULL,
   tarea VARCHAR2(35) NOT NULL,
     CONSTRAINT pk_trabajo
       PRIMARY KEY (cip,nombre_persona, tarea),
     CONSTRAINT fk_Tra_Pel
       FOREIGN KEY (cip)
       REFERENCES Pelicula(cip)
       ON DELETE CASCADE,
     CONSTRAINT fk_tra_per
       FOREIGN KEY (nombre_persona)
       REFERENCES Personaje (nombre_persona)
       ON DELETE CASCADE,
     CONSTRAINT fk_tra_tar
       FOREIGN KEY (tarea)
       REFERENCES Tarea(tarea)
       ON DELETE CASCADE );
CREATE TABLE Festival  (
   festival VARCHAR2(40) NOT NULL,
   fundacion DATE,
     CONSTRAINT pk_festival
       PRIMARY KEY (festival),
     CONSTRAINT ck_fes
       CHECK (festival=INITCAP(festival)) );
CREATE TABLE Certamen  (
   festival VARCHAR2(40) NOT NULL,
```

```
    certamen NUMBER(4) NOT NULL,
    organizador VARCHAR2(60),
     CONSTRAINT pk_certamen
      PRIMARY KEY (festival,certamen),
     CONSTRAINT fk_cer_fes
      FOREIGN KEY (festival)
      REFERENCES Festival (festival)
      ON DELETE CASCADE );
CREATE TABLE Premio  (
   premio VARCHAR2(50),
   tarea VARCHAR2(50),
    CONSTRAINT pk_premio
     PRIMARY KEY (premio),
    CONSTRAINT fk_pre_tar
     FOREIGN KEY (tarea)
     REFERENCES Tarea(tarea) );
CREATE TABLE Festival_Premio  (
   festival VARCHAR2(45) NOT NULL,
   premio VARCHAR2(50) NOT NULL,
   galardon VARCHAR2(50),
    CONSTRAINT pk_fespre
     PRIMARY KEY(festival, premio),
    CONSTRAINT fk_fepr_fes
     FOREIGN KEY (festival)
     REFERENCES Festival(festival)
     ON DELETE CASCADE,
    CONSTRAINT fk_fepr_pre
     FOREIGN KEY (premio)
     REFERENCES Premio(premio)
     ON DELETE CASCADE );
CREATE TABLE  Otorgo  (
   festival VARCHAR2(50) NOT NULL,
   certamen NUMBER(4) NOT NULL,
   premio VARCHAR2(40) NOT NULL,
   cip VARCHAR2(10) NOT NULL,
    CONSTRAINT pk_otorgo
     PRIMARY KEY (festival, certamen, premio),
    CONSTRAINT  fk_otor_fepr
     FOREIGN KEY (festival,premio)
     REFERENCES Festival_Premio (festival, premio)
     ON DELETE CASCADE,
    CONSTRAINT  fk_otor_cer
     FOREIGN KEY (festival, certamen)
     REFERENCES Certamen (festival, certamen)
     ON DELETE CASCADE );
CREATE TABLE Reconocimiento  (
   festival VARCHAR2(45) NOT NULL,
   certamen NUMBER(4) NOT NULL,
```



```
   premio VARCHAR2(40) NOT NULL,
   nombre_persona VARCHAR2(35) NOT NULL,
    CONSTRAINT pk_recono
     PRIMARY KEY (festival, certamen, premio),
    CONSTRAINT fk_reco_fepr
     FOREIGN KEY (festival, premio)
     REFERENCES Festival_Premio (festival,premio)
     ON DELETE CASCADE,
    CONSTRAINT fk_reco_cer
     FOREIGN KEY (festival, certamen)
     REFERENCES Certamen (festival, certamen)
     ON DELETE CASCADE,
    CONSTRAINT fk_reco_per
     FOREIGN KEY (nombre_persona)
     REFERENCES Personaje (nombre_persona)
     ON DELETE CASCADE );
CREATE TABLE Cine  (
   cine VARCHAR2(25) NOT NULL,
   ciudad_cine VARCHAR2(25),
   direccion_cine VARCHAR2(65),
    CONSTRAINT pk_cine
     PRIMARY KEY (cine),
    CONSTRAINT ck_cin
     CHECK (cine=INITCAP(cine)));
CREATE TABLE Sala  (
   cine VARCHAR2(25) NOT NULL,
   sala NUMBER(2) NOT NULL,
   aforo NUMBER(4),
    CONSTRAINT pk_sala
     PRIMARY KEY (cine,sala),
    CONSTRAINT fk_sal_cin
     FOREIGN KEY (cine)
     REFERENCES Cine(cine)
     ON DELETE CASCADE,
    CONSTRAINT ck_afr
     CHECK (aforo > 0) );
CREATE TABLE Proyeccion  (
   cine VARCHAR2(25) NOT NULL,
   sala NUMBER(2) NOT NULL,
   cip VARCHAR2(10) NOT NULL,
   fecha_estreno DATE NOT NULL,
   dias_estreno NUMBER(3),
   espectadores NUMBER(6),
   recaudacion NUMBER(8),
    CONSTRAINT pk_proyección
     PRIMARY KEY (cine,sala, cip, fecha_estreno),
    CONSTRAINT fk_pro_sal
     FOREIGN KEY (cine,sala)
     REFERENCES Sala(cine,sala)
```

```
  ON DELETE CASCADE,
 CONSTRAINT fk_pro_pel
  FOREIGN KEY (cip)
  REFERENCES Pelicula (cip)
  ON DELETE CASCADE,
 CONSTRAINT ck_dia
  CHECK (dias_estreno>0) );
```

12.4.2. Manipulación de la Base de Datos

Cap12ej01.sql

El ministerio de cultura desea realizar un estudio sobre la distribución del cine escocés en nuestro país. Obtener todas las salas que han emitido películas escocesas en el período comprendido entre dos meses antes del 07 de diciembre del año 1996.

De la tabla Pelicula se obtiene el atributo cip de todas las películas que cumplen la condición de que su atributo nacionalidad tenga el valor 'Escocia' y que, gracias a la función MONTHS_BETWEEN, la diferencia de tiempo transcurrida entre el valor del atributo fecha_estreno y el día 07/12/1996 sea menor o igual a 2. El resultado de esta subconsulta se utiliza para obtener de la tabla Proyeccion los atributos cine y sala de todas las tuplas que cumplan la condición de que el cip sea igual al valor obtenido anteriormente.

```
SELECT cine "Nombre del cine", sala "Sala"
   FROM Proyeccion
   WHERE cip IN
     (SELECT cip
      FROM Pelicula
      WHERE nacionalidad='Escocia') AND MONTHS_BETWEEN
            (TO_DATE('07/12/1996', 'DD/MM/YYYY'),
            fecha_estreno) <= 2;
```

Resultado

```
Nombre del cine                Sala
------------------------- ----------
Los Arcos2                          3
```

Cap12ej02.sql

Obtener todas las películas en las que el director ha trabajado también como actor.

Se reúnen las tablas Trabajo y Pelicula para obtener los atributos titulo_p, ano_produccion y nombre_persona; el resultado de esta reunión es restringido con las condiciones de reunión, y quedándonos con aquellas filas en las que el nombre_persona equivale al director. Después el atributo nombre_persona es utilizado con el operador IN con una subconsulta que devolverá para la película que dirige la persona identificada con nombre_persona, los nombres de personas que realizan una labor distinta a Director y Productor.

```
   /* Se formatea la salida */
COLUMN titulo_p FORMAT A30
SELECT titulo_p, ano_produccion "FECHA", nombre_persona
   FROM Trabajo Text, Pelicula
```



```
   WHERE Pelicula.cip=Text.cip AND tarea='Director'
         AND nombre_persona IN
      (SELECT nombre_persona
       FROM Trabajo Tint
       WHERE tarea <> 'Productor' AND tarea <> 'Director'
             AND Tint.cip=Text.cip);
```

Resultado

```
TITULO_P                       FECHA NOMBRE_PERSONA
------------------------------ ----- ---------- ----------------
El proyecto de la Bruja Blair   1999 Gary Oldman
Viaje de novios a la Italia     1997 Matt Leblanc
```

Cap12ej03.sql

Obtener todos los premios obtenidos por películas dirigidas por Isaac Hayes en los Oscars.

Se realiza una reunión de las tablas Peliculas y Otorgo sobre un atributo común y con la condición de que el atributo cip se encuentre en el resultado de proyectar este atributo sobre la selección de la tabla Tarea con las condiciones del enunciado.

```
   /* Se formatea la salida */
COLUMN Película FORMAT A25
COLUMN Festival FORMAT A8
COLUMN Certamen FORMAT A4
COLUMN Premio FORMAT A25
SELECT titulo_p "Película", festival "Festival",
       certamen " Año", premio "Premio"
   FROM Pelicula, Otorgo
   WHERE Pelicula.cip=Otorgo.cip AND Festival='Oscars'
         AND  Otorgo.cip IN
      (SELECT cip
         FROM Trabajo
         WHERE nombre_persona='Isaac Hayes'
               AND tarea='Director');
```

Resultado

```
Película                  Festival  Año Premio
------------------------- -------- ---- ----- --------------------
Soledad en las montañas   Oscars   1999 Mejor Montaje
Soledad en las montañas   Oscars   1999 Mejor dirección artística
Soledad en las montañas   Oscars   1999 Mejor fotografía
Soledad en las montañas   Oscars   1999 Mejor guión
```

Cap12ej04.sql

Obtener la relación de los actores principales que han obtenido 'Oscars' y el número de ellos.

De la tabla Trabajo se obtiene el atributo nombre_persona sobre el que se agrupa y mediante el operador de grupo COUNT se obtiene el número de tuplas de cada grupo. Los grupos deberán cumplir que el premio obtenido sea 'Oscars', lo cual se consigue igualando este valor al atributo galardon de la tabla Festival_Premio. Aparte se deberá igualar el atributo tarea de la tabla Trabajo a 'Actor Principal'.

```
   /* Se formatea la salida */
COLUMN nombre_persona FORMAT A25
SELECT Trabajo.nombre_persona, COUNT(*) "NÚMERO"
```

```
FROM Festival_Premio, Reconocimiento, Trabajo
WHERE Reconocimiento.nombre_persona=Trabajo.nombre_persona
      AND Reconocimiento.premio=Festival_Premio.premio
      AND Reconocimiento.festival=Festival_Premio.festival
      AND tarea='Actor Principal'
      AND galardon='Oscar'
GROUP BY Trabajo.nombre_persona
ORDER BY 2;
```

Resultado

NOMBRE_PERSONA	NÚMERO
Charlie Sheen	2
Jonh Cusack	2
Sam Waterston	2
Sean Connery	2
Eddie Murphy	4

Cap12ej05.sql

¿Existe alguna película en la base de datos con un presupuesto superior al de 'Viaje al centro de la tierra' y producida en 1995?

De la tabla Pelicula se obtiene el atributo presupuesto de la tupla correspondiente a 'Viaje al centro de la tierra', estrenada en 1995. El resultado de esta consulta se utiliza para obtener, de nuevo de la tabla Pelicula, los atributos titulo_p y ano_produccion de todas aquellas tuplas que cumplan la condición de que su atributo presupuesto sea superior al obtenido en la consulta anterior. Si la consulta no genera ninguna tupla, significa que 'Viaje al centro de la tierra' es la película de presupuesto más elevado de las almacenadas en la base de datos.

```
SELECT titulo_p "Título", ano_produccion "Fecha Producción"
   FROM Pelicula
   WHERE presupuesto >
     (SELECT presupuesto
        FROM Pelicula
        WHERE titulo_p='Viaje al centro de la tierra'
              AND ano_produccion=1995);
```

Resultado

Título	Fecha Producción
El fin de los días	1999

Cap12ej06.sql

Obtener el número de salas que tiene cada cine de Madrid.

De la tabla Sala se obtiene el atributo cine y el resultado de contabilizar todas las tuplas con el mismo valor de cine y distinto de sala, siempre y cuando cine tenga los valores obtenidos en una subconsulta sobre la tabla Cine en la que se obtiene el atributo cine de todas las tuplas en que ciudad_cine tiene el valor 'Madrid', es decir, todas las salas de un mismo cine. Es necesario agrupar el resultado por cine para que la consulta sea correcta.

```
SELECT Cine "Cine", COUNT(sala) "Salas"
   FROM Sala
   WHERE Cine IN
```



```
     (SELECT Cine
        FROM Cine
        WHERE ciudad_cine='Madrid')
   GROUP BY cine;
```

Resultado

Cine	Salas
Los Arcos	5
Los Arcos2	5

Cap12ej07.sql

Obtener una relación de los premios que han quedado desiertos en los 'Oscars' de 1997.

En primer lugar se obtienen todos los premios concedidos en el certamen de 1997 de los 'Oscars' de las tablas Otorgo y Reconocimiento a partir de su unión,y la solución se obtiene de restar estas tuplas a las tuplas obtenidas de la reunión de las tablas Certamen y Festival_Premio, mediante la cual se obtienen todos los premios que hubieran sido posible otorgar.

```
/* Se formatea la salida */
COLUMN Festival FORMAT A30
COLUMN Certamen FORMAT 9999
SELECT Certamen.festival "Festival", certamen "Certamen",
       premio "Premio"
   FROM Festival_Premio, Certamen
   WHERE Festival_Premio.festival=Certamen.festival
         AND Festival_Premio.Festival='Oscars'
         AND Certamen.certamen=1997
         AND (premio) NOT IN
    (SELECT premio
     FROM Otorgo
     WHERE Festival='Oscars' AND certamen=1997
    UNION
    SELECT premio
     FROM Reconocimiento
     WHERE Festival='Oscars' AND certamen=1997)
  ORDER BY Certamen.festival, Certamen.certamen;
```

Resultado

Festival	Certamen	Premio
Oscars	1997	Mejor actriz secundaria

Cap12ej08.sql

Borrar todas las películas que no han recibido ningún galardón.

Se borran de la tabla Pelicula todas aquellas tuplas que cumplan la condición de que el valor del atributo cip no coincide con los obtenidos como resultado de la subconsulta sobre la tabla Otorgo del atributo cip de todas las tuplas existentes; es decir, de todas las películas que han resultado premiadas.

```
DELETE
   FROM Pelicula
   WHERE (cip) NOT IN
```

```
   (SELECT DISTINCT cip
      FROM Otorgo);
```

Resultado

```
   4 filas borradas.
```

Cap12ej09.sql

Crear una vista que permita consultar todas las personas que han trabajado en cada película.

Se creará una vista sobre:

- *El atributo titulo_p de la tabla Pelicula.*
- *El atributo nombre_persona de la tabla Trabajo.*
- *El atributo tarea de la tabla Trabajo.*

Las tres consultas que componen la vista se reunirán bajo la condición de que el atributo cip tome el mismo valor para las dos tablas consideradas. Por último, la solución se presentará ordenada por el título de las películas.

```
CREATE VIEW Plantilla AS
   SELECT nombre_persona, tarea, titulo_p,
          ano_produccion "Película"
      FROM Pelicula, Trabajo
      WHERE Pelicula.cip=Trabajo.cip;
```

Resultado

```
   Vista creada.
```

Cap12ej10.sql

Eliminar la vista anteriormente creada.

Para eliminar la vista Plantilla, basta con utilizar el comando DROP VIEW, usando como parámetro el nombre de la vista que se desea eliminar.

```
DROP VIEW Plantilla;
```

Resultado

```
   Vista borrada.
```

Cap12ejP1.sql

Realizar un procedimiento PL/SQL que muestre por pantalla el historial de una persona, cuyo nombre será introducido como argumento del procedimiento. En la salida por pantalla se deberá indicar en todo lo que ha participado dicha persona, así como los premios que ha obtenido.

Este procedimiento se fundamenta en dos métodos del paquete DBMS_OUTPUT que permiten realizar operaciones de E/S por pantalla desde un procedimiento. Los dos procedimientos de este paquete usados son:

- *ENABLE(numero), el cual permite establecer el tamaño del buffer.*
- *PUT_LINE(number|date|string) el cual permite mostrar el argumento por pantalla.*

La información a visualizar será obtenida mediante dos cursores basados en consultas los cuales serán recorridos con dos bucles FOR.

```
CREATE OR REPLACE PROCEDURE Informe_Persona
-- Se definen las variables de la interfaz del procedimiento
   (p_nombre_persona VARCHAR2)
```



```
AS
-- Cursor que permite obtener en qué películas ha trabajo una
persona
   CURSOR participaciones_peliculas IS
     SELECT P.titulo_p,T.tarea,P.ano_produccion
       FROM Pelicula P, Trabajo T
       WHERE P.cip=T.cip AND nombre_persona=p_nombre_persona
       ORDER BY P.ano_produccion;
-- Cursor que permite obtener los premios alcanzados en cada
festival
   CURSOR premios_obtenidos  IS
     SELECT P_P.nombre_persona, premio, Festival, Certamen,
            P_P.titulo_p
       FROM Reconocimiento,
            (SELECT T.nombre_persona,P.titulo_p,
                    P.ano_produccion fecha
               FROM Trabajo T, Pelicula P
               WHERE T.cip=P.cip) P_P
       WHERE P_P.nombre_persona=p_nombre_persona
            AND Reconocimiento.nombre_persona=P_P.nombre_persona
            AND Certamen=P_P.fecha;
BEGIN
-- Procedimiento que me modifica el tamaño del buffer
   DBMS_OUTPUT.ENABLE(100000);
-- Se imprime la cabecera de la salida
   DBMS_OUTPUT.PUT_LINE('-----------------------------------');
   DBMS_OUTPUT.PUT_LINE('-----------------------------------');
   DBMS_OUTPUT.PUT_LINE('Trabajos:   '||p_nombre_persona);
   DBMS_OUTPUT.PUT_LINE('-----------------------------------');
   DBMS_OUTPUT.PUT_LINE('-----------------------------------');
-- Recorrido del primer cursor
   FOR v1 IN participaciones_peliculas LOOP
     DBMS_OUTPUT.PUT_LINE(' ');
     DBMS_OUTPUT.PUT_LINE('Película:  ' || v1.titulo_p);
     DBMS_OUTPUT.PUT_LINE(' ');
     DBMS_OUTPUT.PUT_LINE('Fecha:     ' || v1.ano_produccion);
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('-----------------------------------');
   DBMS_OUTPUT.PUT_LINE('-----------------------------------');
   DBMS_OUTPUT.PUT_LINE('Premios:    ' || p_nombre_persona);
   DBMS_OUTPUT.PUT_LINE('-----------------------------------');
   DBMS_OUTPUT.PUT_LINE('-----------------------------------');
-- Recorrido del segundo cursor
   FOR v2 IN premios_obtenidos LOOP
     DBMS_OUTPUT.PUT_LINE(' ');
     DBMS_OUTPUT.PUT_LINE('Festival: ' || v2.festival);
     DBMS_OUTPUT.PUT_LINE(' ');
     DBMS_OUTPUT.PUT_LINE('Certamen: ' || v2.certamen);
     DBMS_OUTPUT.PUT_LINE(' ');
     DBMS_OUTPUT.PUT_LINE('Película: ' || v2.titulo_p);
```

```
    DBMS_OUTPUT.PUT_LINE(' ');
    DBMS_OUTPUT.PUT_LINE('Premio:  ' || v2.premio);
    DBMS_OUTPUT.PUT_LINE(' ');
  END LOOP;
-- Se comprueban los posibles errores del proceso
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

Prueba de ejecución

```
-- Se ejecuta el procedimiento
SET SERVEROUTPUT ON
    EXEC informe_persona ('Maribel Verdú');
SET SERVEROUTPUT OFF
```

CAPÍTULO 13

LA FIESTA NACIONAL

13.1. ENUNCIADO DEL PROBLEMA

Una asociación de ámbito nacional de aficionados a la Fiesta Nacional —las lidias de los toros— desea desarrollar un sistema de información que permita tener informados a sus afiliados, tanto de los festejos taurinos que se van a producir en cada temporada, como de los ya realizados en temporadas precedentes.

Es interesante, para esta asociación, mantener información de las plazas en las que se realizan festejos taurinos, los toreros que intervinieron o van a intervenir, las cuadrillas de subalternos, los premios que obtuvieron, los toros que se lidiaron, las ganaderías participantes, etc. En definitiva, toda la información correspondiente y relevante en cada uno de estos acontecimientos, tanto en lo referente a la fiesta en sí, como al precio de las entradas y asistencia de público a los mismos.

Se trata de un problema complejo en el cual se necesita la introducción de los siguientes supuestos semánticos para definirlo más claramente:

SUPUESTO 1: Se van a considerar únicamente las fiestas taurinas que se celebran en España, en cualquier localidad del territorio nacional, y no aquellas que se celebren o se puedan celebrar fuera de nuestras fronteras.

SUPUESTO 2: Sólo interesa conocer aquellas plazas, ganaderías, toreros y subalternos que hayan participado en alguna fiesta sobre la cual se tenga información.

SUPUESTO 3: Cada plaza de toros tiene asignada una categoría, tiene un nombre único, está ubicada en una localidad, y tiene un aforo y un apoderado que la dirige.

SUPUESTO 4: El aforo de la plaza está dividido, en cada una de ellas, por zonas, de forma que los asientos o localidades de cada zona tienen, en cada festejo, un precio determinado. Generalmente, las zonas son de *sol* y *sombra*, y dentro de éstas, existe división en función de la distancia del asiento a la *arena* en la que se realiza la lidia de los toros.

SUPUESTO 5: Las fiestas pueden ser de varias categorías: toros, novillos con o sin picador, rejoneo y mixtas, pudiendo variar en cada una de ellas el número de reses que se lidian y el número de matadores[30] que intervienen.

SUPUESTO 6: El número de ganaderías que presentan los toros en cada fiesta puede variar de una fiesta a otra.

SUPUESTO 7: Una res pertenece a una sola ganadería. Cada res tiene asignado un nombre por la ganadería, aunque distintas ganaderías pueden dar el mismo nombre a sus reses. No es usual que una ganadería repita el nombre que le asigna a las suyas.

SUPUESTO 8: Cada res tiene asignado un número que puede repetirse en reses lidiadas en distintas plazas. Cuando se lidia una res, es interesante conocer su peso, color y una serie de características que aportan información sobre su *nobleza* y *bravura*.

SUPUESTO 9: Cada ganadería tiene un nombre único, pertenece a un único propietario o sociedad, tiene unos colores o enseña y un hierro diferente para marcar sus reses.

SUPUESTO 10: Cada matador tiene una cuadrilla de subalternos que participan con él en la lidia de las reses. Los subalternos están catalogados en banderilleros, quitadores y picadores. Un matador puede tener, en su cuadrilla, un número variable de subalternos de cada tipo, siendo interesante conocer la cuadrilla actual de cada matador así como la que participó con él en las fiestas sobre las que se tiene información, pues en ocasiones, no es la cuadrilla actual.

SUPUESTO 11: En cada fiesta un matador puede lidiar un número variable de reses, alcanzando, o no, un premio en cada una de las lidias que realiza. Los premios más usuales que se les da a los toreros son: pitos, silencio (no son premios muy buenos), aplausos, vuelta al ruedo, petición de oreja, oreja, dos orejas, dos orejas y rabo, entre otros. Además, por la labor global en la fiesta, a los matadores se les *puede sacar a hombros y por la puerta grande de la plaza* como premio.

[30] Aunque no sea un término adecuado, inicialmente vamos a denominar por este término a cualquier participante en una lidia de toros.

SUPUESTO 12: Cada matador tiene un apoderado, que gestiona las apariciones del mismo. Los subalternos no tienen apoderado.

SUPUESTO 12+1: No se celebra en el mismo día más de una lidia de toros en una plaza, aunque sí en plazas diferentes.

SUPUESTO 14: El número máximo de reses que se puede lidiar en una fiesta es seis. Si excepcionalmente se lidiaran más, se deberá tener información del resto de los toros.

SUPUESTO 15: Aunque sólo se lidien seis reses en una fiesta es necesario conocer información sobre las reses sobreras o reservas que están disponibles y, a veces, se lidian en sustitución de las reses principales.

SUPUESTO 16: Las reses de reserva que no se lidien en una fiesta pueden ser lidiadas, o ser sobreras, o no, en alguna otra fiesta. Las reses que sean lidiadas en una fiesta no se lidiarán ni serán sobreras en ninguna otra fiesta aunque no se les haya dado muerte.

13.2. MODELO CONCEPTUAL

Como se ha venido haciendo en los capítulos precedentes vamos a ir avanzando en el desarrollo del modelo conceptual paso a paso, describiendo en primer lugar aquellos tipos de entidad e interrelaciones más evidentes para, en un segundo paso, analizar con profundidad el resto de los objetos presentes en el problema propuesto.

13.2.1. Las plazas, fiestas, ganaderías y las reses

Son las plazas y los festejos taurinos que se celebran en ellas los objetos principales del problema. Se puede considerar un tipo de entidad *Plazas* con identificador *nombre_plaza*, en base al SUPUESTO 3, y con atributos *cat_plaza, aforo_plaza, localidad_plaza* y *apoderado_plaza*, representando, respectivamente, la categoría, aforo, localidad y el apoderado de la plaza en cuestión (SUPUESTOS 1, 3).

En las plazas se celebran festejos taurinos. Estos festejos se repiten a lo largo del año y de un año a otro. Entonces, se puede considerar un tipo de entidad *Fiestas* que represente a estos festejos taurinos; un tipo de entidad débil por identificación con respecto al tipo de entidad *Plazas* por no existir un identificador único y apropiado para este tipo de entidad si se considera que en una misma fecha se pueden celebrar varios festejos (SUPUESTO 12+1) y que las fiestas no tienen un nombre único.

Por ello, los atributos de este tipo de entidad serán: la agregación del atributo *nombre_plaza*, heredado del tipo de entidad *Plazas*, y el atributo *fecha_fiesta* como identificador. Además de los atributos *clase_fiesta, hora_fiesta* y *exp_fiesta* para representar el tipo de festejo de que se trata, la hora del día en que se desarrolla y los espectadores que asistieron al mismo.

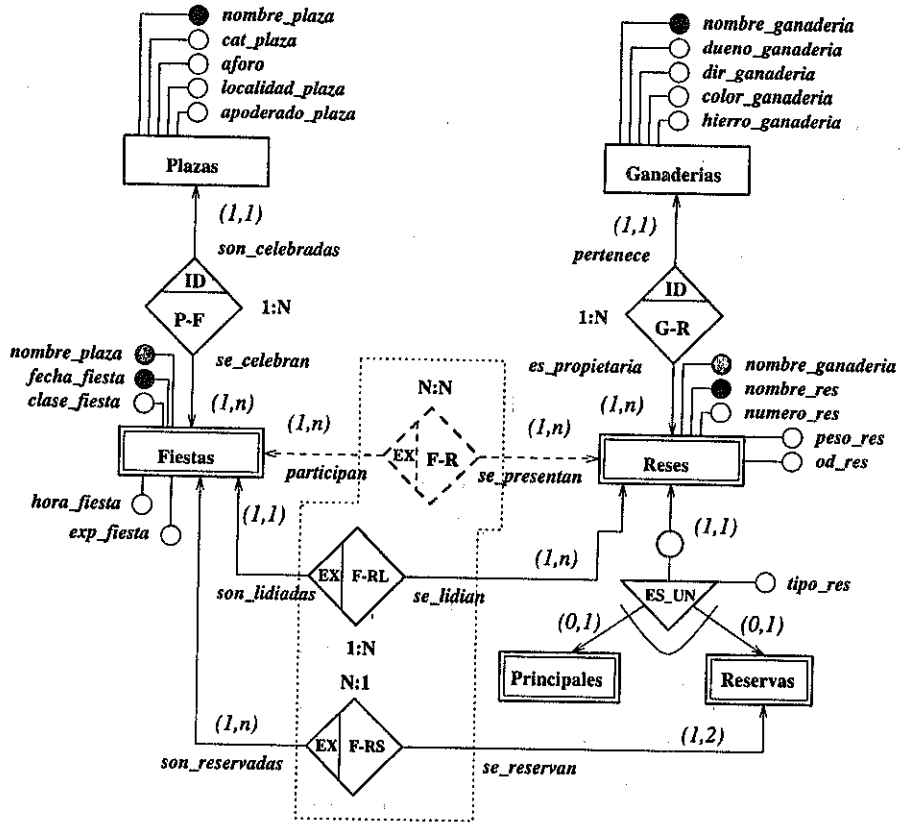


Figura 13.1 Las plazas, fiestas, ganaderías y reses

Se puede extraer del problema propuesto (SUPUESTOS 5-9) la existencia de otros dos tipos de entidad que describen las empresas que aportan las reses para su lidia en los festejos taurinos y las reses que se lidian en los mismos.

Como muestra la figura 13.1 el tipo de entidad *Ganaderias* mantiene una relación *1:N* con el tipo de entidad *Reses* puesto que una res sólo puede pertenecer a una ganadería, como así indica el SUPUESTO 7.

Debido al SUPUESTO 2, el tipo de entidad *Reses* es débil por existencia con respecto al tipo de entidad *Fiesta*. Pero, además, el tipo de entidad *Reses* es débil por identificación con respecto al tipo de entidad *Ganaderias* sobre la base del enunciado del SUPUESTO 7. Los atributos de estos tipos de entidad vienen, en su mayor parte, definidos por los SUPUESTOS 6-9, así:

Tipo de entidad *Ganaderias*: con identificador igual a *nombre_ganaderia*, y con atributos, *dueno_ganaderia, dir_ganaderia, color_ganaderia* y *hierro_ganaderia*.

Tipo de entidad *Reses*: cuyo identificador será la agregación de los atributos *nombre_ganaderia*, heredado del tipo de entidad *Ganaderias* por su relación de debilidad, y *nombre_res*. Otros atributos serán: *numero_res, peso_res* y *od_res* que representan, respectivamente, el número asignado a la res, su peso y cualquier otra información que se quiera representar.

Como se muestra en la figura 13.1, el tipo de entidad *Reses* ha sido especializado en dos tipos de entidad: *Principales* y *Reservas*, las razones son las siguientes:

- En cada fiesta hay dos tipos de reses, aquellas que se tienen preparadas para la lidia y que se lidian aunque se puedan devolver a los corrales, y aquellas que están en reserva por si alguna de las reses principales es ***devuelta a los corrales*** y hay que utilizarla.
- El tipo de interrelación *(F-R)* representa las reses que están presentes en un festejo, pero no aporta información de cuál de ellas es lidiada (SUPUESTOS 14 y 15).
- Según el SUPUESTO 16, la relación que existe entre las reses que se lidian y las que no es muy diferente con respecto a las fiestas en las que participan.

Por otro lado y, como también se muestra en la figura 13.1, el tipo de entidad *Reses* es débil por existencia con respecto al tipo de entidad *Fiestas* según el SUPUESTO 2. Se presentan en la figura 13.1 tres tipos de interrelación:

Tipo de interrelación *(F-R)*: la cual representa los toros que se preparan para cada uno de los festejos taurinos (representada en línea de puntos), y considerando a todas las reses, tanto las principales como las reservas. Es la especialización del tipo de entidad *Reses* la que permite distinguir entre el papel asignado a cada una de las reses antes de que se produzca la lidia de las mismas.

Tipo de interrelación *(F-RL)*: la cual representa a las reses que se lidian en un festejo con independencia del papel que tenían asignado con anterioridad.

Tipo de interrelación *(F-RS)*: representando las reses que no son lidiadas en el festejo y, por tanto, pueden participar en otros festejos.

Más adelante volveremos a tratar estos tipos de interrelación, debido a que otros tipos de entidad que surgirán en el desarrollo del modelo conceptual introducirán en ellos modificaciones necesarias de realizar.

13.2.2. Los asientos y los precios

Según el SUPUESTO 4 las plazas tienen diferentes categorías de localidades para el público que asiste a las fiestas taurinas y, en función de estos tipos, existen unos precios para cada fiesta (ver el tipo de entidad *Plazas* descrito en la sección anterior).

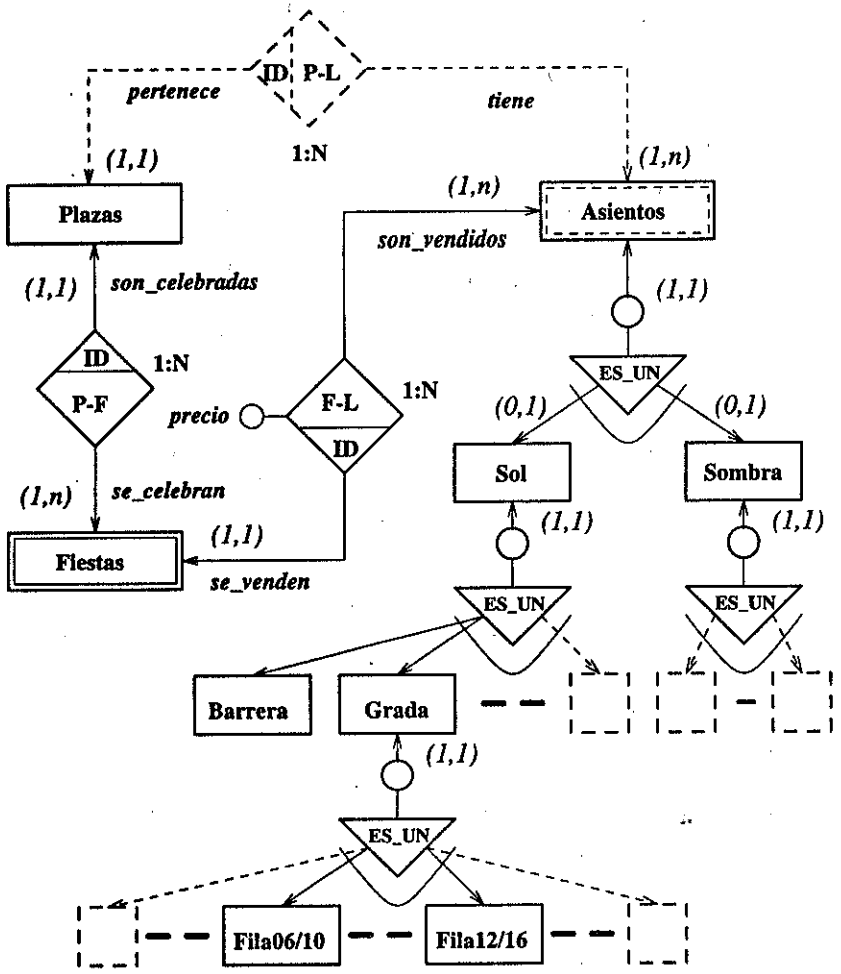


Figura 13.2 Los asientos y los precios

Es necesario, por tanto, considerar un tipo de entidad denominado *Asientos* que represente a estos tipos de localidades distintos. Este tipo de entidad, como se muestra en la figura 13.2, debe refinarse en los distintos tipos de asientos y, como se muestra también en la misma figura, los subtipos obtenidos deben a su vez ser refinados[31].

Los asientos de las plazas están categorizados en tipos generales como sol y sombra. Para cada tipo general se clasifican en subtipos como: barrera, contrabarrera, etc., y para cada uno de estos subtipos se vuelven a clasificar en otros subtipos en

[31] Se ha simplificado la especialización del tipo de entidad *Asientos* con el fin de no complicar innecesariamente el diagrama que representa el modelo conceptual.

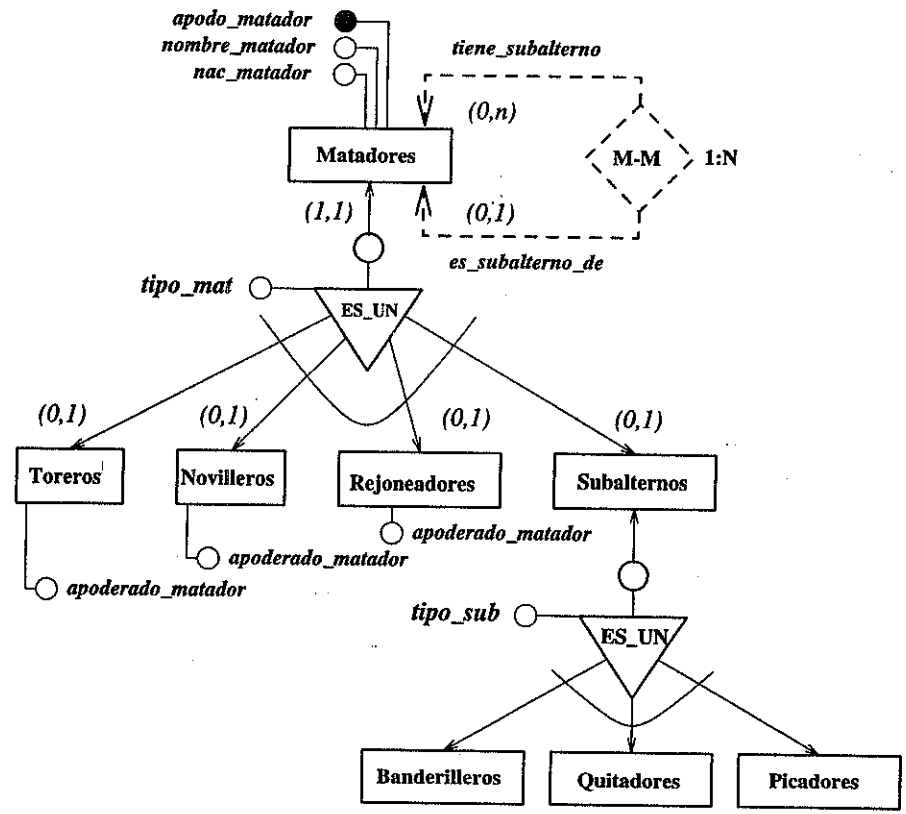


Figura 13.3 Los matadores de toros

función de la fila en la que se encuentra el asiento o, lo que es lo mismo, la distancia del mismo al ruedo.

Para este tipo de entidad se va a considerar únicamente un atributo *asiento#* que representa el tipo del asiento, grupo de filas, etc.

Por otro lado, y como así introduce el SUPUESTO 4, los precios de estos grupos de asientos cambian para cada festejo, luego existe una relación entre el tipo de entidad *Asientos* y el tipo de entidad *Fiestas* que ha sido denominado *(F-L)* (ver figura 13.2) que viene caracterizado por el atributo *precio*, representando, para cada festejo, los precios que tienen los distintos tipos de asientos.

La relación existente entre el tipo de entidad *Fiestas* y *Asientos*, debido a la debilidad de identificación de la primera con respecto al tipo de entidad *Plazas*, permite representar la debilidad por identificación de los asientos con respecto a las plazas. No se representa una relación directa entre los tipos de entidad *Plazas* y *Asientos* porque estaríamos violando el SUPUESTO 2, que expone que no se debe tener

información de plazas y, por tanto, ni de sus asientos, en las que no se realicen festejos taurinos.

La no-existencia de una relación directa entre plazas y asientos implicará que mientras no se tenga constancia de un festejo que se va a celebrar o que se ha celebrado en una plaza, no se tiene conocimiento de las localidades para el público de esa plaza y que, por tanto, puede variar de un festejo a otro la disposición de los grupos de asientos. Si éste no fuera el caso bastaría con definir un tipo de interrelación entre los tipos de entidad *Plazas* y *Asientos* (ver en la figura 13.2 el tipo de interrelación *(P-L)* representada en líneas discontinuas).

13.2.3. Los matadores de toros

Los matadores (toreros, novilleros, rejoneadores y subalternos) son los otros objetos a considerar en el problema. Según el SUPUESTO 2, no se tendrá información de los matadores a no ser que hayan participado o vayan a participar en alguno de los festejos sobre los que se tenga información.

Podemos definir un tipo de entidad *Matadores*, como se muestra en la figura 13.3, que represente a este conjunto de objetos y, podemos refinarla en un conjunto de subtipos que diferencian las diferentes categorías de los mismos.

Además, hay que tener en cuenta que los subalternos están a su vez catalogados (SUPUESTO 10), por lo que es necesario especializar este subtipo de entidad como así se muestra en la figura 13.3.

Para el tipo de entidad *Matadores* se van a considerar los atributos: *apodo_matador* (si alguno no lo tiene se usará su nombre), *nombre_matador* y *nac_matador*, representando el apodo asignado, su nombre completo y el lugar de nacimiento, respectivamente. Además, los subtipos *Toreros, Novilleros* y *Rejoneadores* tendrán un atributo más que representa al apoderado de los mismos (los subalternos no tienen apoderado, según el SUPUESTO 12).

Elegir un identificador para este tipo de entidad es complejo, pues pueden existir matadores con el mismo apodo e incluso con el mismo nombre. Para no complicar más el problema introduciremos el siguiente supuesto:

SUPUESTO 17: Los matadores de toros tienen un apodo único. Si dos matadores tuvieran un mismo apodo, a alguno de ellos se le modificará el mismo asociándole un ordinal (ejemplo: I, II, etc.).

Vamos a analizar a continuación los tipos de interrelación que mantienen estos tipos de entidad:

Los toreros y los subalternos: según el SUPUESTO 10 es necesario conocer la cuadrilla de subalternos de cada uno de los matadores (a excepción de los subalternos), por lo que existe un tipo de interrelación reflexiva en el tipo de entidad *Matadores* en el que el tipo de entidad *Matadores*, en su papel de subalterno, participa con cardinalidades *(0,n)* puesto que los subalternos no tienen, a su vez, subalternos. Se puede observar que no es una representación muy adecuada y, por lo tanto, debería substituirse por tres tipos de interrelación en las que intervendría el subtipo de entidad *Subalternos* y los otros tres subtipos de entidad de *Matadores*. Por lo que más adelante en el desarrollo se propondrá una solución más adecuada.

El cartel de las fiestas: cuando se programa una fiesta existe una previsión o cartel de los matadores que van a participar en la misma. Casi siempre son éstos los matadores que lidian las reses pero, en ocasiones, esto no ocurre y hay un cambio de última hora.

Como el enunciado del problema informa de que se desea conocer con anterioridad las fiestas que se van a realizar, es necesario introducir un tipo de interrelación *(F-M)* que represente el cartel previsto para una determinada fiesta, como se muestra en la figura 13.4.

En este tipo de interrelación, el tipo de entidad *Matadores* es débil por existencia sobre la base del SUPUESTO 2.

La lidia en la fiesta: una vez celebrada la fiesta, en la que se ha lidiado un número de toros variable (generalmente seis, según el SUPUESTO 14), cada uno de ellos lidiado por un matador. Es necesario representar esta relación que además debe considerar el premio que alcanza cada matador en la *faena* realizada (SUPUESTO 11).

Existirá, por tanto, un nuevo tipo de interrelación entre los tipos de entidad *Fiestas*, *Matadores* y *Reses* (sólo las lidiadas) para representar esta información. La figura 13.4 muestra la solución adoptada para una representación simplificada del problema.

Si se considera que los matadores que intervienen en una fiesta son considerados, con independencia del cartel, a que realizan la lidia de algún toro, entonces existirá un tipo de interrelación entre *Matadores* y *Reses*. Pero se necesita al mismo tiempo informar del resultado de cada matador en cada fiesta con independencia del resultado obtenido en cada lidia (SUPUESTO 11). Y, además, se necesita conocer los subalternos de cada matador en cada fiesta, con independencia de su cuadrilla actual (SUPUESTO 10).

Por todo ello, es necesario definir los siguientes objetos en el modelo conceptual:

1. Generar un tipo de entidad débil por identificación con respecto a *Matadores* que represente a aquellos matadores que realizan faenas en una fiesta. Este tipo de entidad al que denominaremos *Apariciones* mantiene un tipo de interrelación 1:1 con respecto al tipo de entidad *Matadores*.

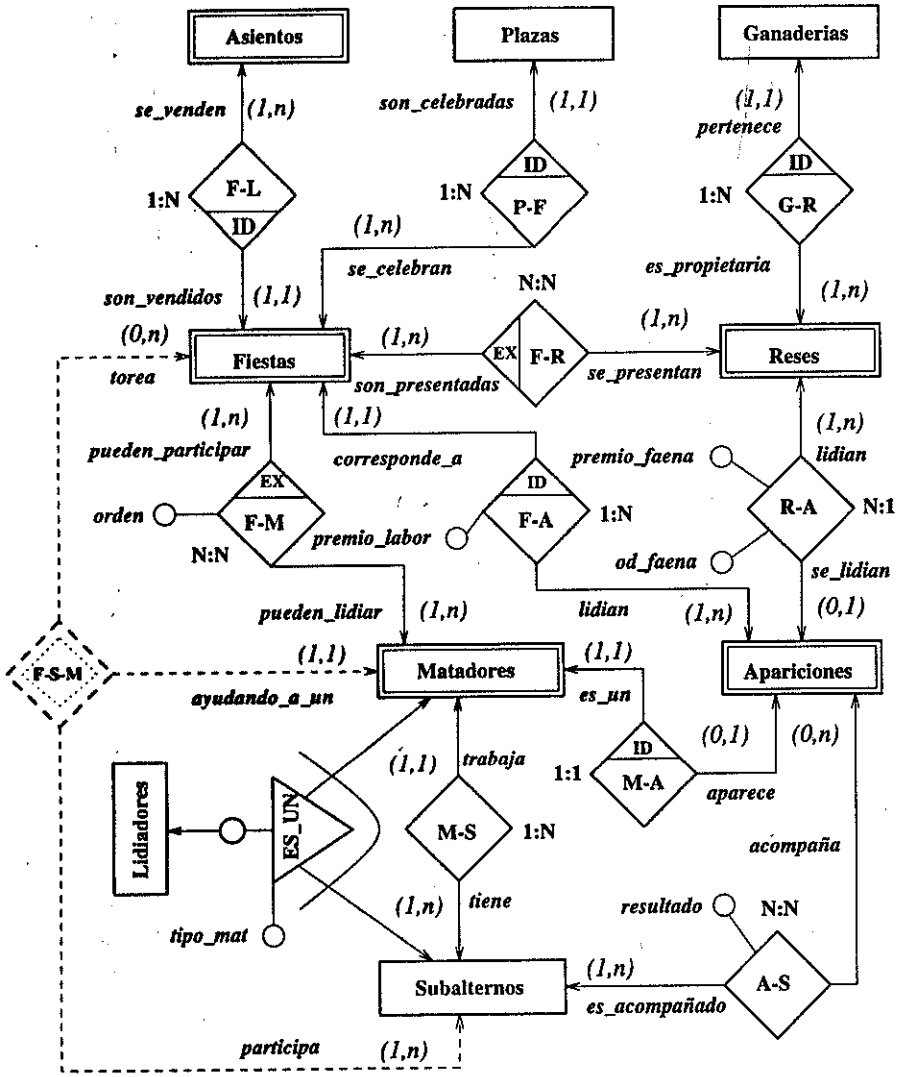


Figura 13.4 La lidia de los toros

Por otra parte, el tipo de entidad *Apariciones* es también débil por identificación con respecto al tipo de entidad *Fiestas*, estando este tipo de interrelación cualificado por el atributo *premio_labor*, como así introduce el SUPUESTO 11.

El tipo de entidad *Apariciones* está representando cada una de las intervenciones de los matadores de toros en los festejos taurinos. En la figura 13.4 se puede observar que en el tipo de interrelación *(M-A)*, el tipo de entidad *Apariciones* participa con cardinalidades *(0,1)*, debido a que pueden existir

matadores que hayan estado considerados para un festejo determinado, y por tanto se consideren, y que posteriormente no hayan participado en el mismo.

2. Por otro lado, el tipo de entidad *Apariciones* mantendrá un tipo de interrelación con el tipo de entidad *Reses*, representando las reses que lidia cada matador en ese festejo, y este tipo de interrelación tendrá asociado el atributo *premio_faena* representando el premio concedido a la lidia de la res.

   En este tipo de interrelación *(R-A)* se considera que una res es lidiada por un solo matador en un festejo, por lo que se está considerando que si por algún accidente del matador, la res debe ser muerta por un segundo matador, éste será considerado únicamente en el campo de información asociado al tipo de interrelación *(R-A)*.

La figura 13.4 muestra el esquema conceptual final en el cual es necesario detenernos. Recuérdese que en las secciones precedentes se presentaron las relaciones que existían entre los tipos de entidad *Fiestas* y *Reses* y los subtipos de esta última. Pero, a la vista de las relaciones propuestas entre los tipos de entidad *Reses* y *Apariciones* esos tipos son, ahora, redundantes por lo siguiente (ver figura 13.1):

- El tipo de interrelación *(F-RL)* está ahora representado por el tipo de interrelación *(R-A)*, en el que se representan las reses que son lidiadas y el matador que las lidió.
- El tipo de interrelación *(R-RS)* es espurio, debido a que las reses que no son lidiadas son conocidas en base a la existencia de dos tipos de interrelación: *(R-A)* y *(F-R)*, este último representando las reses que participan en una fiesta (las lidiadas y las sobrantes) y *(R-A)* representando las lidiadas.

Por ello, sólo el tipo de interrelación *(F-R)* (en líneas discontinuas) debe ser considerado en la figura 13.1, como así se muestra en la figura 13.4.

Como es necesario representar los subalternos de cada matador que le ayudan en su labor en cada fiesta, el tipo de entidad *Matadores* debe modificarse con respecto al presentado en la figura 13.3.

Como muestra la figura 13.4, es necesario generalizar aún más y generar un nuevo supertipo de entidad denominado *Lidiadores*. El tipo de entidad *Lidiadores* se puede especializar en los subtipos *Matadores* y *Subalternos* de forma total y exclusiva (como se muestra en la figura 13.4) y éstos a su vez especializarse en otros subtipos como se mostró en la figura 13.3. De esta forma se pueden definir las siguientes relaciones:

- Un tipo de interrelación *(M-S)* representando la cuadrilla actual de cada matador de toros.
- Un tipo de interrelación *(A-S)* representando la cuadrilla que participa con cada matador en cada festejo, como así impone el SUPUESTO 10. Este tipo de interrelación viene caracterizado por el atributo *resultado* que permitirá

mantener información de la labor realizada por cada subalterno en los festejos en los que participó *su matador*.

13.3. MODELO RELACIONAL

Del esquema conceptual propuesto anteriormente y representado en la figura 13.4 puede derivarse el siguiente esquema relacional:

BASE DE DATOS DE LA FIESTA NACIONAL

Plazas	*(nombre_plaza, cat_plaza, aforo, localidad_plaza, apoderado_plaza)*
Ganaderias	*(nombre_ganaderia, dueno_ganaderia, dir_ganaderia, color_ganaderia, hierro_ganaderia)*
Fiestas	**(nombre_plaza**, *fecha_fiesta, clase_fiesta, hora_fiesta, exp_fiesta)*
Reses	**(nombre_ganaderia**, *nombre_res, numero_res, peso_res, od_res)*
Localidades	**(nombre_plaza, fecha_fiesta**, *asiento#, precio)*
Carteles	**(nombre_plaza, fecha_fiesta, apodo_matador**, *orden)*
Apariciones	**(nombre_plaza, fecha_fiesta, apodo_matador**, *premio_labor)*
Presencias	**(nombre_plaza, fecha_fiesta, nombre_ganaderia, nombre_res** *tipo_res)*
Faenas	**(nombre_ganaderia, nombre_res, nombre_plaza, fecha_fiesta, apodo_matador**, *premio_faena, od_faena)*
Lidiadores	*(apodo_lidiador, nombre_lidiador, tipo_mat, nac_lidiador)*
Matadores	**(apodo_matador**, *apoderado_matador)*
Subalternos	**(apodo_subalterno**, *tipo_sub*, **apodo_matador**)
Faenillas	**(nombre_plaza, fecha_fiesta, apodo_matador, apodo_subalterno,** *resultado)*
	`*** Esta tabla será modificada posteriormente ***`

El esquema relacional presentado ha sido derivado del modelo conceptual propuesto realizando las siguientes acciones (véanse las figuras 13.1 y 4):

- Las tablas *Plazas* y *Ganaderias* por aplicación de la regla *RTECAR-1*.
- Las tablas *Fiestas* y *Reses* por aplicación de la regla *RTECAR-3.1*.
- Las tablas *Lidiadores, Matadores* y *Subalternos* por aplicación de la regla *PRTECAR-5*, la cual elimina el tipo de interrelación jerárquica. Además, la tabla *Subalternos* es obtenida por:
  - Aplicación de la regla *PRTECAR-4* al tipo de interrelación jerárquica entre el tipo de entidad *Subalternos* y sus subtipos.
  - Aplicación de la regla *RTECAR-3.1* al tipo de interrelación *(M-S)*.
  - Por otro lado, la tabla *Matadores* es derivada por aplicación de la regla *RTECAR-2.2* al tipo de interrelación *(M-A)*, y la regla *RTECAR-1* al tipo de entidad *Matadores*.
- Las tablas *Carteles, Faenillas, Presencias* y *Localidades* son derivadas por aplicación de la regla *RTECAR-4* a los tipos de interrelación *(F-M), (A-S), (F-R)* y *(F-L)*.

  Como se ha argumentado en el desarrollo de la solución al problema propuesto, no interesa representar en el modelo los diferentes tipos de asientos que tiene cada plaza, sino únicamente los precios de los diferentes tipos de asientos para cada festejo. Por ello, no se ha derivado una tabla desde el tipo de entidad *Asientos*.

  De esta forma, los asientos con sus correspondientes precios puede verse como un atributo múltiple del tipo de entidad *Fiestas*, el cual por aplicación de las reglas *PRTECAR-1* y *PRTECAR-2* se transforma en un tipo de entidad débil por identificación con respecto al tipo de entidad *Fiestas* sobre el cual no se deriva una tabla independiente.
- La tabla *Apariciones* es derivada por aplicación de la regla *RTECAR-2.2* al tipo de interrelación *(M-A)*, y por aplicación de la regla *RTECAR-3.1* al tipo de interrelación *(F-A)*.
- Por último, la tabla *Faenas* es derivada de la aplicación de la regla *RTECAR-3.2* al tipo de interrelación *(R-A)*.

13.4. NORMALIZACIÓN DEL MODELO

Todas las tablas presentes en el esquema relacional se encuentran normalizadas a excepción de la tabla *Faenillas*. Veamos la razón:

1. Es el SUPUESTO 10 el que impone la necesidad de conocer los subalternos que participan con cada matador en cada fiesta independientemente de que pertenezcan, o no, a su cuadrilla habitual.
2. Para ello se ha considerado en el modelo conceptual el tipo de interrelación *(A-S)*, una relación *N:N* caracterizada por el atributo *resultado*, que representa las participaciones de cada subalterno en cada aparición de un matador en una fiesta.
3. Al derivar este tipo de interrelación basándose en la regla *RTECAR-4*, se ha obtenido la tabla *Faenillas*, cuya clave es la agregación de los identificadores de las tablas *Apariciones* y *Subalternos*, y con *resultado* como único atributo no primo.
4. Pero esta tabla está representando erróneamente los puntos 1 y 2 descritos anteriormente, puesto que:

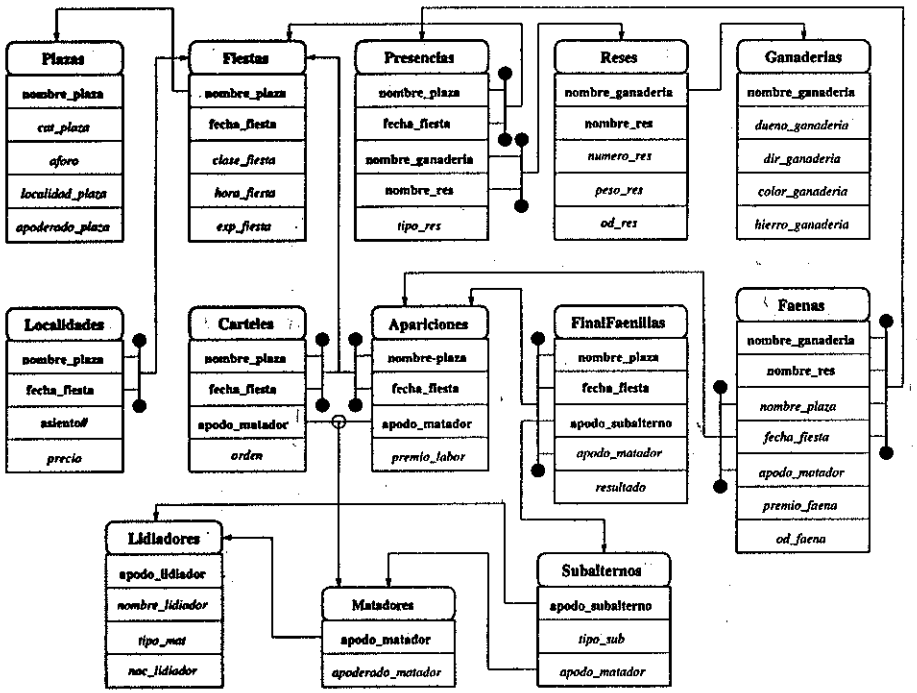


Figura 13.5 Diagrama Relacional de la Fiesta Nacional

a) Sería posible representar que un subalterno, en una misma fiesta, podría participar con más de un matador.

b) El atributo *resultado* representaría el comportamiento del subalterno para cada aparición de algún matador en la fiesta (cada lidia), mientras que este atributo es independiente del matador, y sólo lo es de la presencia del subalterno en la fiesta.

Por ello, existe una dependencia no completa entre el atributo *resultado* y la clave de la tabla *Faenillas*, puesto que este atributo no depende del atributo *apodo_matador*.

La normalización de esta tabla supone su descomposición en las siguientes tablas:

NuevaFaenillas (**nombre_plaza, fecha_fiesta, apodo_subalterno**, *resultado*)

Ayudas (**nombre_plaza, fecha_fiesta, apodo_matador, apodo_subalterno**)

Como se puede observar, en la tabla *NuevaFaenillas* se ha eliminado el problema de la dependencia incompleta del atributo *resultado* que existía en la tabla

Faenillas. Pero en la tabla *Ayudas* no se ha eliminado el problema de que puedan existir tuplas para un mismo subalterno, en una misma fiesta, para distintos matadores.

La única forma de evitar este inconveniente sería que en lugar de considerar la relación existente entre los tipos de entidad *Subalternos* y *Apariciones* (el tipo de interrelación *(A-S)*, ver figura 13.4), considerar un tipo de interrelación entre los tipos de entidad *Fiestas* y *Subalternos* cualificada por el atributo *apodo_matador* o, lo que es lo mismo, un tipo de interrelación ternaria en el que participe el tipo de entidad *Matadores* con las cardinalidades *(1,1)*, como así se ha representado en trazos discontinuos en el esquema de la figura 13.4.

La transformación de este subesquema daría lugar a la aparición de la siguiente tabla por aplicación de la regla *RTECAR-4*:

FinalFaenillas (**nombre_plaza, fecha_fiesta, apodo_subalterno, apodo_matador**, *resultado*)

que ya sí representa el problema planteado, y así se ha considerado como se observa en el diagrama relacional de la figura 13.5, aunque hay que tener en cuenta que:

1. Deberá existir una referencia del agregado *nombre_plaza, fecha_fiesta, apodo_matador* a la tabla *Apariciones*, puesto que los subalternos no van solos a las plazas.
2. Por tanto, el atributo *apodo_matador* no podrá tomar valores nulos.

13.5. CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

El modelo relacional propuesto queda de la forma representada en la figura 13.5 y se procede a continuación a la definición del esquema relacional de la base de datos

13.5.1. Definición sintáctica de las tablas

```
/* Inicialmente se borran las tablas */
DROP TABLE Faenas;
DROP TABLE FinalFaenillas;
DROP TABLE Subalternos;
DROP TABLE Presencias;
DROP TABLE Reses;
DROP TABLE Ganaderias;
DROP TABLE Apariciones;
DROP TABLE Carteles;
DROP TABLE Matadores;
DROP TABLE Lidiadores;
DROP TABLE Localidades;
DROP TABLE Fiestas;
DROP TABLE Plazas;
```

```
/* Se crean las tablas del esquema relacional propuesto */
CREATE TABLE Plazas  (
   nombre_plaza VARCHAR2(20) NOT NULL,
   cat_plaza NUMBER(1),
   aforo NUMBER(5),
   localidad_plaza VARCHAR2(15),
   apoderado_plaza VARCHAR2(30),
    CONSTRAINT pk_plazas
     PRIMARY KEY (nombre_plaza),
    CONSTRAINT ck_nop
     CHECK (nombre_plaza = INITCAP(nombre_plaza)),
    CONSTRAINT ck_apo
   CHECK (apoderado_plaza = INITCAP(apoderado_plaza)),
    CONSTRAINT ck_afo
     CHECK (aforo > 0) );
CREATE TABLE Ganaderias  (
   nombre_ganaderia VARCHAR2(20) NOT NULL,
   dueno_ganaderia VARCHAR2(25),
   dir_ganaderia VARCHAR2(40),
   color_ganaderia VARCHAR2(30),
   hierro_ganaderia VARCHAR2(3),
    CONSTRAINT pk_ganaderias
     PRIMARY KEY (nombre_ganaderia),
    CONSTRAINT ck_nga
     CHECK (nombre_ganaderia = INITCAP(nombre_ganaderia)),
    CONSTRAINT ck_due
     CHECK (dueno_ganaderia = INITCAP(dueno_ganaderia)) );
CREATE TABLE Fiestas  (
   nombre_plaza VARCHAR2(20) NOT NULL,
   fecha_fiesta DATE NOT NULL,
   clase_fiesta NUMBER(1),
   hora_fiesta DATE,
   exp_fiesta VARCHAR2(10),
    CONSTRAINT pk_fiestas
     PRIMARY KEY (nombre_plaza, fecha_fiesta),
    CONSTRAINT fk_Fie_Pla
     FOREIGN KEY (nombre_plaza)
     REFERENCES Plazas(nombre_plaza)
     ON DELETE CASCADE );
CREATE TABLE Reses  (
   nombre_ganaderia VARCHAR2(20) NOT NULL,
   nombre_res VARCHAR2(20) NOT NULL,
   numero_res NUMBER(3),
   peso_res NUMBER(3),
   od_res LONG,
   CONSTRAINT pk_reses
     PRIMARY KEY (nombre_ganaderia, nombre_res),
   CONSTRAINT fk_res_gan
```



```
     FOREIGN KEY (nombre_ganaderia)
     REFERENCES Ganaderias(nombre_ganaderia)
     ON DELETE CASCADE,
   CONSTRAINT ck_nres
     CHECK (nombre_res = INITCAP(nombre_res)),
   CONSTRAINT ck_pes
     CHECK (peso_res > 100) );
CREATE TABLE Localidades  (
   nombre_plaza VARCHAR2(20) NOT NULL,
   fecha_fiesta DATE NOT NULL,
   asiento VARCHAR2(20) NOT NULL,
   precio NUMBER(4),
    CONSTRAINT pk_loc
     PRIMARY KEY (nombre_plaza, fecha_fiesta, asiento),
    CONSTRAINT fk_loc_fie
     FOREIGN KEY (nombre_plaza, fecha_fiesta)
     REFERENCES Fiestas(nombre_plaza, fecha_fiesta)
     ON DELETE CASCADE,
    CONSTRAINT ck_locpre
     CHECK (precio > 0) );
CREATE TABLE Lidiadores  (
   apodo_lidiador VARCHAR2(25) NOT NULL,
   nombre_lidiador VARCHAR2(35) NOT NULL,
   tipo_mat VARCHAR2(10) NOT NULL,
   nac_lidiador VARCHAR2(20),
    CONSTRAINT pk_lid
     PRIMARY KEY (apodo_lidiador),
    CONSTRAINT ck_apd
     CHECK (apodo_lidiador = INITCAP(apodo_lidiador)),
    CONSTRAINT ck_nli
     CHECK (nombre_lidiador = INITCAP(nombre_lidiador)) );
CREATE TABLE Matadores  (
   apodo_matador VARCHAR2(25) NOT NULL,
   apoderado_matador VARCHAR2(30),
    CONSTRAINT pk_mat
     PRIMARY KEY (apodo_matador),
    CONSTRAINT fk_mat_lid
     FOREIGN KEY (apodo_matador)
     REFERENCES Lidiadores(apodo_lidiador)
     ON DELETE CASCADE,
    CONSTRAINT ck_ado
     CHECK (apoderado_matador = INITCAP(apoderado_matador)) );
CREATE TABLE Subalternos  (
   apodo_subalterno VARCHAR2(25) NOT NULL,
   tipo_sub VARCHAR2(10) NOT NULL,
   apodo_matador VARCHAR2(25),
    CONSTRAINT pk_sub
     PRIMARY KEY (apodo_subalterno),
```

```
    CONSTRAINT fk_sub_lid
      FOREIGN KEY (apodo_subalterno)
      REFERENCES Lidiadores(apodo_lidiador)
      ON DELETE CASCADE,
    CONSTRAINT fk_sub_mat
      FOREIGN KEY (apodo_matador)
      REFERENCES Matadores(apodo_matador) );
CREATE TABLE Carteles  (
   nombre_plaza VARCHAR2(20) NOT NULL,
   fecha_fiesta DATE NOT NULL,
   apodo_matador VARCHAR2(25) NOT NULL,
   orden NUMBER(1),
    CONSTRAINT pk_carteles
      PRIMARY KEY (nombre_plaza, fecha_fiesta, apodo_matador),
    CONSTRAINT fk_car_fie
      FOREIGN KEY (nombre_plaza, fecha_fiesta)
      REFERENCES Fiestas(nombre_plaza, fecha_fiesta)
      ON DELETE CASCADE,
    CONSTRAINT fk_car_mat
      FOREIGN KEY (apodo_matador)
      REFERENCES Matadores(apodo_matador)
      ON DELETE CASCADE );
CREATE TABLE Apariciones  (
   nombre_plaza VARCHAR2(30) NOT NULL,
   fecha_fiesta DATE NOT NULL,
   apodo_matador VARCHAR2(25) NOT NULL,
   premio_labor VARCHAR2(25),
   CONSTRAINT pk_apa
      PRIMARY KEY (nombre_plaza, fecha_fiesta, apodo_matador),
   CONSTRAINT fk_apa_fie
      FOREIGN KEY (nombre_plaza, fecha_fiesta)
      REFERENCES Fiestas(nombre_plaza, fecha_fiesta)
      ON DELETE CASCADE,
   CONSTRAINT fk_apa_mat
      FOREIGN KEY (apodo_matador)
      REFERENCES Matadores(apodo_matador)
      ON DELETE CASCADE );
CREATE TABLE Presencias  (
   nombre_plaza VARCHAR2(20) NOT NULL,
   fecha_fiesta DATE NOT NULL,
   nombre_ganaderia VARCHAR2(20) NOT NULL,
   nombre_res VARCHAR2(20) NOT NULL,
   tipo_res CHAR(1),
    CONSTRAINT pk_pre
      PRIMARY KEY (nombre_plaza, fecha_fiesta,
                   nombre_ganaderia, nombre_res),
    CONSTRAINT fk_pre_fie
      FOREIGN KEY (nombre_plaza, fecha_fiesta)
      REFERENCES Fiestas(nombre_plaza, fecha_fiesta)
```



```
      ON DELETE CASCADE,
    CONSTRAINT fk_pre_res
      FOREIGN KEY (nombre_ganaderia, nombre_res)
      REFERENCES Reses(nombre_ganaderia, nombre_res)
      ON DELETE CASCADE,
    CONSTRAINT ck_rmv
      CHECK (tipo_res IN ('L', 'R')) );
CREATE TABLE Faenas  (
   nombre_ganaderia VARCHAR2(20) NOT NULL,
   nombre_res VARCHAR2(20) NOT NULL,
   nombre_plaza VARCHAR2(30) NOT NULL,
   fecha_fiesta DATE NOT NULL,
   apodo_matador VARCHAR2(30) NOT NULL,
   premio_faena VARCHAR2(20),
   od_faena LONG,
    CONSTRAINT pk_fae
      PRIMARY KEY (nombre_ganaderia, nombre_res),
    CONSTRAINT fk_fae_pre
      FOREIGN KEY (nombre_ganaderia, nombre_res, nombre_plaza,
                   fecha_fiesta)
      REFERENCES Presencias(nombre_ganaderia, nombre_res,
                            nombre_plaza, fecha_fiesta)
      ON DELETE CASCADE,
    CONSTRAINT fk_fae_apa
      FOREIGN KEY (nombre_plaza, fecha_fiesta, apodo_matador)
      REFERENCES Apariciones(nombre_plaza, fecha_fiesta,
                             apodo_matador)
      ON DELETE CASCADE );
CREATE TABLE FinalFaenillas  (
   nombre_plaza VARCHAR2(30) NOT NULL,
   fecha_fiesta DATE NOT NULL,
   apodo_subalterno VARCHAR2(30) NOT NULL,
   apodo_matador VARCHAR2(30) NOT NULL,
   resultado VARCHAR2(25),
    CONSTRAINT pk_fla
      PRIMARY KEY (nombre_plaza, fecha_fiesta,
                   apodo_subalterno),
    CONSTRAINT fk_fna_apa
      FOREIGN KEY (nombre_plaza, fecha_fiesta, apodo_matador)
      REFERENCES Apariciones(nombre_plaza, fecha_fiesta,
                             apodo_matador)
      ON DELETE CASCADE,
    CONSTRAINT fk_fna_sub
      FOREIGN KEY (apodo_subalterno)
      REFERENCES Subalternos(apodo_subalterno) );
```

13.5.2. Manipulación de la Base de Datos

Cap13ej01.sql

Obtener todas las plazas de segunda clase en las que se han celebrado festejos durante el mes de septiembre de 1996.

De la tabla Plazas se obtiene el atributo nombre_plaza de todas aquellas plazas que cumplen la condición de que su atributo cat_plaza tiene el valor 2, es decir, es de segunda clase. El resultado de esta subconsulta se utiliza para obtener de la tabla Fiestas el atributo nombre_plaza de todas aquellas plazas que, entre las devueltas por la consulta anterior, cumplen la condición de que en ellas se celebró un festejo entre las fechas comprendidas entre el 1 de septiembre de 1996 (inclusive) y el 1 de octubre de 1996. Para ello se utiliza el operador BETWEEN, que comprueba que el valor del atributo fecha_fiesta está comprendido entre ambas fechas.

```
SELECT DISTINCT nombre_plaza "Plazas"
   FROM Fiestas
   WHERE nombre_plaza IN
      (SELECT nombre_plaza
       FROM Plazas
       WHERE cat_plaza = 2)  AND fecha_fiesta BETWEEN
             TO_DATE('01/09/1996','DD/MM/YYYY')
             AND TO_DATE('30/09/1996','DD/MM/YYYY')
   ORDER BY nombre_plaza;
```

Resultado

```
Plazas
--------------------
Jaén
```

Cap13ej02.sql

Obtener los matadores que están apoderados por Julio Pérez Muñoz y los miembros de sus cuadrillas.

De la tabla Matadores se obtiene el campo apodo_matador de todos los matadores que cumplen la condición de que su atributo apoderado_matador tiene el valor 'Julio Pérez Muñoz', y se obtienen de la tabla Subalternos los campos apodo_subalterno y tipo_sub. El resultado de ambas consultas se reúne bajo la condición de que el atributo apodo_matador de ambas tablas tenga el mismo valor, y el resultado de la consulta se muestra ordenado por dicho campo.

```
/* La cláusula BREAK ON columna, permite que no se repitan las
salidas de una columna con el mismo valor */
BREAK ON Matador
SELECT Matadores.apodo_matador "Matador", apodo_subalterno
       "Subalterno", tipo_sub "Tipo"
   FROM Matadores, Subalternos
   WHERE apoderado_matador = 'Julio Pérez Muñoz' AND
         Matadores.apodo_matador = Subalternos.apodo_matador
   ORDER BY Matadores.apodo_matador;
```

Resultado

```
Matador                   Subalterno                Tipo
------------------------- ------------------------- ----------
Cristina Sánchez          Alberto Elvira            B
                          Alejandro Silveti         Q
El Javi                   Antonio Mondéjar          P
                          Manolo Mejía              B
                          Manuel Caballero          Q
                          Manuel De Paz             P
```


Cap13ej03.sql

Obtener el peso medio de las reses que han sido toreadas en la plaza 'Los Califas'.

De la tabla Reses se obtiene, mediante el operador numérico AVG, la media aritmética de los valores del atributo peso_res, de aquellas tuplas que satisfacen la condición de que los atributos nombre_ganaderia y nombre_res son el resultado de proyectar, sobre estos mismos atributos, las tuplas de la tabla Faenas para las cuales el valor del atributo nombre_plaza = 'Los Califas'. (Obsérvese que es la tabla Faenas la que mantiene la información de las reses que son lidiadas.).

```
SELECT AVG(peso_res) "Peso Medio de las Reses"
   FROM Reses
   WHERE (nombre_ganaderia, nombre_res) IN
      (SELECT nombre_ganaderia, nombre_res
         FROM Faenas
         WHERE nombre_plaza = 'Los Califas');
```

Resultado

```
Peso Medio de las Reses
-----------------------
              636,208333
```

Cap13ej04.sql

Obtener todas las ocasiones en las que algún subalterno ha participado en un festejo sin el matador a cuya cuadrilla pertenece.

De la tabla FinalFaenillas se obtienen los atributos apodo_subalterno, nombre_plaza y fecha_fiesta de todas las tuplas que cumplen la condición de que el atributo apodo_matador tiene el valor resultante de la NEGACIÓN de la subconsulta sobre la tabla Subalternos en la cual se obtiene el atributo apodo_matador de todos los subalternos cuyo atributo apodo_subalterno es igual al evaluado en la consulta exterior. De este modo, se obtiene el apodo de todos los matadores que no tienen en su cuadrilla al subalterno que se está evaluando en cada momento y, por tanto, todas las faenillas que ha realizado este subalterno con ellos.

```
SELECT apodo_subalterno "Subalterno", nombre_plaza
       "Plaza", TO_CHAR(fecha_fiesta,'DD/MM/YYYY') "Fecha"
   FROM FinalFaenillas
   WHERE apodo_matador NOT IN
      (SELECT apodo_matador
         FROM Subalternos
         WHERE apodo_subalterno =
               FinalFaenillas.apodo_subalterno )
   ORDER BY apodo_subalterno,3;
```

Resultado

```
Subalterno            Plaza                        Fecha
--------------------- ---------------------------- ----------
Cantero               Real Maestranza              09/06/1993
Cantero               Real Maestranza              09/06/1996
```

Cap13ej05.sql

Obtener el precio de la localidad más cara que se ha vendido en alguna plaza de segunda categoría.

De la tabla Plazas se obtiene el atributo nombre_plaza de todas las plazas cuyo atributo cat_plaza tiene el valor 2. El resultado de esta subconsulta es utilizado para obtener de la tabla Localidades el atributo precio de todas las localidades vendidas en dichas plazas. Estos precios serán comparados con el resultado de una subconsulta llamada Aux contenida en la cláusula FROM de la consulta más externa, y mediante la cual se calcula el precio máximo de las localidades de las plazas de segunda categoría. Este precio máximo será calculado gracias al operador MAX.

```
SELECT precio "Importe", nombre_plaza "Plaza",
       TO_CHAR(fecha_fiesta,'DD/MM/YYYY') "Fiesta"
   FROM Localidades , (SELECT MAX(precio) precio_max
                       FROM Localidades
                       WHERE nombre_plaza IN
                         (SELECT nombre_plaza
                            FROM Plazas
                            WHERE cat_plaza = '2')) Aux
   WHERE precio=Aux.precio_max AND nombre_plaza IN
     (SELECT nombre_plaza
      FROM Plazas
      WHERE cat_plaza = '2');
```

Resultado

```
   Importe Plaza                Fiesta
---------- -------------------- ----------
      6500 Andújar              10/09/1991
      6500 Andújar              10/09/1992
```

Cap13ej06.sql

Se supone que la base de datos almacena un histórico de muchas temporadas. Obtener todos los lidiadores existentes de la saga 'Limero'.

El operador de comparación LIKE devuelve el valor verdadero si la cadena que le precede se ajusta al patrón que sigue a dicho operador "cadena LIKE patrón". Por otra parte, el carácter '%' sirve como patrón para cualquier cadena de 0 o más caracteres excepto NULL. De este modo, para que la operación de consulta muestre la información de todas las tuplas de la tabla Lidiadores cuyo atributo apodo_lidiador sea de la forma 'Limero 1' o 'Limero Chico', se debe ejecutar la siguiente sentencia: apodo_lidiador LIKE '%Limero%'.

```
/* Se formatea la salida */
COLUMN apodo_lidiador    FORMAT A20
COLUMN nac_lidiador      FORMAT A15
COLUMN nombre_lidiador   FORMAT A25
SELECT *
   FROM Lidiadores
   WHERE apodo_lidiador LIKE '%Limero%';
```


Resultado

```
APODO_LIDIADOR  NOMBRE_LIDIADOR          TIPO_MAT  NAC_LIDIADOR
--------------- ------------------------ --------- -------------
Limero 1        Antonio Borrero Borrero  Matador   Española
Limero Chico    Francisco Romero López   Matador   Española
```

Cap13ej07.sql

Obtener la plaza en la que más toros ha lidiado el matador Finito De Córdoba.

De la tabla Faenas se obtiene el número de toros que ha lidiado 'Finito De Córdoba' y la plaza correspondiente, para aquella plaza en la que más toros ha lidiado, y este valor es comprobado en base a la selección del número máximo de toros que ha lidiado este matador en la misma plaza, mediante la búsqueda controlada en la tabla Faenas.

```
SELECT COUNT(nombre_plaza) "Numero Toros", nombre_plaza
   FROM Faenas
   WHERE apodo_matador = 'Finito De Córdoba'
   GROUP BY nombre_plaza
   HAVING COUNT(nombre_plaza) =
     (SELECT MAX(COUNT(nombre_plaza))
        FROM Faenas
        WHERE apodo_matador = 'Finito De Córdoba'
        GROUP BY nombre_plaza);
```

Resultado

```
Numero Toros NOMBRE_PLAZA
------------ ------------------------------
          12 La Malagueta
```

Cap13ej08.sql

Crear una vista en la base de datos de modo que se pueda obtener rápidamente sólo el nombre, categoría y aforo de una plaza.

Como se ha indicado en ejemplos previos, una vista (VIEW) es una tabla lógica que se crea en la base de datos y que permite acceder a información de otras tablas (tablas base) y vistas. En esta ocasión, se creará una vista denominada CategoriaPlazas sobre la tabla Plazas para acceder únicamente a la información contenida en sus campos nombre_plaza, cat_plaza, aforo.

```
DROP VIEW CategoriaPlazas;
CREATE VIEW CategoriaPlazas  AS
   SELECT nombre_plaza, cat_plaza, aforo
    FROM Plazas;
```

Resultado

```
Vista borrada.
Vista creada.
```

Cap13ej09.sql

Realizar una consulta sobre la vista CategoriaPlazas de todas las plazas de primera categoría con un aforo mayor de 15.000.

Como se indicó anteriormente, una vista es una tabla lógica, de modo que realizar una consulta sobre ella es exactamente igual que realizar una consulta sobre cualquier tabla de la base de datos.

```
SELECT *
   FROM CategoriaPlazas
```

```
   WHERE cat_plaza = 1 AND aforo > 15000;
```

Resultado

```
NOMBRE_PLAZA          CAT_PLAZA      AFORO
--------------------- ---------- ----------
Los Califas                    1      16900
La Monumental                  1      23000
Valencia                       1      16851
```

Cap13ej10.sql

El matador 'Emilio Muñoz' se lesiona. Las corridas que tenía contratadas entre el 01/04/1990 y el 31/05/1990 las lidiará 'Finito De Córdoba'. Actualizar esta información en la base de datos.

Se actualizará el campo apodo_matador de todas las tuplas de la tabla Carteles que cumplen la condición de que el valor del atributo apodo_matador sea 'Emilio Muñoz', y el campo fecha_fiesta tenga sus valores comprendidos entre las fechas 01/04/1990 y 31/05/1990.

```
UPDATE carteles
   SET  apodo_matador = 'Finito De Córdoba'
    WHERE apodo_matador = 'Emilio Muñoz' AND
          fecha_fiesta BETWEEN TO_DATE
          ('01/04/1990','DD/MM/YYYY')
          AND TO_DATE ('31/05/1990','DD/MM/YYYY');
```

Resultado

```
1 fila modificada.
```

Cap13ejP1.sql

Se ha cambiado el nombre de la plaza de toros de Jaén en lugar de 'Jaén' ha pasado a nombrarse 'Real De Jaén'. Crear un procedimiento PL/SQL que realice la actualización de todos los datos necesarios.

El atributo nombre_plaza de la tabla Plazas, es referenciado en una gran cantidad de tablas como clave externa, en algunos casos formando parte de la clave primaria de la tabla en cuestión. Por tanto, la actualización del mismo implica modificar datos en una gran cantidad de tablas. Esto es debido a que la utilización de las claves externas permite construir una jerarquía entre las tablas. Lo cual, implica que la actualización debe realizarse siguiendo el siguiente proceso:

1. *Se identifica la tabla que está al principio de la jerarquía, en este caso es la tabla Plazas. Se salva en una variable la fila de la tabla Plazas correspondiente a la plaza 'Jaén'.*
2. *Se inserta en la tabla Plazas una nueva tupla con la información almacenada en la variable, pero con el nuevo valor para el campo que identifica a la plaza.*
3. *Se realiza la misma operación para el resto de las tablas de la jerarquía que son referenciadas por otras, en este caso Fiestas, Apariciones, Presencias.*
4. *Para las tablas que forman el último nivel de la jerarquía, es decir, los valores del campo nombre_plaza de éstas, ya no son clave foránea de ninguna otra tabla se utiliza el mandato UPDATE. En este caso para las tablas Localidades, Carteles, FinalFaenillas y Faenas.*
5. *Se borran las filas que no han sido actualizadas con UPDATE.*

```
CREATE OR REPLACE PROCEDURE actualizar_nombre_plaza
```



```
AS
   /* Se definen las variables internas de trabajo y el cursor
   para recorrer la tabla */
   fila_plaza plazas%ROWTYPE;
   /* Se declaran los cursores que recorren las tablas Fiestas,
   Apariciones y Presencias para la plaza a modificar */
   CURSOR cursor_fiestas IS
    SELECT *
      FROM Fiestas
      WHERE UPPER(nombre_plaza)=UPPER('Jaén');
   CURSOR cursor_apariciones IS
    SELECT *
      FROM Apariciones
      WHERE UPPER(nombre_plaza)=UPPER('Jaén');
   CURSOR cursor_presencias IS
    SELECT *
      FROM Presencias
      WHERE UPPER(nombre_plaza)=UPPER('Jaén');
BEGIN
   /* Se almacena en una variable de tipo registro información
   actual de la plaza */
   SELECT * INTO fila_plaza
    FROM Plazas
    WHERE nombre_plaza='Jaén';
   /* Se inserta la información almacenada en la variable de
   tipo registro en la tabla Plazas */
   INSERT INTO Plazas (nombre_plaza ,cat_plaza, aforo,
               localidad_plaza, apoderado_plaza)
    VALUES('Real De Jaén',fila_plaza.cat_plaza,
           fila_plaza.aforo, fila_plaza.localidad_plaza,
           fila_plaza.apoderado_plaza);
   /* Se insertan las nuevas tuplas con el nuevo nombre de la
   plaza para todas las ocurrencias existentes en las tablas
   relacionadas directamente con la tabla Plazas */
   FOR fila_fiestas  IN cursor_fiestas LOOP
    INSERT INTO Fiestas (nombre_plaza, fecha_fiesta,
                clase_fiesta, hora_fiesta, exp_fiesta)
      VALUES ('Real De Jaén',fila_fiestas.fecha_fiesta,
              fila_fiestas.clase_fiesta,
              fila_fiestas.hora_fiesta,
              fila_fiestas.exp_fiesta);
   END LOOP;
   FOR fila_apariciones  IN cursor_apariciones LOOP
    INSERT INTO Apariciones (nombre_plaza, fecha_fiesta,
                apodo_matador, premio_labor)
      VALUES ('Real De Jaén', fila_apariciones.fecha_fiesta,
              fila_apariciones.apodo_matador,
              fila_apariciones.premio_labor);
   END LOOP;
   FOR fila_presencias  IN cursor_presencias LOOP
```

```
    INSERT INTO Presencias (nombre_plaza, fecha_fiesta,
                nombre_ganaderia, nombre_res,Tipo_res)
      VALUES ('Real De Jaén', fila_presencias.fecha_fiesta,
              fila_presencias.nombre_ganaderia,
              fila_presencias.nombre_res,
              fila_presencias.Tipo_res);
  END LOOP;
  /* Se actualizan las tablas que componen el último nivel de
  la jerarquía */
  UPDATE Localidades
    SET nombre_plaza='Real De Jaén'
      WHERE nombre_plaza='Jaén';
  UPDATE Carteles
    SET nombre_plaza='Real De Jaén'
      WHERE nombre_plaza='Jaén';
  UPDATE FinalFaenillas
    SET nombre_plaza='Real De Jaén'
      WHERE nombre_plaza='Jaén';
    UPDATE Faenas
  SET nombre_plaza='Real De Jaén'
      WHERE nombre_plaza='Jaén';
  /* Se borran las tuplas con información obsoleta de la tabla
  Plaza y de todas las otras tablas gracias a la cláusula ON
  DELETE CASCADE definida en la clave foránea de las mismas */
  DELETE
    FROM Plazas
      WHERE nombre_plaza='Jaén';
COMMIT;
  /* Se controlan los posibles errores de operación */
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
ROLLBACK;
END;
/

Prueba de ejecución
-- Se ejecuta el procedimiento
SET SERVEROUTPUT ON
    EXEC actualizar_nombre_plaza ();
SET SERVEROUTPUT OFF
```

CAPÍTULO 14

COMIDAS A DOMICILIO

14.1. ENUNCIADO DEL PROBLEMA

Eat'n Go, una empresa internacional dedicada a la comercialización de comidas rápidas, tiene la intención de ubicar una franquicia en nuestra localidad, siéndole necesario informatizar la gestión del negocio adaptándolo a las características propias de la clientela esperada. Esta empresa se dedica a la venta de pizzas y bocadillos, además de productos complementarios como refrescos, helados, etc.

Según las especificaciones aportadas por esta empresa se sabe que:

SUPUESTO 1: Tanto las pizzas como los bocadillos pueden condimentarse con un número de ingredientes de entre un conjunto de ellos con los que trabaja la empresa.

SUPUESTO 2: Los ingredientes con los que se hacen los bocadillos pueden ser iguales o distintos a aquellos con los que se hacen las pizzas.

SUPUESTO 3: El número de ingredientes que intervienen en un artículo que se vende (pizza o bocadillo) no está delimitado, pudiendo realizarse una venta de estos artículos sin ningún ingrediente. Es decir, la empresa vende también las bases de las pizzas y el pan con los que prepara las pizzas y los bocadillos, respectivamente.

SUPUESTO 4: Cada artículo que vende la empresa (pizzas, bocadillos y productos complementarios) tiene un precio base asignado (el precio de estos productos cuando se venden de forma independiente), sin contar los ingredientes que pueden acompañar a alguno de estos tipos de artículos.

SUPUESTO 5: Cada ingrediente tiene un precio para los bocadillos, mientras que para las pizzas todos los ingredientes tienen el mismo precio (los productos complementarios no llevan ingredientes).

SUPUESTO 6: Los artículos se pueden vender en distinto tamaño, en cuyo caso el precio base es distinto según el tamaño, y el precio de los ingredientes también. Existen, actualmente, tres tamaños en los que se venden los bocadillos y las pizzas (pequeño, normal y grande).

SUPUESTO 7: Las ventas se pueden hacer de tres formas diferentes: para consumir en el local, para recoger en el local y llevar o consumir en el mismo, y para servir a domicilio, en cuyo caso puede incrementarse un cargo añadido por el porte de la venta.

SUPUESTO 8: Los artículos complementarios que vende la empresa tienen un precio fijo en base a su tipo, tamaño, sabor, etc.

SUPUESTO 9: Los clientes pueden solicitar un servicio de la empresa (un pedido) tanto personalmente en el local como telefónicamente.

SUPUESTO 10: En los pedidos telefónicos se tomarán los datos completos del cliente: en los de *recoger* sólo el documento nacional de identidad y su nombre completo, y en los de *consumir en el local* no se tomará ningún dato, a no ser que sea necesario por otras razones.

SUPUESTO 11: A la empresa le interesa mantener información de todos los clientes a los que se les recoge información por dos razones:

1. Simplemente a escala informativa, de consumo y marketing.
2. Para llevar el control del consumo y favorecer, mediante obsequios, a aquellos clientes que alcancen un cierto consumo.

SUPUESTO 12: La empresa organiza, a veces, promociones para sus clientes. Éstas se basan en el obsequio de algún regalo o artículo de propaganda sobre la base de:

1. Las unidades consumidas de alguno de los tipos de artículos: pizzas o bocadillos.
2. Cargo total alcanzado en los pedidos.

SUPUESTO 12+1: Cuando la empresa obsequia a los clientes con algún artículo de promoción siempre recaba de los mismos toda su información y, además, mantiene información sobre los regalos de promoción que se les entrega para, a ser posible, no hacer entregas duplicadas de los mismos.

SUPUESTO 14: La empresa cuenta con una serie de repartidores encargados del reparto a domicilio y de una batería de scooters para ayudarles en su cometido. Cada reparto se le asigna a un repartidor, el cual tiene asignado un scooter, aunque un mismo scooter puede ser utilizado por distintos repartidores (por supuesto en distintos turnos de trabajo).

SUPUESTO 15: A la empresa le interesa conocer información del coste de los scooters sobre la base del consumo de gasolina de los mismos en la realización de los repartos.

SUPUESTO 16: Los precios de todos los artículos que vende la empresa tienen el IVA incluido.

SUPUESTO 17: En los pedidos a domicilio existe un mínimo, en lo referente al valor del pedido, para que éste sea servido. Si el valor de los artículos que componen el pedido no alcanza este mínimo y el cliente desea que se le sirva a domicilio, se le cobrará ese mínimo.

SUPUESTO 18: La empresa cuenta con una serie de pizzas y bocadillos ***"estrellas"***, los cuales están formados por un conjunto de ingredientes predeterminados. Estos artículos tienen un nombre comercial único y su precio es el que resulta del acumulado de los ingredientes que incorporan.

14.2. MODELO CONCEPTUAL

14.2.1. Los artículos y sus ingredientes

Eat'n Go está especializada en la venta de tres tipos de artículos: las pizzas, los bocadillos y los artículos complementarios. Entonces, se puede, definir un supertipo de entidad que represente a todos los tipos de artículos y especializarlo en los subtipos correspondientes:

Tipo de entidad *Articulos*: representando a *"todos los tipos de artículos que son comercializados por la empresa"*. Este tipo de entidad viene caracterizado por los atributos *articulo#*, como identificador del artículo, y *precio_articulo*, representando al precio *base* de ese artículo, lo que permite validar el SUPUESTO 4.

Este tipo de entidad se especializa en los siguientes subtipos: ***Pizzas***, ***Bocadillos*** y *Complementarios*, como así se ha representado en la figura 14.1.

A su vez, el SUPUESTO 6 introduce el que cada artículo de cada tipo existe en una serie de tamaños, lo que permitiría especializar los subtipos *Pizzas* y *Bocadillos* en los tamaños correspondientes. Cada subtipo incorpora los atributos:

Pizzas: *precio_ingrediente,* representando el precio de cada ingrediente que puede entrar a formar parte de las pizzas, puesto que según el SUPUESTO 5 todos los ingredientes tienen el mismo precio.

Bocadillos, Complementarios: no incorporan ningún otro atributo, puesto que el precio de los bocadillos es diferente según el ingrediente y los artículos complementarios no llevan ingredientes.

Tipo de entidad *Ingredientes:* según los SUPUESTOS 1-3, tanto las pizzas como los bocadillos se preparan con un conjunto de ingredientes, además de los ingredientes base. Es necesario, entonces, definir un tipo de entidad que represente a los diferentes ingredientes alimenticios que pueden entrar a formar parte de estos artículos (ver figura 14.1).

El tipo de entidad *Ingredientes* vendrá representado por los atributos: *ingrediente#*, como identificador, y formado por un conjunto de caracteres que representa, abreviada e inequívocamente, a un ingrediente, y *nombre_ingrediente*, representando el nombre comercial de dicho ingrediente.

Estos tipos de entidad se encuentran relacionados por los siguientes tipos de interrelación como se muestra en la misma figura 14.1:

Tipo de Interrelación *Pizzas/Ingredientes* **(P-I)**: representando los diferentes ingredientes que pueden formar parte de un artículo de este tipo. Se trata de un tipo de interrelación *N:N*, puesto que una pizza puede llevar de cero a varios ingredientes (puede venderse la base de la pizza solamente), y un ingrediente puede formar parte de las pizzas, o no. Estos dos tipos de entidad mantienen otro tipo de interrelación:

Tipo de interrelación *Ingredientes/Pizzas* **(I-P)**: representando el conjunto de ingredientes que forman parte de la base de las pizzas, como así introduce el SUPUESTO 4. De esta forma, se puede considerar como ingrediente, por ejemplo, el tomate, mozzarela, orégano e incluso la masa que forma la pizza, pudiéndose considerar un conjunto de ingredientes base por cada pizza que pueden o no, además, acompañar a las mismas como ingredientes añadidos.

Se trata de un tipo de interrelación *N:N*, en el que el tipo de entidad *Ingredientes* participa con cardinalidades *(1,n)* puesto que debe existir al menos algún ingrediente como base de la pizza.

Estos tipos de interrelación están representando que las pizzas pueden ser vendidas sin que el cliente tenga que adquirir por ello algún ingrediente añadido a aquellos que se consideran como base. Así, la empresa considera que estos artículos están acompañados de una serie de ingredientes con independencia de los que los clientes, a su gusto, puedan añadirle.

Hay que tener en cuenta, como se observa en la figura 14.1, que en estos tipos de interrelación participa el tipo de entidad *Pizzas* y no los subtipos, representando que

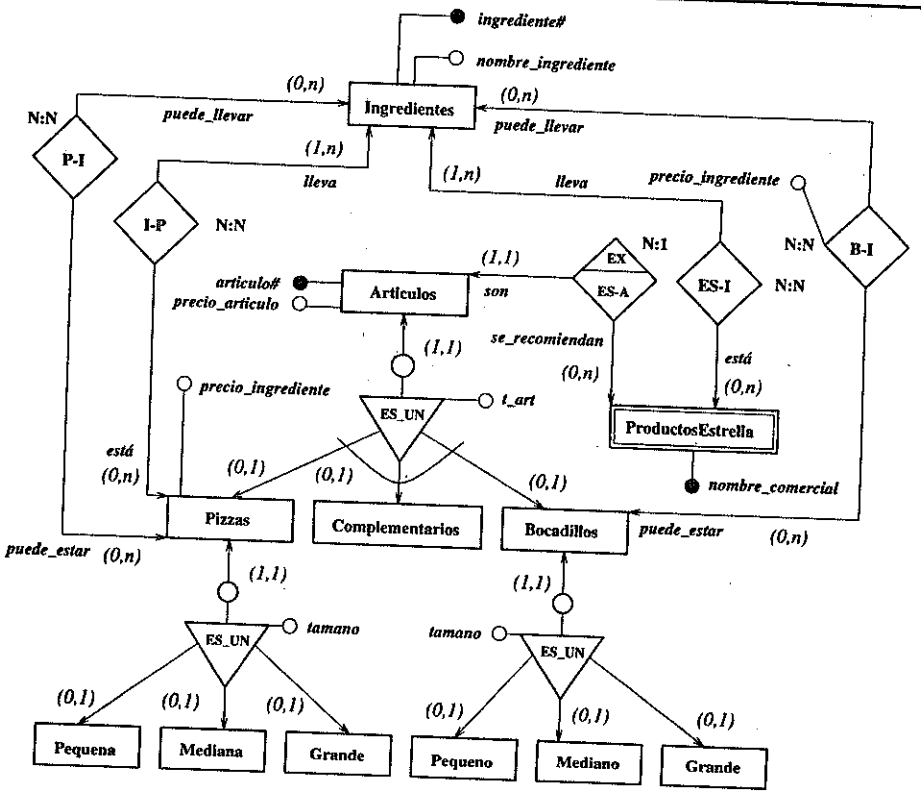


Figura 14.1 Los artículos y sus ingredientes

para todos los subtipos, el conjunto de ingredientes que pueden formar parte de la base y de las posibles pizzas es el mismo.

Las mismas consideraciones que se han tomado para las relaciones existentes entre las pizzas y los ingredientes pueden tomarse para los bocadillos y los ingredientes, a excepción de que no existe una relación entre ingredientes y bocadillos para representar los ingredientes base de éstos (los bocadillos no tienen, a diferencia de las pizzas, ingredientes base). Así, se tendrá el siguiente tipo de interrelación:

Tipo de interrelación *Bocadillos/Ingredientes* **(B-I)**: representando a los diferentes ingredientes que se pueden utilizar para la manufactura de bocadillos. Se trata de un tipo de interrelación *N:N* que viene caracterizado por el atributo *precio_ingrediente* debido a que según el SUPUESTO 5 el precio de los ingredientes es diferente, según el ingrediente de que se trate, en la manufactura de un tipo de bocadillo.

Según el SUPUESTO 18, la empresa tiene una serie de pizzas y bocadillos tipo (los productos estrella), las cuales están constituidos por un conjunto de ingredientes

predeterminado. Así, se deben considerar los siguientes objetos en el modelo (ver figura 14.1):

Tipo de entidad *ProductosEstrella:* representando a estos productos estrella de la empresa. Este tipo de entidad es débil por existencia con respecto al tipo de entidad *Articulo* con el cual participa en un tipo de interrelación *N:1*. Así, un producto estrella corresponde, en realidad, a una receta propuesta por la empresa para un determinado tipo de artículo.

Esta receta sería el conjunto de ingredientes que componen al producto estrella. Por lo tanto, es necesario representar también un tipo de interrelación entre el tipo de entidad *ProductosEstrella* y el tipo de entidad *Ingredientes* (un tipo de interrelación *N:N* como se puede observar en la figura 14.1).

Debido al SUPUESTO 18, se ha considerado el atributo *nombre_comercial* como único atributo e identificador válido para este tipo de entidad.

14.2.2. Los clientes y los pedidos de artículos

La empresa realiza una serie de ventas basándose en los pedidos que realizan los clientes los cuales, según el SUPUESTO 7, se pueden realizar de tres formas diferentes y, dependiendo de éstas, se requerirá más o menos información de los clientes. Se pueden considerar los siguientes tipos de entidad:

Tipo de entidad *Clientes*: representando a *"cualquier cliente que realice un pedido, en cualquiera de sus modalidades, de artículos a la empresa"*. Este tipo de entidad vendrá caracterizado por: el atributo *cliente#*, representando el código de cliente que le asigna la empresa al mismo, *nombre_cliente, apellidos_cliente, direccion_cliente, telf_cliente*, representando la información, necesaria para la empresa, correspondiente a sus clientes.

Tipo de entidad *TiposPedidos*: representando a *"los diferentes tipos de pedidos que pueden realizar los clientes"* (SUPUESTO 7). Consideraremos, para este tipo de entidad, únicamente el atributo identificador del mismo, *clase_pedido*, que representa al tipo de pedido que puede realizar el cliente[32].

Este tipo de entidad puede especializarse en tres subtipos como muestra la figura 14.2, según el tipo de pedido que realiza el cliente. El subtipo de entidad *PedidosDomicilio* incorpora un nuevo atributo denominado *incremento* el cual representa el incremento del coste que supone el servicio del pedido al domicilio del cliente, y además otro atributo denominado *minimo* que representa la cantidad mínima del pedido para que se sirva a domicilio, como así introduce el SUPUESTO 17.

[32] El lector podrá, en estos momentos, considerar superflua la definición de este tipo de entidad, pues puede verlo como un atributo de las relaciones que existen entre los clientes y los pedidos que realizan, pero como podrá observar a continuación, es conveniente tal definición para poder diferenciar los diferentes atributos que representan y acompañan a los diferentes tipos de pedidos que pueden realizar los clientes.

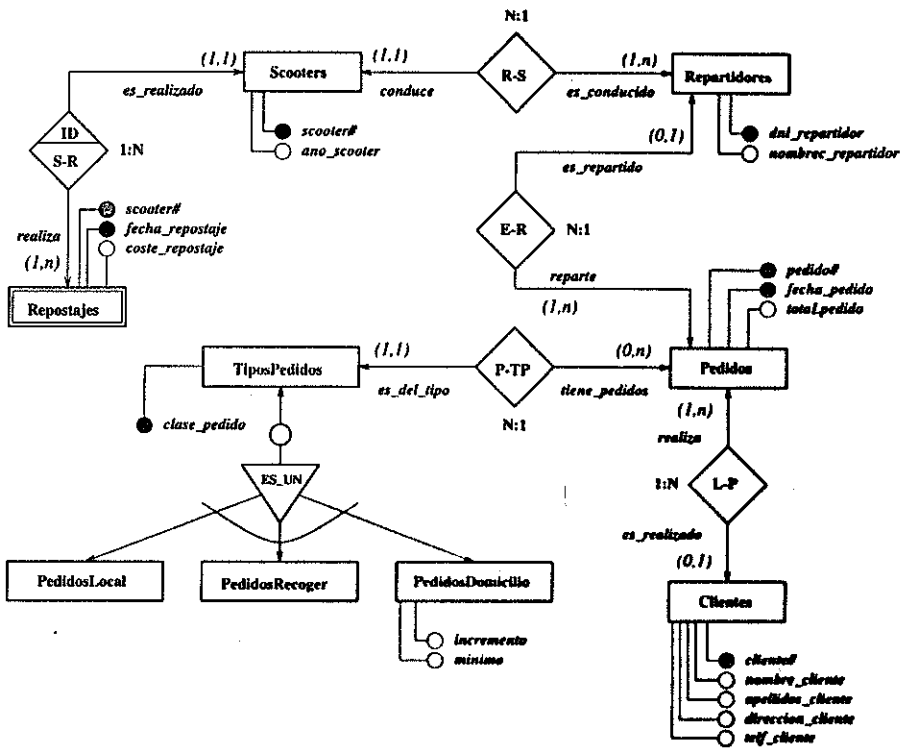


Figura 14.2 Los pedidos, sus tipos y los clientes

Tipo de entidad *Pedidos*: representando a ***"los pedidos de artículos que realizan los clientes"***. Vamos a considerar que la empresa asigna cada día un número o código identificativo del pedido de los clientes. Este código —un contador— comienza cada día con el valor de cero, por lo que para identificar cada uno de los pedidos se tendrá que considerar la agregación de los atributos ***pedido#*** y ***fecha_pedido***. Es conveniente considerar, en este tipo de entidad, el atributo ***total_pedido*** para representar el valor o coste total del mismo, teniendo en cuenta que se trata de un atributo derivado cuyo valor para cada instancia de esta entidad será calculado sobre la base del acumulado del pedido.

Este tipo de entidad participa en un tipo de interrelación ***1:N*** con el tipo de entidad *TiposPedidos*, puesto que un pedido sólo puede ser de un tipo.

Como muestra la figura 14.2, existe un tipo de interrelación entre los tipos de entidad *Clientes* y *Pedidos* en el que el tipo de entidad ***Clientes*** participa con las cardinalidades *(0,1)*, ya que no es obligado mantener información del cliente en los pedidos; por ejemplo, en los pedidos de local, mientras que el tipo de entidad ***Pedidos*** participa con las cardinalidades *(1,n)*, puesto que para ser considerado como cliente de la empresa, éste ha tenido que realizar al menos algún pedido, pudiéndose realizar muchos, naturalmente (SUPUESTOS 9 y 10).

14.2.3. Las ventas

Consideraremos como ventas al conjunto de artículos que un cliente solicita en un pedido y que, por tanto, supone un ingreso de una sola vez para la empresa.

En el modelo, las ventas pueden ser vistas como la relación existente entre los pedidos y los artículos que son solicitados en el mismo. Pero como los clientes pueden solicitar los artículos con diferentes ingredientes, en realidad, una venta tendría que ser representada como una relación ternaria entre los tipos de entidad *Articulos* (o sus subtipos), *Pedidos* e *Ingredientes* (ver figura 14.3), la cual estaría cualificada por el atributo *unidades*, para representar el número de artículos de un mismo tipo (receta) que está presente en un pedido.

La representación de los artículos que forman parte de un pedido como una relación ternaria es equívoca, y no satisface los requisitos del problema. La razón es obvia, y vamos a ponerla de manifiesto, para mayor claridad, con un ejemplo práctico:

> *Supóngase que se solicitan en un pedido dos pizzas grandes, una con atún y jamón york, y la otra con atún, extra de mozzarella, anchoas, picante, parmesano y palitos de mar*[33].

En este pedido, en la relación ternaria *(E-A-I)* se tendría:

1. El artículo que interviene en la relación sería aquel que identificara a las pizzas grandes. Por ejemplo, los valores de los identificadores serían:

   articulo# = 'Pizza' y *tamano* = 'Grande'

2. El pedido, correspondería a un único pedido y, por ejemplo, se tendría:

   fecha_pedido = '10/10/2000' y *pedido#* = 345

3. Los ingredientes serían todos aquellos que forman parte de las dos pizzas; es decir:

   ingrediente# = 'York','Mozz','Atun','Anch','Pali','Parm','Pica'

4. Y el valor del atributo *unidades* sería igual a 2.

El lector ya habrá podido observar en estos momentos el problema que plantea esta relación ternaria, y es que en lugar de considerarse que el pedido está formado por dos artículos diferentes, sólo se ha considerado uno, y dos unidades del mismo, puesto que el identificador es el mismo para ambos artículos: '*Pizza Grande*'.

Es necesario, por tanto, diferenciar cada uno de los artículos que forman parte de un pedido, y la forma adecuada de realizar esta diferenciación se muestra también en la figura 14.3 y consiste en considerar un nuevo tipo de entidad denominado ***Ventas***.

[33] ¡Recetas más raras se han visto!, y hasta posiblemente estén buenas.

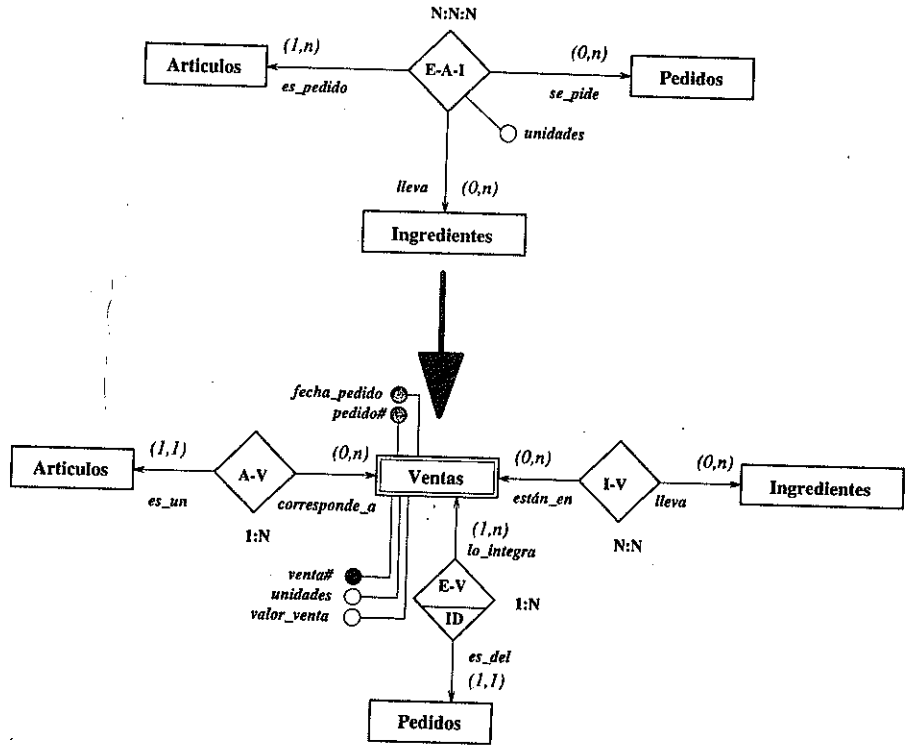


Figura 14.3 Las ventas de los artículos

El tipo de entidad *Ventas* representa cada uno de los artículos que forman parte de un pedido; es decir, las líneas del pedido que hace la empresa en cada pedido.

El tipo de entidad *Ventas* participará en los siguientes tipos de interrelación:

- Tipo de interrelación *Articulos/Ventas* (**A-V**): un tipo de interrelación *1:N*.
- Tipo de interrelación *Ingredientes/Venta* (**I-V**): un tipo de interrelación *N:N*.
- Tipo de interrelación *Pedidos/Ventas* (**E-V**): un tipo de interrelación *1:N* en el que el tipo de entidad *Ventas* es débil por identificación con respecto al tipo de entidad *Pedidos*.

El tipo de entidad *Ventas* vendrá caracterizado por los atributos: *fecha_pedido* y *pedido#* heredados del tipo de entidad *Pedidos*, y *venta#* cuya agregación sirve como identificador y, además, los atributos *unidades* y *valor_venta*, representando las unidades de un mismo artículo vendidas en un pedido (una misma receta) y el valor parcial de esa venta (de nuevo un atributo derivado que interesa considerar, simplemente para una mejora del desempeño del sistema que se construya).

14.2.4. Los repartos, repartidores y scooters

Algunos de los pedidos que realizan los clientes tienen que ser servidos a domicilio, y de este servicio se encargan una serie de repartidores que lo realizan en los scooters que tienen asignados. Pueden considerarse, por tanto, los siguientes elementos en el modelo:

Tipo de entidad *Repartidores*: representando a "*cada uno de los empleados dedicados al reparto a domicilio*". Consideraremos, únicamente, para este tipo de entidad los atributos *dni_repartidor*, como identificador, y *nombrec_repartidor*.

Tipo de entidad *Scooters*: representando a "*las diferentes motocicletas que utilizan los repartidores para hacer el reparto*". Para este tipo de entidad se considerarán los atributos *scooter#*, como identificador del mismo (por ejemplo, su matrícula), y *ano_scooters* para identificar el año de compra del mismo.

Estos dos tipos de entidad están relacionados mediante un tipo de interrelación *(R-S)*, muchos a uno, que representa el scooter asignado a cada repartidor, como así se muestra en la figura 14.2 (SUPUESTO 14).

Por otra parte, a la empresa le interesa conocer el consumo de los scooters en gasolina y los repartos que realizan, así, es necesario considerar lo siguiente (SUPUESTO 15):

Tipo de entidad *Repostajes*: representando a "*las distintas operaciones de repostaje de gasolina para cada scooter*". Se trata, por tanto, de un tipo de entidad débil por identificación con el tipo de entidad *Scooters*, teniendo como identificador el atributo identificador de ésta, agregado con el atributo *fecha_repostaje*, suponiendo que un scooter no reposta más de una vez en un mismo día. Además, incorpora el atributo *coste_repostaje*, representando el coste de esta operación.

Como un repartidor sólo tiene asignado un scooter (ver figura 14.2), el tipo de entidad *Pedidos* es necesario relacionarlo con el tipo de entidad *Repartidores*. Se trata de un tipo de interrelación *N:1*, puesto que un pedido sólo lo reparte cero o un repartidor (sólo se reparten los pedidos a domicilio).

14.2.5. Los obsequios y el consumo de los clientes

La empresa, como estrategia de marketing, acostumbra obsequiar a sus clientes con una serie de regalos en base, tanto a la cantidad alcanzada en un pedido como al consumo acumulado (SUPUESTOS 11 y 12). Por ello, es necesario considerar un nuevo atributo en el tipo de entidad *Cliente*, el atributo *consumo* que represente el consumo acumulado por el cliente.

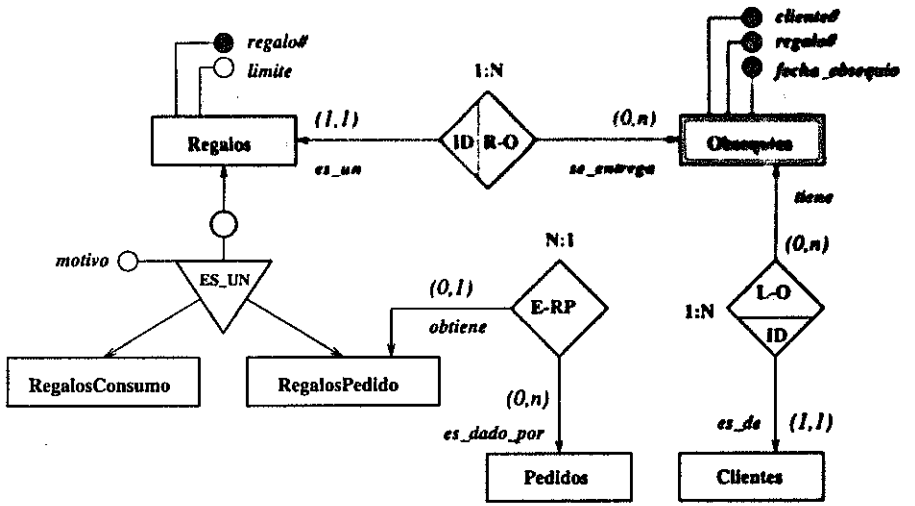


Figura 14.4 Los regalos y los obsequios a los clientes

La empresa, cuando obsequia un regalo por alcanzar un determinado consumo, actualiza el valor de este atributo. Como la empresa obsequia con distintos regalos en base al consumo de los diferentes artículos, el atributo *consumo* es un atributo múltiple que puede tener tres ocurrencias (tantas como tipos de artículos existen). Por ello, en lugar de considerar un solo atributo es necesario considerar tres: *consumo_p*, *consumo_b* y *consumo_c*, para representar el número de unidades consumidas de pizzas, bocadillos y productos complementarios respectivamente.

Si la empresa obsequia a sus clientes es necesario entonces considerar un tipo de entidad *Regalos* que represente a los diferentes regalos que puede obsequiar la misma.

El tipo de entidad *Regalos*, como muestra la figura 14.4, debe especializarse en dos subtipos de entidad: *RegalosConsumo* y *RegalosPedido* para representar a los regalos preparados para obsequiar a los clientes por esos diferentes conceptos. Se trata de un tipo de interrelación jerárquica total e inclusiva, puesto que un mismo regalo puede obsequiarse por uno o ambos conceptos.

Como la empresa no desea obsequiar con regalos repetidos a sus clientes, a no ser que ellos lo deseen, le interesa mantener información histórica de los obsequios entregados a los mismos (SUPUESTO 12+1). Por ello, es necesario considerar un tipo de entidad *Obsequios*, débil por identificación con respecto a los tipos de entidad *Clientes* y *Regalos*. Este tipo de entidad viene caracterizado, además de por los atributos heredados de los tipos de entidad *Clientes* y *Regalos*, por el atributo *fecha_regalo* para representar la fecha en que fue entregado el mismo y, como se puede entregar un mismo regalo más de una vez, este atributo también forma parte del identificador de este tipo de entidad.

Como el lector puede apreciar, en realidad se trata de un tipo de interrelación ternaria en la que participan los tipos de entidad *Fecha*, *Clientes* y *regalos*, que para simplificar, y como ya se ha descrito en capítulos previos, es conveniente abstraerlo a un tipo de entidad débil por identificación.

Por otra parte, el tipo de entidad *RegalosPedido* está relacionado con el tipo de entidad *Pedidos*, puesto que en función del valor del pedido éste puede ser merecedor de un regalo. Se trata de un tipo de interrelación *N:1*, representando que en un pedido se puede entregar cero o un regalo, y que un regalo puede ser entregado en cero o varios pedidos, como así se ha representado en la figura 14.4. Obsérvese que es conveniente relacionar al tipo de entidad *Pedidos* con el subtipo de entidad *RegalosPedido* y no con *Obsequios*, puesto que este tipo de regalos se obsequia necesariamente por un pedido, el cual es conocido, y no se necesita más información como en el caso de los otros tipos de regalos, puesto que el regalo se obsequia en la misma fecha que se realiza el pedido.

14.3. MODELO RELACIONAL

Vamos, a continuación, a proponer las tablas del esquema relacional que representan el problema planteado. Más adelante, en esta misma sección se describirán una serie de consideraciones que ha sido necesario tomar para la simplificación del problema.

BASE DE DATOS DEL PROBLEMA DE COMIDAS A DOMICILIO

ClaseArticulos	*(articulo#, t_art)*
Articulos	**(articulo#, tamano**, *precio_articulo)*
Pizzas	**(articulo#, tamano**, *precio_ingrediente)*
Ingredientes	*(ingrediente#, nombre_ingrediente)*
PizzasBase	**(articulo#, ingrediente#)**
RecetasPizzas	**(articulo#, ingrediente#)**
RecetasBocadillos	**(articulo#, tamano, ingrediente#**, *precio_ingrediente)*
ProductosEstrella	*(nombre_comercial*, **articulo#)**
RecetasEstrella	**(nombre_comercial, ingrediente#)**
Clientes	*(cliente#, nombre_cliente, apellidos_cliente, direccion_cliente, telf_cliente, consumo_p, consumo_b, consumo_c)*
Scooters	*(scooter#, ano_scooter)*
Repartidores	*(dni_repartidor#, nombrec_repartidor*, **scooter#)**
Repostajes	*(scooter#, fecha_repostaje, coste_repostaje)*
TiposPedidos	*(clase_pedido, incremento, minimo)*

Pedidos	*(fecha_pedido, pedido#*, **clase_pedido**, *total_pedido*, **cliente#, dni_repartidor#**, *valor_receta, incremento)*
Regalos	*(regalo#, motivo, limite)*
ObsequiosC	**(regalo#, motivo, cliente#**, *fecha_obsequio)*
ObsequiosP	**(fecha_pedido, pedido#, regalo#, motivo)**
VentasProductos	**(fecha_pedido, pedido#**, *venta#*, **articulo#, tamano**, *unidades, valor_venta)*
RecetasVenta	**(fecha_pedido, pedido#, venta#, ingrediente#)**

14.3.1. Análisis del modelo propuesto

Inicialmente, se ha aplicado la regla *PRTECAR-4* a los tipos de interrelación jerárquica entre los tipos de entidad *Pizzas* y *Bocadillos* con sus correspondientes subtipos para eliminar éstos (ver figura 14.1). En este caso no es necesario especializar hasta ese nivel, pues todos los subtipos se comportan de igual forma para ambos tipos de entidad. En este proceso de eliminación de los subtipos, el atributo *tamano* pasa a formar parte del identificador de los supertipos correspondientes.

Además, se ha aplicado la regla *PRTECAR-5* para eliminar el tipo de interrelación jerárquica entre el tipo de entidad *Articulos* y sus subtipos, debido a que el supertipo y los subtipos participan de forma diferente en otros tipos de interrelación y, por tanto, es necesario mantener en la representación a todos los tipos de entidad. En este proceso es necesario propagar el atributo *precio_articulo* a los subtipos debido a que este precio es diferente para cada subtipo por la presencia del atributo *tamano*, el cual ha sido propagado como se ha descrito anteriormente.

En este proceso se han obtenido:

- La tabla *ClaseArticulos*, la cual se encuentra normalizada en *FNBC*, es derivada del tipo de entidad *Articulos*.
- La tabla *Articulos*, también en *FNBC*, la cual es obtenida en un proceso de simplificación en el que todos los subtipos del tipo de entidad *Articulos* se han agrupado en una única tabla en lugar de considerar tres tablas diferentes, una para cada uno de los subtipos de entidad.

  Este proceso de agrupación de tablas que representan diferentes características de un mismo objeto es, también, una práctica muy usual, en base a la cual se simplifica el esquema relacional obtenido sin, por ello, perder información ni semántica la representación lógica obtenida.
- La tabla *Pizzas*, se obtiene del tipo de entidad del mismo nombre y mediante una especialización de la tabla *Articulos*. Obsérvese que para cualquier tipo de pizzas el precio de los ingredientes es el mismo.

Esta tabla podría haberse eliminado y considerar el atributo *precio_ingrediente* en la tabla *Articulos*, pero en este caso existiría un número de tuplas elevado en el que este atributo tomaría valores nulos (para todas aquellas tuplas correspondientes a los artículos de los tipos bocadillos y complementarios) y, además, se perdería la visión de que son sólo los artículos de la clase pizzas los que tienen esta característica.

- La tabla *Ingredientes* se deriva directamente por la aplicación de la regla *RTECAR-1* sobre el tipo de entidad del mismo nombre y, como se puede observar, se encuentra en *FNBC*.
- La tabla *PizzasBase* se deriva en base a la aplicación de la regla *RTECAR-4* al tipo de interrelación *(I-P)* y su posterior normalización de la forma siguiente:
  1. Por aplicación directa de la regla *RTECAR-4* al tipo de interrelación *(I-P)*, se obtiene la tabla:

     PBase-0 *(**articulo#, tamano, ingrediente#**)*

     en la cual se puede observar que:

     - Todos los atributos de la tabla son primos, existiendo un único determinante funcional, por lo que la tabla está en *FNBC*.
     - Existe una dependencia multivaluada entre los atributos del determinante funcional de la forma:

       PizzasBase.articulo →→ PizzasBase.(tamano, ingrediente#)

       puesto que los ingredientes que pueden llevar las pizzas en su base son independientes del tamaño de las pizzas y, viceversa.
     - Al eliminar esta dependencia se obtienen las siguientes tablas:

       PBase-1 *(**articulo#, ingrediente#**)*

       PBase-2 *(**articulo#, tamano**)*

  2. Si se analizan estas tablas se observa que la tabla *PBase-1* es igual a la tabla *PizzasBase* propuesta, y que la tabla *PBase-2* ya está considerada en el esquema, pues se trata de la tabla *Articulos*.
- La tabla *RecetasPizzas* se deriva de la aplicación de la regla *RTECAR-4* al tipo de interrelación *(P-I)* y, como se puede apreciar, se ha realizado el mismo proceso de normalización descrito par la tabla *PizzasBase* para la derivación de esta tabla.

En el esquema de la figura 14.1 se podría haber eliminado el tipo de interrelación *(I-P)*, y considerar en su lugar un atributo, por ejemplo, denominado *base*, que cualificara el tipo de interrelación *(P-I)* y representara si el ingrediente que participa en la relación es, o no, un ingrediente base.

Si se hubiera tomado esta consideración, la tabla *RecetasPizzas* hubiera quedado de la forma:

RecetasPizzas *(**articulo#, ingrediente#**, base)*

eliminándose la tabla *PizzasBase*, pero introduciendo los siguientes problemas:

- Como el número de ingredientes que son base es mucho menor que el total de ingredientes que pueden estar en las pizzas, un gran número de tuplas de esta tabla tendrán valores nulos para el atributo *base*.
- No puede existir un ingrediente base que no se pueda añadir a las pizzas. Y esto no es correcto, pues existen ingredientes que son base de las mismas pero que el cliente no puede utilizar para construirse (su receta) su pizza, por ejemplo, el tomate.

Estos problemas se eliminarían si se considerara que el atributo *base* forma parte del identificador de la tabla *RecetasPizzas*. De esta forma, con una misma tabla se podría representar la información correspondiente a los ingredientes que forman parte de la base de las pizzas y los ingredientes que pueden utilizarse para su condimentación en las múltiples posibles recetas a realizar.

Si bien esta opción simplificaría el esquema resultante al considerar sólo una tabla en lugar de dos, presenta el inconveniente de que por una lado engloba dos relaciones diferentes en las que participan los tipos de entidad *Pizzas* e *Ingredientes* (perdiéndose capacidad de representación semántica en el esquema), y por otra parte, es necesario inspeccionar tuplas no necesarias cuando se desea hacer una consulta en la que sólo es necesario conocer bien los ingredientes que forman parte de la base, o bien los ingredientes que forman parte de las recetas.

Aunque esta opción podría ser considerada, hemos elegido para el posterior desarrollo del problema tablas independientes como se puede apreciar en el esquema propuesto.

- La tabla *RecetasBocadillos* se deriva de la aplicación de la regla *RTECAR-4* al tipo de interrelación *(B-I)*. Esta tabla está normalizada en *FNBC*, puesto que el precio de los ingredientes para los bocadillos es dependiente de la clase de bocadillo (el atributo *articulo#*), el tamaño del bocadillo (el atributo *tamano*) y el ingrediente (el atributo *ingrediente#*).
- La tabla *ProductosEstrellas* se deriva de aplicar la regla *RTECAR-3.1* al tipo de interrelación *(ES-A)*, propagando el identificador del tipo de entidad *Articulo* al tipo de entidad *ProductosEstrella*, representándose que un producto estrella existe para una sola clase de artículos.
- Observe el lector que los productos estrella son nombres comerciales que se le asignan a recetas de determinadas clase de artículos, con independencia del tamaño en que se vendan. Así, surge la tabla *RecetasEstrella*, derivada por aplicación de la regla *RTECAR-4* al tipo de interrelación *ES-I*.

- Las tablas *Clientes, Scooters* y *Repartidores* se derivan de aplicar la regla *RTECAR-1* a los tipos de entidad del mismo nombre, respectivamente. Obsérvese que, en la tabla *Clientes*, se han introducido tres atributos *consumo_p, consumo_b* y *consumo_c* cuyo objetivo es representar y, por tanto, mantener el acumulado del consumo de cada cliente en cada uno de los tipos de productos. Es conveniente el considerar estos atributos para evitar la inspección de la tabla *Ventas* cuando se desee conocer la misma para, por ejemplo, conocer si a un determinado cliente le corresponde algún regalo por consumo.
- La tabla *TiposPedidos* se deriva directamente de aplicar la regla *PRTECAR-4* al tipo de interrelación jerárquica en la que participa, hecho por el cual se eliminan los subtipos, y la tabla *Pedidos* se deriva de aplicar la regla *RTECAR-3.1* a los tipos de interrelación *(P-PT)* y *(L-P)*. Se puede observar también que en esta tabla se ha incluido:
  - El atributo *incremento* correspondiente al tipo de entidad *TiposPedidos* con el cual el tipo de entidad *Pedidos* está relacionado. La razón de esta propagación es la de mantener en la tabla *Pedidos* el valor de este atributo en la fecha en que se hizo el pedido, puesto que este valor puede variar con el tiempo en la tabla *TiposPedidos*.
  - El atributo *valor_receta* representando el acumulado de todas las ventas incluidas en un pedido, por las mismas razones que se explicaron anteriormente en el caso de la tabla *Clientes*.
- Por otro lado, la tabla *Regalos* surge de aplicar la *PRTECAR-4* al tipo de interrelación jeráquica en que participa, por lo que el atributo *motivo* pasa a formar parte, agregado al identificador, de la tabla *Regalos* que se deriva de aplicar la regla *RTECAR-1* al tipo de entidad del mismo nombre.
- La tabla *Repostajes* surge de aplicar la regla *RTECAR-3.1* al tipo de interrelación *(S-R)* entre los tipos de entidad *Scooters* y *Repostajes*, y la tabla *ObsequiosC* surge de aplicar la regla *RTECAR-3.1* al tipo de interrelación *(R-O)* entre los tipos de entidad *Regalos* y *Obsequios*.
- Por último, la tabla *ObsequiosP* se deriva de aplicar la regla *RTECAR-3.2* a la relación entre el tipo de entidad *Pedidos* y *RegalosPedido*, que es al supertipo *Regalos*, por la aplicación de la regla *PRTECAR-4*.
- Finalmente, las tablas *VentasProductos* y *RecetasVenta* surgen de la aplicación de las reglas *RTECAR-3.1* a los tipos de interrelación *(A-V)* y *(E-V)*, y la regla *RTECAR-4* al tipo de interrelación *(I-V)*, respectivamente.

14.4. CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

Una vez normalizado el modelo relacional propuesto, procederemos a la definición sintáctica del esquema relacional correspondiente.

14.4.1. Definición sintáctica de las tablas

```
   /* Se borran objetos de la base de datos que son creados
   durante el desarrollo de los ejercicios */
DROP SEQUENCE indice_regalo;
   /* Se borran las tablas del esquema relacional */
DROP TABLE RecetasVenta;
DROP TABLE VentasProductos;
DROP TABLE ObsequiosP;
DROP TABLE ObsequiosC;
DROP TABLE Regalos;
DROP TABLE Pedidos;
DROP TABLE TiposPedidos;
DROP TABLE Repostajes;
DROP TABLE Clientes;
DROP TABLE RecetasEstrella;
DROP TABLE ProductosEstrella;
DROP TABLE RecetasBocadillos;
DROP TABLE RecetasPizzas;
DROP TABLE PizzasBase;
DROP TABLE Ingredientes;
DROP TABLE Pizzas;
DROP TABLE Repartidores;
DROP TABLE Scooters;
DROP TABLE Articulos;
DROP TABLE ClaseArticulos;
   /* Se crean las tablas del esquema relacional */
CREATE TABLE ClaseArticulos  (
   articulo VARCHAR2(10) NOT NULL,
   t_art CHAR(1) NOT NULL,
    CONSTRAINT pk_ClaseArticulos
      PRIMARY KEY (articulo),
    CONSTRAINT ck_art
      CHECK (articulo = INITCAP(articulo)),
    CONSTRAINT ck_tar
      CHECK (t_art IN ('P', 'B', 'C')) );
CREATE TABLE Articulos  (
   articulo VARCHAR2(10) NOT NULL,
   tamano CHAR(1) NOT NULL,
   precio_articulo NUMBER(4) NOT NULL,
    CONSTRAINT pk_Articulos
      PRIMARY KEY (articulo, tamano),
    CONSTRAINT ck_par
      CHECK (precio_articulo > = 0),
    CONSTRAINT ck_tam
      CHECK (tamano IN ('P', 'M', 'G')) );
CREATE TABLE Pizzas  (
   articulo VARCHAR2(10) NOT NULL,
   tamano CHAR(1) NOT NULL,
```

```
    precio_ingrediente NUMBER(3),
      CONSTRAINT pk_Pizzas
        PRIMARY KEY (articulo, tamano),
      CONSTRAINT fk_piz_art
        FOREIGN KEY (articulo, tamano)
        REFERENCES Articulos(articulo, tamano)
        ON DELETE CASCADE,
      CONSTRAINT ck_pin
        CHECK (precio_ingrediente > = 0) );
CREATE TABLE Ingredientes  (
    ingrediente     VARCHAR2(4) NOT NULL,
    nombre_ingrediente VARCHAR2(10) NOT NULL,
      CONSTRAINT pk_Ingredientes
        PRIMARY KEY (ingrediente),
      CONSTRAINT un_ing
        UNIQUE (nombre_ingrediente),
      CONSTRAINT ck_nin
        CHECK (nombre_ingrediente =
              INITCAP(nombre_ingrediente)) );
CREATE TABLE PizzasBase  (
    articulo VARCHAR2(10) NOT NULL,
    ingrediente     VARCHAR2(4) NOT NULL,
      CONSTRAINT pk_PizzasBase
        PRIMARY KEY (articulo, ingrediente),
      CONSTRAINT fk_pba_car
        FOREIGN KEY (articulo)
        REFERENCES ClaseArticulos(articulo)
        ON DELETE CASCADE,
      CONSTRAINT fk_pba_ing
        FOREIGN KEY (ingrediente)
        REFERENCES Ingredientes(ingrediente)
        ON DELETE CASCADE  );
CREATE TABLE RecetasPizzas  (
    articulo VARCHAR2(10) NOT NULL,
    ingrediente VARCHAR2(4) NOT NULL,
      CONSTRAINT pk_RecetasPizzas
        PRIMARY KEY (articulo, ingrediente),
      CONSTRAINT fk_rpi_car
        FOREIGN KEY (articulo)
        REFERENCES ClaseArticulos(articulo)
        ON DELETE CASCADE,
      CONSTRAINT fk_rpi_ing
        FOREIGN KEY (ingrediente)
        REFERENCES Ingredientes(ingrediente)
        ON DELETE CASCADE  );
CREATE TABLE RecetasBocadillos  (
    articulo VARCHAR2(10) NOT NULL,
    tamano CHAR(1) NOT NULL,
```



```
    ingrediente VARCHAR2(4) NOT NULL,
    precio_ingrediente NUMBER(3),
      CONSTRAINT pk_RecetasBocadillos
        PRIMARY KEY (articulo, tamano, ingrediente),
      CONSTRAINT fk_rbo_art
        FOREIGN KEY (articulo, tamano)
        REFERENCES Articulos(articulo, tamano)
        ON DELETE CASCADE,
      CONSTRAINT fk_rbo_ing
        FOREIGN KEY (ingrediente)
        REFERENCES Ingredientes(ingrediente)
        ON DELETE CASCADE,
      CONSTRAINT ck_prin
        CHECK (precio_ingrediente > = 0)  );
CREATE TABLE ProductosEstrella  (
    nombre_comercial VARCHAR2(15) NOT NULL,
    articulo VARCHAR2(10) NOT NULL,
      CONSTRAINT pk_ProductosEstrella
        PRIMARY KEY (nombre_comercial),
      CONSTRAINT fk_pes_art
        FOREIGN KEY (articulo)
        REFERENCES ClaseArticulos(articulo)
        ON DELETE CASCADE,
      CONSTRAINT ck_nco
        CHECK (nombre_comercial = INITCAP(nombre_comercial))  );
CREATE TABLE RecetasEstrella  (
    nombre_comercial VARCHAR2(15) NOT NULL,
    ingrediente VARCHAR2(4) NOT NULL,
      CONSTRAINT pk_RecetasEstrella
        PRIMARY KEY (nombre_comercial, ingrediente),
      CONSTRAINT fk_res_pes
        FOREIGN KEY (nombre_comercial)
        REFERENCES ProductosEstrella(nombre_comercial),
      CONSTRAINT fk_res_ing
        FOREIGN KEY (ingrediente)
        REFERENCES Ingredientes(ingrediente)
        ON DELETE CASCADE  );
CREATE TABLE Clientes  (
    cliente NUMBER(5) NOT NULL,
    nombre_cliente VARCHAR2(20) NOT NULL,
    apellidos_cliente     VARCHAR2(40) NOT NULL,
    direccion_cliente     VARCHAR2(30),
    telf_cliente NUMBER(9),
    consumo_p NUMBER(3),
    consumo_b NUMBER(3),
    consumo_c NUMBER(3),
      CONSTRAINT pk_Clientes
        PRIMARY KEY (cliente),
      CONSTRAINT ck_ncl
```

```
      CHECK (nombre_cliente = INITCAP (nombre_cliente)),
    CONSTRAINT ck_acl
      CHECK (apellidos_cliente = INITCAP (apellidos_cliente)),
    CONSTRAINT ck_cop
      CHECK (consumo_p > = 0),
    CONSTRAINT ck_cob
      CHECK (consumo_b > = 0),
    CONSTRAINT ck_coc
      CHECK (consumo_c > = 0)  );
CREATE TABLE Scooters  (
   scooter VARCHAR2(6) NOT NULL,
   ano_scooter NUMBER(4),
    CONSTRAINT pk_Scooters
      PRIMARY KEY (scooter)  );
CREATE TABLE Repartidores  (
   dni_repartidor VARCHAR2(9) NOT NULL,
   nombrec_repartidor VARCHAR2(30) NOT NULL,
   scooter VARCHAR2(6) NOT NULL,
    CONSTRAINT pk_Repartidores
      PRIMARY KEY (dni_repartidor),
    CONSTRAINT fk_rep_sco
      FOREIGN KEY (scooter)
      REFERENCES Scooters(scooter)
      ON DELETE CASCADE,
    CONSTRAINT ck_nre
      CHECK (nombrec_repartidor =
            INITCAP(nombrec_repartidor))  );
CREATE TABLE Repostajes  (
   scooter VARCHAR2(6) NOT NULL,
   fecha_repostaje DATE NOT NULL,
   coste_repostaje NUMBER(3) NOT NULL,
    CONSTRAINT pk_Repostajes
      PRIMARY KEY (scooter, fecha_repostaje),
    CONSTRAINT fk_res_sco
      FOREIGN KEY (scooter)
      REFERENCES Scooters(scooter),
    CONSTRAINT ck_cre
      CHECK (coste_repostaje > = 0)  );
CREATE TABLE TiposPedidos  (
   clase_pedido VARCHAR2(10) NOT NULL,
   incremento NUMBER(3),
   minimo NUMBER(4) NOT NULL,
    CONSTRAINT pk_TiposPedidos
      PRIMARY KEY (clase_pedido),
    CONSTRAINT ck_cla
      CHECK (clase_pedido IN ('L', 'R', 'D')),
    CONSTRAINT ck_inc
      CHECK (incremento > 0),
```



```
    CONSTRAINT ck_min
      CHECK (minimo > 0)  );
CREATE TABLE Pedidos  (
   fecha_pedido DATE NOT NULL,
   pedido NUMBER(3) NOT NULL,
   clase_pedido VARCHAR2(10) NOT NULL,
   total_pedido NUMBER(4) NOT NULL,
   cliente NUMBER(5),
   dni_repartidor VARCHAR2(9),
   valor_receta NUMBER(4) NOT NULL,
   incremento NUMBER(3) NOT NULL,
    CONSTRAINT pk_Pedidos
      PRIMARY KEY (fecha_pedido, pedido),
    CONSTRAINT fk_ped_cli
      FOREIGN KEY (cliente)
      REFERENCES Clientes(cliente)
      ON DELETE CASCADE,
    CONSTRAINT fk_ped_rep
      FOREIGN KEY (dni_repartidor)
      REFERENCES Repartidores (dni_repartidor),
    CONSTRAINT fk_ped_tpe
      FOREIGN KEY (clase_pedido)
      REFERENCES TiposPedidos(clase_pedido)
      ON DELETE CASCADE,
    CONSTRAINT ck_vre
      CHECK (valor_receta > = 0),
    CONSTRAINT ck_int
      CHECK (incremento > = 0)  );
CREATE TABLE Regalos  (
   regalo NUMBER(4) NOT NULL,
   motivo CHAR(1) NOT NULL,
   limite NUMBER(2) NOT NULL,
    CONSTRAINT pk_Regalos
      PRIMARY KEY (regalo, motivo),
    CONSTRAINT ck_mot
      CHECK (motivo IN ('C', 'P'))  );
CREATE TABLE ObsequiosC  (
   regalo NUMBER(4) NOT NULL,
   motivo CHAR(1) NOT NULL,
   cliente NUMBER(5) NOT NULL,
   fecha_obsequio DATE NOT NULL,
    CONSTRAINT pk_ObsequiosC
      PRIMARY KEY (regalo, motivo, cliente, fecha_obsequio),
    CONSTRAINT fk_obsC_reg
      FOREIGN KEY (regalo, motivo)
      REFERENCES Regalos (regalo, motivo)
      ON DELETE CASCADE,
    CONSTRAINT fk_obsC_cli
      FOREIGN KEY (cliente)
```

```
      REFERENCES Clientes(cliente)
      ON DELETE CASCADE,
    CONSTRAINT ck_obsC_mot
      CHECK (motivo = 'C')  );
CREATE TABLE ObsequiosP  (
   fecha_pedido DATE NOT NULL,
   pedido NUMBER(3) NOT NULL,
   regalo NUMBER(4) NOT NULL,
   motivo CHAR(1) NULL,
    CONSTRAINT pk_ObsequiosP
      PRIMARY KEY (fecha_pedido, pedido),
    CONSTRAINT fk_obsP_reg
      FOREIGN KEY (regalo, motivo)
      REFERENCES Regalos (regalo, motivo)
      ON DELETE CASCADE,
    CONSTRAINT fk_obsP_ped
      FOREIGN KEY (fecha_pedido, pedido)
      REFERENCES Pedidos(fecha_pedido, pedido)
      ON DELETE CASCADE,
    CONSTRAINT ck_obsP_mot
      CHECK (motivo = 'P')  );
CREATE TABLE VentasProductos  (
   fecha_pedido DATE NOT NULL,
   pedido NUMBER(3) NOT NULL,
   venta NUMBER(3) NOT NULL,
   articulo VARCHAR2(10) NOT NULL,
   tamano CHAR(1),
   unidades NUMBER(2),
   valor_venta NUMBER(4),
    CONSTRAINT pk_VentasProductos
      PRIMARY KEY (fecha_pedido, pedido, venta),
    CONSTRAINT fk_ven_ped
      FOREIGN KEY (fecha_pedido, pedido)
      REFERENCES Pedidos(fecha_pedido, pedido)
      ON DELETE CASCADE,
    CONSTRAINT fk_ven_art
      FOREIGN KEY (articulo, tamano)
      REFERENCES Articulos(articulo, tamano)  );
CREATE TABLE RecetasVenta  (
   fecha_pedido DATE NOT NULL,
   pedido NUMBER(3) NOT NULL,
   venta NUMBER(3) NOT NULL,
   ingrediente VARCHAR2(4) NOT NULL,
    CONSTRAINT pk_RecetasVenta
      PRIMARY KEY (fecha_pedido, pedido, venta, ingrediente),
    CONSTRAINT fk_rve_ven
      FOREIGN KEY (fecha_pedido, pedido, venta)
      REFERENCES VentasProductos (fecha_pedido, pedido, venta)
      ON DELETE CASCADE,
```



```
    CONSTRAINT fk_rve_ing
      FOREIGN KEY (ingrediente)
      REFERENCES Ingredientes (ingrediente)  );
```

14.4.2. Manipulación de la Base de Datos

Cap14ej01.sql

Obtener el precio del artículo pizza de dos ingredientes.

De la tabla Pizzas se obtiene el valor de los ingredientes para cada tamaño de este tipo de artículos, y de la tabla Articulos, sobre la cual se realiza una reunión, se obtiene el valor base de estos productos. A final se proyecta sobre la información solicitada.

```
SELECT Pizzas.articulo, Pizzas.tamano,
       precio_articulo + (2 * precio_ingrediente) "PRECIO"
   FROM Articulos, Pizzas
   WHERE Articulos.articulo = Pizzas.articulo
         AND Articulos.tamano = Pizzas.tamano;
```

Resultado

```
ARTICULO   T    PRECIO
---------- - ---------
Jalisco    M       800
Jalisco    G      1400
Ahumados   M       900
Ahumados   G      1400
Carbonara  M       800
Carbonara  G      1400
Chips      M       800
Chips      G      1400
```

Cap14ej02.sql

Obtener la cantidad de ingresos obtenidos hasta el momento por la venta de artículos que no sean 'Pizzas', agrupados por artículos.

Se realiza una proyección sobre los atributos articulo y tamano de la tabla VentasProductos y, haciendo uso del operador SUM, se calcula el total de las ventas hasta el momento. Se hace uso de las cláusulas GROUP BY y ORDER BY para agrupar por los atributos por los cuales se proyecta y ordenar la salida.

```
SELECT articulo, tamano, SUM(valor_venta) "INGRESOS"
   FROM VentasProductos
   WHERE articulo IN
     (SELECT articulo
      FROM ClaseArticulos
      WHERE t_art NOT LIKE 'P')
   GROUP BY articulo, tamano
   ORDER BY articulo, tamano;
```

Resultado

```
ARTICULO    T  INGRESOS
----------  - ---------
Bocajamón   G      1500
Bocatort    G      1125
Bocatort    P      1125
Bocayork    G       375
Coca-Cola   G      1000
```

Cap14ej03.sql

Obtener el cliente que más pedidos ha realizado en el último mes.

De la tabla Pedidos se obtiene el máximo número de tuplas existentes que cumplen la condición de que el número de meses transcurridos entre el valor del atributo fecha_pedido y la fecha del sistema (SYSDATE) esté comprendido entre 0 y 1. Tras esto, en una nueva consulta sobre la tabla Pedidos, se obtiene el valor del atributo cliente que aparece en todas las tuplas que cumplieron las condiciones requeridas en la subconsulta anterior.

```
SELECT cliente, COUNT(cliente) "PEDIDOS"
   FROM Pedidos
   WHERE MONTHS_BETWEEN(SYSDATE,fecha_pedido) < = 1
   GROUP BY cliente
   HAVING COUNT(cliente) =
     (SELECT MAX(COUNT(cliente))
        FROM Pedidos
        WHERE MONTHS_BETWEEN(SYSDATE, fecha_pedido) < = 1
        GROUP BY cliente);
```

Resultado

```
   CLIENTE       PEDIDOS
---------- -------------
     11117             6
```

Cap14ej04.sql

Obtener el scooter y el repartidor que realizó el servicio 111 del 08/06/1999.

De la tabla Repartidores se obtiene el atributo scooter de aquellas tuplas que cumplen la condición de que el atributo dni_repartidor tiene el valor devuelto por una subconsulta sobre la tabla Pedidos del valor del atributo dni_repartidor de aquellas tuplas que cumplen la condición de que pedido es igual a 111 y fecha_pedido coincide con '08/06/1999'.

```
   /* Se formatea la salida */
COLUMN scooter FORMAT A8
SELECT scooter, dni_repartidor "DNI",
       nombrec_repartidor "NOMBRE"
   FROM Repartidores
   WHERE DNI_REPARTIDOR IN
     (SELECT dni_repartidor
       FROM Pedidos
       WHERE pedido = 111 AND fecha_pedido =
            TO_DATE('08/06/1999','DD/MM/YYYY'));
```

Resultado

```
SCOOTER  DNI       NOMBRE
-------- --------- ------------------------------
Sco1     22222222T Jorge López
```

Cap14ej05.sql

Cree una consulta que muestre la cantidad de repostajes y pedidos asignados (SERVICIOS) del repartidor 22222222T en el último mes.

Se realiza una primera consulta para obtener la cantidad de repostajes realizados por la scooter asignada al repartidor 22222222T, mostrando sólo los repostajes del último mes. Se realiza una consulta sobre la tabla pedidos para obtener la cantidad de pedidos asignados al repartidor 22222222T, obteniendo la cantidad de pedidos asignados al repartidor 22222222T. Estas consultas son tratadas como dos vistas (c_repostajes,c_pedido), sobre la cuales se realiza la consulta principal, realizando ambas consultas en una sola sentencia.

La segunda solución es similar a la anterior, con la diferencia que la cantidad de repostajes se obtiene en la primera consulta con la reunión de las tablas repartidores y repostajes, bajo la condición que el atributo scooter sea igual.


```
   /* Se formatea la salida */
COLUMN REPARTIDOR FORMAT A11
--Solución-1
SELECT '22222222T' "REPARTIDOR", c_repostajes.cant_rep
       "REPOSTAJES", c_pedido.cant_ped "SERVICIOS"
   FROM (SELECT  COUNT(*) cant_rep
           FROM Repostajes
           WHERE MONTHS_BETWEEN (SYSDATE,fecha_repostaje)<= 1
             AND scooter =
     (SELECT scooter
      FROM Repartidores
      WHERE dni_repartidor LIKE '22222222T')) c_repostajes,
      (SELECT COUNT(*) cant_ped
       FROM Pedidos
       WHERE dni_repartidor LIKE '22222222T'
       AND MONTHS_BETWEEN (SYSDATE,fecha_pedido)<= 1) c_pedido;
--Solución-2
SELECT '22222222T' "REPARTIDOR", c_repostajes.cant_rep
       "REPOSTAJES", c_pedido.cant_ped "SERVICIOS"
   FROM (SELECT COUNT(repo.scooter) cant_rep
           FROM Repartidores repa, repostajes repo
           WHERE repa.scooter=repo.scooter
           AND repa.dni_repartidor LIKE '22222222T'
           AND MONTHS_BETWEEN (SYSDATE,repo.fecha_repostaje)
               <= 1) c_repostajes,
         (SELECT COUNT(*) cant_ped
           FROM Pedidos
           WHERE dni_repartidor LIKE '22222222T'
           AND MONTHS_BETWEEN (SYSDATE,fecha_pedido)<= 1)
           c_pedido;
```

Resultado

```
REPARTIDOR  REPOSTAJES SERVICIOS
----------- ---------- ---------
22222222T            2         3
```

Cap14ej06.sql

¿Cuál es el nombre del ingrediente más utilizado en las recetas de pizzas?

Se obtiene el atributo nombre_ingrediente de la tabla Ingredientes, mostrando sólo aquellas tuplas para las que el atributo ingrediente, coincide con el valor obtenido por la subconsulta que muestra el ingrediente más utilizado en las recetas de pizzas.

```
   /* Se formatea la salida */
COLUMN "Ingrediente más utilizado" FORMAT A30
```

```
SELECT nombre_ingrediente "Ingrediente más utilizado"
   FROM Ingredientes
   WHERE ingrediente IN
     (SELECT  ingrediente
      FROM RecetasPizzas
      GROUP BY ingrediente
      HAVING COUNT(*)=
        (SELECT MAX(COUNT (*))
           FROM RecetasPizzas
           GROUP BY ingrediente));
```

Resultado

```
Ingrediente más utilizado
------------------------------
Cebolla
```

Cap14ej07.sql

El repartidor 22222222T dice que repostó la Scooter que tiene asignado 2 veces, una el 10/03/1999 y otra hoy. Comprobar si este hecho aparece reflejado en la base de datos.

Se calcula el número de tuplas de la tabla Repostajes que satisfacen que el atributo fecha_repostaje es el indicado en el enunciado, seleccionando sólo aquellas tuplas cuyo valor del atributo scooter es el obtenido de una subconsulta realizada sobre la tabla Repartidores en la que se selecciona únicamente el repartidor propuesto en el enunciado.

```
SELECT '22222222T' "REPARTIDOR", fecha_repostaje "FECHA",
       coste_repostaje  "CANTIDAD"
   FROM Repostajes
   WHERE scooter IN
     (SELECT scooter
        FROM Repartidores
        WHERE dni_repartidor = '22222222T') AND (fecha_repostaje
              = TO_DATE('10/03/1999','DD/MM/YYYY')
              OR fecha_repostaje LIKE SYSDATE);
```

Resultado

```
REPARTIDOR  FECHA       CANTIDAD
----------  ---------  ---------
22222222T  10-MAR-99         400
22222222T  13-FEB-01         400
```

Cap14ej08.sql

Subir el precio de todos los artículos de venta al público en un 10% si su precio actual es superior a 500 ptas.

Se actualizan todas las tuplas de la tabla Articulos que cumplen la condición de que su atributo precio_articulo tenga un valor superior a 500, de modo que precio_articulo pasará a valer el equivalente a incrementar en un 10% dicho valor.

```
UPDATE Articulos
   SET precio_articulo = (precio_articulo * 1.1)
    WHERE precio_articulo > 500;
```

Resultado

```
8 filas modificadas.
```


Cap14ej09.sql

Borrar todos los obsequios que no han sido recibidos por ningún cliente.

Se borran todas las tuplas de la tabla Regalos que cumplen la condición de que no existe ninguna tupla en las tablas ObsequiosC y ObsequiosP en las que aparezca el valor del atributo regalo de la tabla Regalos.

```
DELETE Regalos
   WHERE NOT EXISTS
     (SELECT *
      FROM ObsequiosC
      WHERE Regalos.regalo = ObsequiosC.Regalo) AND NOT EXISTS
        (SELECT *
           FROM ObsequiosP
           WHERE Regalos.regalo = ObsequiosP.regalo);
```

Resultado

```
2 filas borradas.
```

Cap14ej10.sql

Obtener el número de productos que se vendieron en febrero de 2000 que no eran bocadillos.

Se contabiliza el número de tuplas existentes en la tabla VentasProductos que cumplen la condición de que su atributo articulo no tiene el valor resultante de una subconsulta sobre la tabla ClaseArticulos en la que se obtiene el artículo que corresponde a los bocadillos. También deben cumplir las tuplas contabilizadas la condición de que el valor del atributo fecha_pedido esté comprendido entre el 1 de febrero de 2000 y el último día de ese mes, obtenido con la función LAST_DAY().

```
SELECT COUNT(*) "Ventas N/B (02/2000)"
   FROM VentasProductos
   WHERE articulo NOT IN
     (SELECT articulo
        FROM ClaseArticulos
        WHERE t_art = 'B') AND fecha_pedido BETWEEN
              TO_DATE('01/02/2000','DD/MM/YYYY') AND
              LAST_DAY(TO_DATE('01/02/2000','DD/MM/YYYY'));
```

Resultado

```
Ventas N/B (02/2000)
--------------------
                   3
```

Cap14ejP1.sql

Cree un procedimiento PL/SQL para cargar los datos de la tabla reagalos a partir de un fichero, con los campos delimitados por comas *(regalo, motivo, límite)*.

Se utiliza el paquete UTL_FILE para realizar las operaciones de E/S a ficheros Se detecta la posición de las comas para detectar los campos. Los métodos del paquete UTL_FILE son los siguientes:

- *UTL_FILE.FOPEN : Para abrir el fichero.*
- *UTL_FILE.GET_LINE: Para leer del fichero.*
- *UTL_FILE.FCLOSE: Para cerrar el fichero.*

```
CREATE OR REPLACE PROCEDURE cargar (
   dir_Fichero  IN VARCHAR2,
   nombre_fichero IN VARCHAR2 )
AS
   /* Se definen las variables de trabajo */
   controlador_fichero UTL_FILE.FILE_TYPE;
   nueva_linea  VARCHAR2(100);
   regalo regalos.regalo%TYPE;
   motivo regalos.motivo%TYPE;
   limite regalos.limite%TYPE;
-- Posición de las comas dentro de la línea
   primera_coma NUMBER;
   segunda_coma NUMBER;
BEGIN
   DBMS_OUTPUT.ENABLE(100000);
   /* Se abre el fichero para la lectura y se lee línea a línea
   (GET_LINE) */
   controlador_fichero := UTL_FILE.FOPEN(dir_Fichero,
      nombre_fichero, 'r');
   LOOP
     BEGIN
-- Se lee el fichero hasta que no quedan más líneas
      UTL_FILE.GET_LINE(controlador_fichero, nueva_linea);
      EXCEPTION
         WHEN NO_DATA_FOUND THEN EXIT;
     END;
   /* Como cada uno de los campos está delimitado por comas, es
   necesario encontrar dos comas en la línea para validar un
   campo, para ello se utiliza la sentencia INSTR */
     primera_coma := INSTR(nueva_linea, ',', 1, 1);
     segunda_coma := INSTR(nueva_linea, ',', 1, 2);
-- Se extraen los campos
     regalo := TO_NUMBER(SUBSTR(nueva_linea, 1,
               primera_coma - 1));
     motivo := SUBSTR(nueva_linea, primera_coma + 1,
               segunda_coma - primera_coma - 1);
     limite := TO_NUMBER(SUBSTR(nueva_linea, segunda_coma + 1));
     DBMS_OUTPUT.PUT_LINE(regalo ||' '||motivo||' '||limite);
-- Se insertan los campos en la tabla
     INSERT INTO Regalos (regalos.regalo, regalos.motivo,
                          regalos.limite)
      VALUES (regalo, motivo, limite);
     DBMS_OUTPUT.PUT_LINE('Insertado');
   END LOOP;
-- Se cierra el fichero
   UTL_FILE.FCLOSE(controlador_fichero);
```



```
COMMIT;
-- Se comprueban los posibles errores del proceso
EXCEPTION
   WHEN OTHERS THEN
     UTL_FILE.FCLOSE(controlador_fichero);
     DBMS_OUTPUT.PUT_LINE(SQLERRM);
     ROLLBACK;
END;
/
```

Prueba de ejecución

```
-- Se ejecuta el procedimiento
     EXEC cargar ('C:\Archiv~1\chencodd','regalos.dat');
```

CAPÍTULO 15

EXPLOTACIONES MINERAS

15.1. ENUNCIADO DEL PROBLEMA

Nos encontramos en el año 2069 y hace ya muchos años que se está procediendo a la explotación de los recursos naturales de los astros de nuestro sistema solar[34].

La empresa Explotaciones Mineras Interplanetarias S.A. (EMISA) ha aumentado en los últimos años el número de centros de explotación, necesitando para la gestión, organización y control de los mismos un sistema informatizado que le ayude en las tareas logísticas, administrativas y decisorias sobre los mismos.

En este sentido EMISA desea mantener información acerca de los astros del sistema solar en los cuales están o han estado en activo explotaciones mineras, las características de esas explotaciones, los minerales que se extraen, la producción de las mismas, el personal que está empleado y la labor que realiza, etc.

Asimismo, a EMISA también le interesa conocer información sobre otros astros del sistema solar a los que ya ha llegado el ser humano y sobre los cuales se tiene información sobre las características mineralógicas de los mismos, su habitabilidad, etc., con fines a la introducción de futuras explotaciones.

[34] Se podría haber elegido cualquier otro año para que de todas formas siguiera siendo un problema de ciencia-ficción.

Siguiendo con la ficción que presenta el problema propuesto vamos, a continuación, a introducir una serie de supuestos semánticos necesarios para caracterizar la naturaleza del problema planteado:

SUPUESTO 1: Sólo se va a considerar que las explotaciones mineras de *EMISA* se realizan en nuestro sistema solar[35].

SUPUESTO 2: A *EMISA* le interesa conocer las características mineralógicas de todos los astros del sistema solar sobre los cuales se tenga información, con independencia de que en ellos esté activa alguna explotación (Factoría) de la compañía.

SUPUESTO 3: A *EMISA* le interesa conocer información acerca de los lugares en los cuales se han realizado prospecciones mineras. De estos lugares le interesa conocer su situación, minerales encontrados e información correspondiente a la posible explotación de los recursos minerales del mismo.

SUPUESTO 4: De las explotaciones mineras en activo, a *EMISA* le interesa conocer toda la información correspondiente a las mismas. Es decir, minerales que se están extrayendo, cantidad del mineral que se extrae diariamente, calidad de ese mineral en el proceso de transformación, etc.

SUPUESTO 5: De cada explotación en activo se realizan envíos de cargamentos de mineral a la tierra. Estos envíos son realizados en naves de la propia compañía de las cuales *EMISA* desea mantener también información. Los cargamentos de mineral son almacenados en sus correspondientes lugares desde que se extraen hasta que son enviados a la tierra.

SUPUESTO 6: De cada uno de los envíos de mineral a la tierra, a la compañía le interesa conocer información correspondiente a las personas (astronautas y personal de apoyo) implicadas, además de los datos correspondientes a la duración del viaje, escalas, etc.

SUPUESTO 7: En la tierra, *EMISA* tiene diferentes centros de transformación del mineral. Un centro, en la tierra, puede transformar sólo un tipo de mineral, aunque *EMISA* por razones estratégicas dispone de más de un centro que transforme un mismo mineral. Cada centro tiene un nombre que le identifica sin ambigüedad.

SUPUESTO 8: En cada envío de mineral a la tierra sólo se transporta un tipo de mineral, aunque éste puede ser recogido de más de una explotación minera. Por ello, hay que considerar el mineral que envía a la tierra cada explotación en cada envío.

SUPUESTO 9: A *EMISA* le interesa conocer datos acerca de su personal, por lo que tendrá que manejar información sobre cada empleado: además de su información

[35] Sería una ficción excesiva que en el 2069 ya hubiéramos alcanzado otras galaxias, ¿o a lo mejor no?

general, le interesa conocer información histórica de sus destinos fuera de la tierra, así como los trabajos realizados.

SUPUESTO 10: Los trabajos u ocupaciones que puede realizar el personal de *EMISA* están perfectamente catalogados y todos ellos tienen asignado un nivel de responsabilidad y un salario con independencia del destino del mismo.

SUPUESTO 11: Muchas de las explotaciones mineras de *EMISA* no pueden autoabastecerse de los recursos naturales para la supervivencia del personal que trabaja en ellas. Por ello, *EMISA* envía periódicamente naves con suministros a estas explotaciones, ya que le interesa conocer la periodicidad con la que se realizan estos envíos, el cargamento que deben llevar estas naves y el coste que supone el envío. Las prospecciones pueden autoabastecerse (tienen un tiempo corto de duración) y, por tanto, no reciben suministros desde la tierra.

SUPUESTO 12: El cargamento de suministro para la supervivencia de los empleados en las explotaciones mineras lo realiza *EMISA* basándose en un factor (unidades) que necesita cada empleado de cada uno de los productos alimenticios que los médicos de la empresa consideran apropiados. Así, la empresa cuenta con un almacén de productos alimenticios y a cada uno de los productos le tienen asignado un factor de consumo por persona y mes, el cual es utilizado para calcular los cargamentos enviados a las explotaciones mineras.

SUPUESTO 12+1: Los viajes de suministros y recogidas a las factorías que tiene *EMISA* son realizados por personal especializado de la tierra, no interviniendo el personal de las factorías.

15.2. MODELO CONCEPTUAL

Se trata de un problema combinado de almacenaje y suministros de material de una empresa (con un toque de ciencia-ficción) de forma que la solución propuesta (naturalmente, con la adaptación correspondiente) puede aplicarse a cualquier problema de este tipo.

Del problema propuesto pueden extraerse los siguientes objetos:

15.2.1. Los lugares de interés para *EMISA*

A la empresa únicamente le interesa conocer aquellos lugares del sistema solar en los cuales tiene factorías y/o ha realizado prospecciones, además le interesa conocer aquellos lugares en los cuales pueden establecer factorías en un futuro. Así, podemos definir los siguientes tipos de entidad e interrelación:

Tipo de entidad *Astros*: representando a "*cualquier astro del sistema solar*" (SUPUESTOS 1 y 2). Este tipo de entidad se puede especializar de forma parcial y exclusiva en dos subtipos de entidad: *Planetas* y *Satelites*, los cuales mantienen un tipo de interrelación débil por existencia, puesto que los satélites siempre están

asociados a un planeta sobre el cual mantienen una órbita gravitacional. La especialización parcial permitirá poder representar en el supertipo a cualquier otro astro estelar (asteroides, por ejemplo)[36] (ver figura 15.1).

Para estos tipos de entidad sólo se van a considerar los atributos: *nombre_astro* —como identificador principal— y *od_astro*, para representar cualquier información de interés para *EMISA* acerca del mismo.

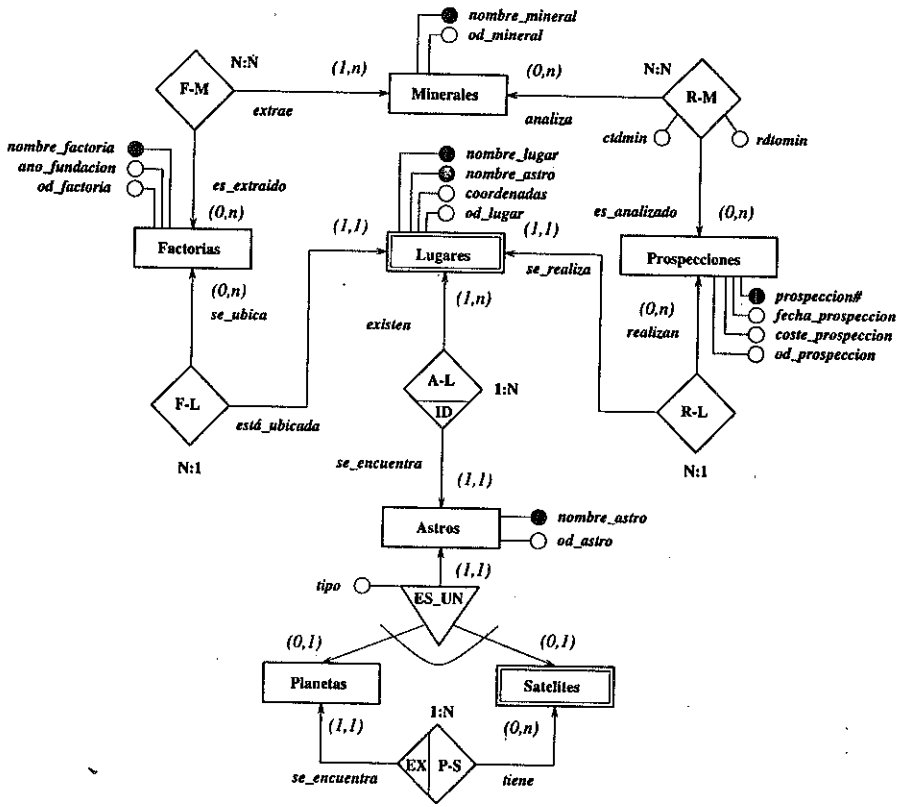


Figura 15.1 Las ubicaciones de EMISA

Tipo de entidad *Lugares*: representando a "*cualquier lugar de un astro que sea de interés para la empresa*", bien por que se haya establecido una factoría, o bien por que se haya hecho una prospección (SUPUESTO 3).

Este tipo de entidad es débil por identificación con respecto al tipo de entidad *Astros* si se considera que en diferentes astros pueden existir lugares con el mismo nombre. Para este tipo de entidad se van a considerar los atributos ***nombre_lugar***, *coordenadas* y *od_lugar*.

El identificador de este tipo de entidad será la agregación de los atributos *nombre_astro* (heredado del tipo de entidad *Astros*) y ***nombre_lugar***.

Se podrían haber elegido como atributo e identificador único de este tipo de entidad las coordenadas del mismo, las cuales no se repiten de un astro a otro aunque el nombre del lugar sea el mismo. Pero aparecería otro problema, y es que en unas mismas coordenadas de un astro existieran dos factorías y/o prospecciones de la empresa, no sirviendo, en tal caso, este atributo como identificador (ver figura 15.1).

[36] El problema podría haberse ampliado fácilmente fuera del sistema solar, simplemente con haber considerado un tipo de entidad *Galaxias* y representar un tipo de interrelación, también débil por existencia, entre este tipo de entidad y el tipo de entidad *Astros*.

Se puede introducir un nuevo supuesto que garantice la identificación única propuesta:

SUPUESTO 14: Los lugares de un astro tienen diferente nombre independientemente de las coordenadas del mismo.

15.2.2. Las factorías y las prospecciones de minerales

EMISA está dedicada a la explotación de recursos mineros. Por ello, un tipo de entidad fuerte a considerar es precisamente los minerales que son de interés para la empresa.

Definiremos, por tanto, un tipo de entidad *Minerales*, con atributo identificador *nombre_mineral* y con otro atributo *od_mineral* para representar toda la información que le interesa a la empresa sobre estos objetos.

Por otra parte, tanto las factorías como las prospecciones son realizadas en distintos lugares del sistema solar para explotar los recursos de uno o varios minerales o para intentar encontrar las vetas de los mismos. Se pueden entonces definir los siguientes tipos de entidad:

Tipo de entidad *Factorias*: representando los distintos centros de explotación de minerales. Para este tipo de entidad vamos a considerar como identificador el atributo *nombre_factoria*, y otros atributos como: *ano_fundacion* y *od_factoria*.

Si consideramos que *EMISA* no da nombres duplicados a sus factorías, se trata de un tipo de entidad fuerte que mantiene un tipo de interrelación ***N:1*** con el tipo de entidad *Lugares*, y un tipo de interrelación ***N:N*** con el tipo de entidad *Minerales*, como se muestra en la figura 15.1 (SUPUESTOS 3 y 4).

Tipo de entidad *Prospecciones*: por otro lado, se puede considerar que *EMISA* le asigna un código único a cada uno de sus proyectos de prospección de minerales. Consideraremos, entonces, un tipo de entidad ***Prospecciones*** que represente a cada uno de estos proyectos de búsqueda de nuevos yacimientos mineros (SUPUESTO 3).

Para este tipo de entidad se considerará el atributo *prospeccion#* como identificador principal, y los atributos *fecha_prospeccion, coste_prospeccion* y *od_prospeccion* para representar la información correspondiente a las entidades de este tipo.

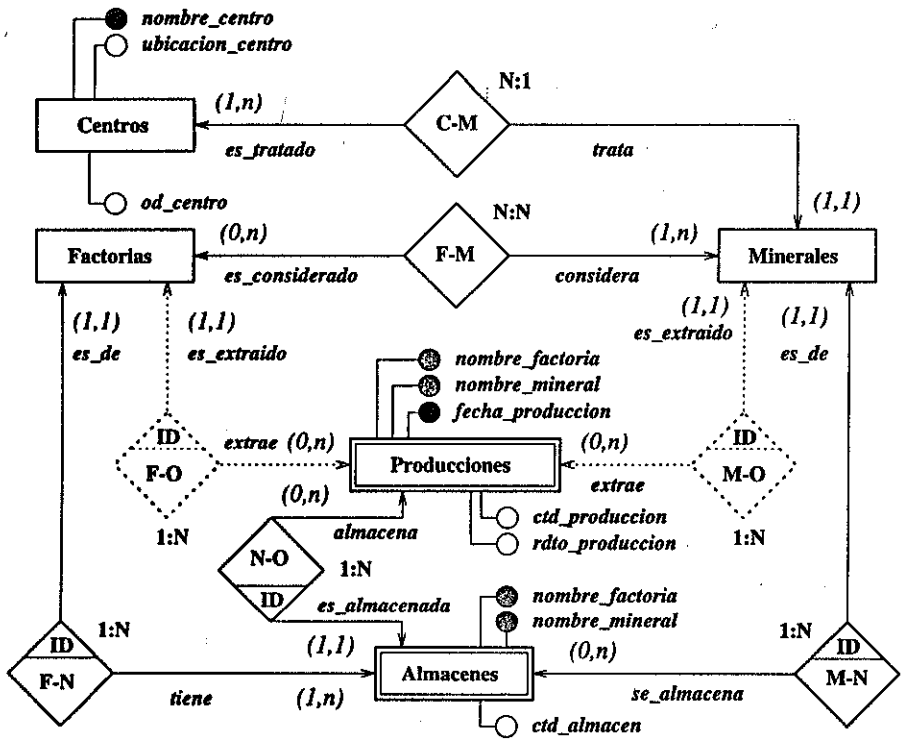


Figura 15.2 Las factorías y las explotaciones de mineral

El tipo de entidad *Prospecciones* participa en un tipo de interrelación *N:1* con el tipo de entidad *Lugares*, representando que cada prospección se realiza en un único lugar. Además, participa en un tipo de interrelación *N:N* con el tipo de entidad *Minerales*, puesto que en un proyecto de prospección se pueden encontrar varios minerales. Este tipo de interrelación *(R-M)* está cualificado por dos atributos: *ctdmin* y *rdtomin* que representan la cantidad de mineral en la veta y el rendimiento esperado en la explotación de la misma (ver figura 15.1), como así introduce el SUPUESTO 3.

15.2.3. Las producciones de mineral

De forma distinta a como *EMISA* trata los proyectos de prospección de mineral, son tratadas las extracciones de éstos en las factorías. A la empresa le interesa conocer cuál es la producción de estas factorías en cada uno de estos minerales y, además, hay

que tener en cuenta que estas producciones son almacenadas y posteriormente enviadas a la tierra a los centros de refinamiento y transformación correspondientes. Por ello, es necesario definir otros tipos de entidad:

Tipo de entidad *Producciones*: representando la extracción que realizan las factorías diariamente de cada uno de los minerales con los que trabaja. Se trata de un tipo de entidad débil con respecto a los tipos de entidad *Factorias* y *Minerales* y, por tanto, hereda los atributos identificadores de éstos. Además, y formando parte también del identificador, está presente el atributo *fecha_produccion* representando el día de extracción, y los atributos: *rdto_produccion* y *ctd_produccion* representando el rendimiento de la producción y la cantidad de mineral extraída, respectivamente (SUPUESTO 4).

Se puede apreciar fácilmente que este tipo de entidad está representando un tipo de interrelación ternaria entre los tipos de entidad *Factorias, Minerales* y *Fecha*, y que por razones de claridad y sencillez del modelo propuesto es más conveniente abstraerlo a un tipo de entidad débil.

Si se consideran, entonces, los tipos de interrelación *(F-O)* y *(M-O)*, entonces el tipo de interrelación *(F-M)*, entre los tipos de entidad *Factorias* y *Minerales* podría resultar redundante (ver figura 15.2), a no ser que se quiera representar información de los minerales que se pueden extraer en una factoría con independencia de que haya habido, hasta la fecha, alguna explotación de la veta (criterio que se considerará en este ejemplo)[37].

Tipo de entidad *Almacenes*: por otro lado el mineral que se extrae diariamente se va almacenando en cada factoría (SUPUESTO 5). Cada factoría cuenta con almacenes independientes para cada uno de los minerales, desde los cuales se realizan los distintos envíos a la tierra. La naturaleza de este tipo de entidad ha sido ampliamente descrito en el Capítulo 10 *(Venta y Consumo de Cigarrillos)*, por lo que no nos detendremos en ella (véase la figura 15.2).

15.2.4. Los suministros y recogidas

EMISA realiza dos tipos de viajes a sus factorías. Unos para recoger el mineral de los almacenes o silos, y otros para proporcionar suministros alimenticios al personal que trabaja en las mismas. Para realizar estos viajes, la empresa cuenta con una flota de naves, por una parte, y un almacén de alimentos por otra (SUPUESTOS 5 y 11).

Comenzaremos por las recogidas de mineral. Con cierta periodicidad, se realizan cargamentos de mineral de los diferentes almacenes a la tierra y estos cargamentos son recogidos por la nave enviada por la empresa. Cada cargamento está compuesto del mineral existente en un almacén de una factoría (no pueden enviarse minerales mezclados) y en un viaje se pueden recoger cargamentos de almacenes de diferentes factorías (SUPUESTOS 5 y 8).

[37] Si se desea considerar el otro criterio bastaría con eliminar el tipo de interrelación *(F-M)*

Se pueden, entonces, definir los siguientes tipos de entidad:

Tipo de entidad *Recogidas*: representando a "*los diferentes viajes que se realizan para recoger cargamentos de las factorías*" (SUPUESTO 5).

Tipo de entidad *Cargamentos*: representando a "*las distintas partidas de mineral que se envían desde los almacenes a la tierra*" (SUPUESTO 8).

El tipo de entidad *Cargamentos* vendrá identificado por un número de cargamento que le asigna cada factoría al envío, por ejemplo. Pero si se supone (para complicar algo más el problema) que distintos almacenes de una misma factoría pueden dar el mismo número, entonces hay que considerar al tipo de entidad *Cargamentos* débil por identificación con respecto al tipo de entidad *Almacenes*, y utilizará, entonces, como identificador la agregación de los atributos *nombre_factoria, nombre_mineral* (heredados del tipo de entidad *Almacenes*) y *num_carga* (representando el número o código asignado al cargamento).

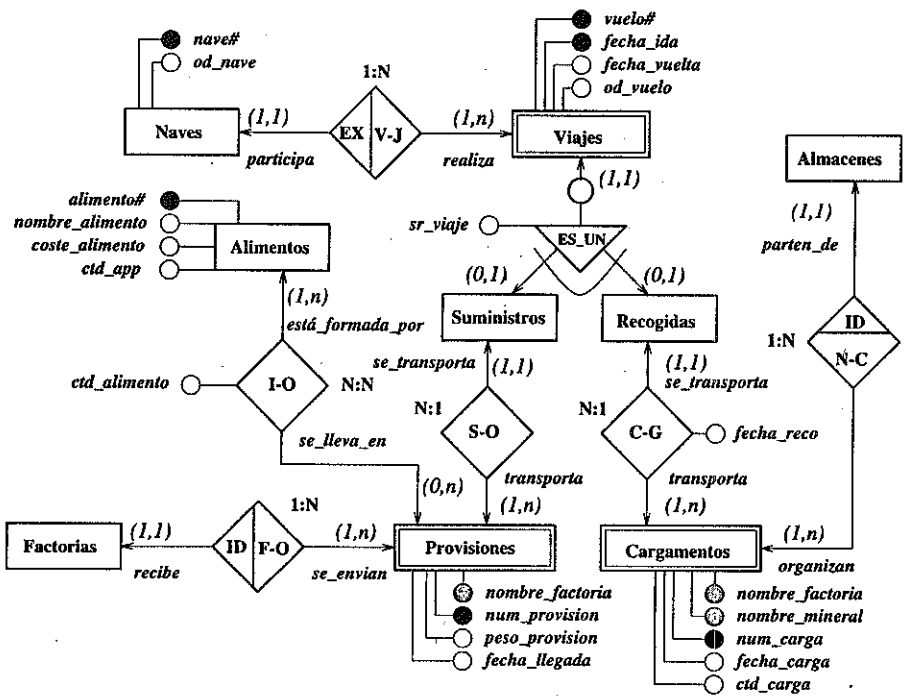


Figura 15.3 Los viajes de suministros y las recogidas

Además, este tipo de entidad tendrá otros atributos como: *fecha_carga* y *ctd_carga* representando la fecha en que se preparó el cargamento y la cantidad de mineral que se envía a la tierra (ver figura 15.3).

Por otro lado, el tipo de entidad *Cargamentos* participa con el tipo de entidad *Recogidas* en un tipo de interrelación *N:1* (SUPUESTO 8), puesto que en un viaje se pueden recoger cargamentos de distintas factorías, y este tipo de interrelación viene caracterizado por el atributo *fecha_reco*, representando la fecha en que se cargó el mineral en la nave (ver figura 15.3).

El caso de los viajes dedicados a enviar alimentos a las factorías es muy similar al anterior, y se pueden definir los siguientes objetos:

Tipo de entidad *Suministros*: representando a "*los diferentes viajes que se realizan para enviar alimentos a las factorías*" (SUPUESTO 11).

Tipo de entidad *Provisiones*: representando a "*las distintas partidas de alimentos que se envían desde la tierra a las factorías*" (SUPUESTO 12).

Y un nuevo tipo de entidad denominado *Alimentos*, representando a todos aquellos artículos alimenticios con que cuenta *EMISA* para proveer de provisiones a las distintas factorías, y para el cual se han definido los atributos *alimento#* como identificador de cada alimento, *nombre_alimento, coste_alimento* y *ctd_app*, representando, este último, la cantidad de alimento por persona y mes necesaria para su supervivencia en las distintas factorías (SUPUESTO 12).

El tipo de entidad *Provisiones* es muy similar al tipo de entidad *Cargamentos*, sólo que en este caso está relacionado con el tipo de entidad *Factorias* en lugar de con *Almacenes*, pues los suministros son enviados a todo el personal que trabaja en una determinada factoría. Para este tipo de entidad se han considerado los atributos: *nombre_factoria*, heredado del tipo de entidad *Factorias* con la que mantiene un tipo de interrelación débil por identificación, agregado con el atributo *num_provision* (el código o número de expedición del viaje destinado a la provisión de las distintas factorías) como identificador, *peso_provision* representando el peso total de la provisión que se envía, y *fecha_llegada*, que representa la fecha de llegada de cada provisión de alimentos a cada factoría.

Además, el tipo de entidad *Provisiones* está relacionado con el tipo de entidad *Alimentos* en un tipo de interrelación *N:N* (como se muestra en la figura 15.3) el cual está cualificado por el atributo *ctd_alimento* que representa la cantidad de alimento que se envía en cada suministro.

15.2.5. Las naves y los viajes

Como se ha descrito anteriormente, la empresa realiza vuelos destinados a la recogida de mineral de los distintos almacenes y al suministro de alimentos a las factorías correspondientes. *EMISA* cuenta para ello con una flotilla de naves que realizan una serie de viajes. Por tanto, deben ser definidos los siguientes objetos en el modelo conceptual (ver figura 15.3):

Tipo de entidad *Naves*: representando a "*la flotilla de naves que son propiedad de la empresa y son usadas en la realización de los viajes de suministros y recogidas*". Para este tipo de entidad se van a definir los atributos: *nave#*, representando el nombre, número o cualquier sarta de caracteres que identifique a cada una de estas naves, y *od_nave*, para representar cualquier otra información de interés acerca de las mismas.

Tipo de entidad *Viajes*: representando a "*cada uno de los viajes que realizan estas naves para suministrar alimentos o recoger mineral de las factorías*". Si se considera que existen una serie de rutas preestablecidas y que a cada una de estas rutas se le tiene asignado un código identificador (al igual que ocurre actualmente con los vuelos comerciales), el atributo *vuelo#* sería un identificador adecuado para este tipo de entidad. Además, se pueden considerar otros atributos como: *fecha_ida, fecha_vuelta* y *od_vuelo* para representar la fecha de salida y llegada del viaje y cualquier otro tipo de información, respectivamente.

El tipo de entidad *Viajes* participa, como débil por existencia, en el tipo de interrelación *(V-J)* con el tipo de entidad *Naves*, puesto que sin una nave no puede realizarse un viaje.

Por otra parte, este tipo de entidad puede especializarse, como muestra la figura 15.3, en dos subtipos: *Suministros* y *Recogidas*, representando los distintos tipos de viajes que pueden realizarse. Este tipo de interrelación jerárquica es total y exclusiva, puesto que un viaje es para recoger mineral o para suministrar pero no para ambos cometidos.

15.2.6. Los centros de transformación

Es el SUPUESTO 7 el que introduce la necesidad de considerar un tipo de entidad que represente a aquellos centros de transformación con que cuenta *EMISA* para el tratamiento de los distintos cargamentos de mineral enviados desde las factorías.

El tipo de entidad *Centros* vendrá identificado por el atributo *nombre_centro*, pudiéndose considerar otros atributos como *ubicacion_centro* y *od_centro* para representar el nombre del centro —en la empresa no hay dos centros con el mismo nombre—, su ubicación y cualquier otro tipo de información.

Este tipo de entidad participa en el siguiente tipo de interrelación:

Tipo de interrelación *Centros/Mineral* **(C-M)**: representando los minerales que pueden ser tratados en cada centro. Se trata de un tipo de interrelación *1:N*, como se muestra en la figura 15.2, puesto que un mineral es tratado en varios centros, mientras que cada centro puede tratar sólo un mineral (SUPUESTO 7).

15.2.7. El personal de *EMISA*

Los SUPUESTOS 9 y 10 introducen la necesidad de considerar al personal que tiene la empresa, así como los diferentes trabajos que éstos realizan a lo largo de su

trayectoria profesional. Podemos considerar, entonces, los siguientes elementos en el modelo conceptual:

Tipo de entidad *Personal*: representando a "*todos aquellos trabajadores de la empresa*". Para este tipo de entidad se van a considerar los atributos: *id_personal, nombrec_personal* y *od_personal*.

Tipo de entidad *Categorias*: en el SUPUESTO 10 se informa que la empresa considera una serie de categorías profesionales a las cuales les tiene asignado un salario, y que el personal de una categoría determinada tiene un sueldo con independencia de su destino.

Se puede considerar un tipo de entidad *Categorias* para representar estas categorías profesionales, en la que están presentes los atributos: *trabajo#*, como identificador principal y representando el tipo de trabajo que considera la empresa, *categoria*, para representar el nivel profesional de ese trabajo, y *sueldo* representando el sueldo asignado a esa categoría.

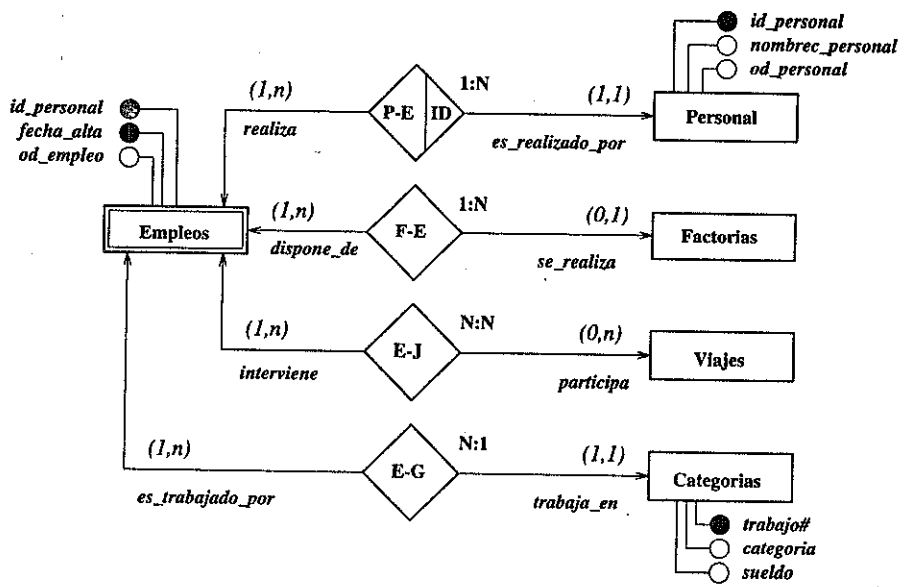


Figura 15.4 Los trabajadores de EMISA *y sus destinos*

Como indica el SUPUESTO 9, existe la necesidad de representar la labor desempeñada por cada trabajador de la empresa a lo largo de su vida profesional. Por ello, se deben considerar los siguientes tipos de entidad (ver figura 15.4).

Tipo de entidad *Empleos*: representando a "*cada uno de los trabajos realizados por el personal de la empresa*". Se trata de un tipo de entidad débil por identificación

con respecto al tipo de entidad *Personal* de la cual hereda su atributo identificador. Además, y formando parte también del identificador de este tipo de entidad, se incorpora el atributo *fecha_alta*, representando la fecha de alta en el trabajo por los trabajadores, y *od_empleo* para representar cualquier otra información de interés.

El tipo de entidad *Empleos* va a mantener dos tipos de interrelación:

Tipo de interrelación *Factorias/Empleos* **(F-E)**: representando los distintos destinos que ha tenido y tiene el personal de la empresa en las diferentes factorías desempeñado un trabajo determinado.

Se trata de un tipo de interrelación *1:N*, de forma que en una factoría están trabajando *(1,n)* trabajadores realizando cada uno un trabajo, mientras que un trabajador realizando un trabajo (un empleo) puede existir o no (está en la tierra) en una factoría *(0,1)*, como así se ha representado en la figura 15.4.

Por otra parte, el tipo de entidad *Empleos* participa en un tipo de interrelación *1:N* con el tipo de entidad *Viajes* (ver figura 15.4), lo que permite representar a aquellos empleados de la empresa que realizan cada uno de los viajes de abastecimiento y recogida a las factorías[38].

15.3. MODELO RELACIONAL

A partir del modelo conceptual se deriva el siguiente esquema relacional:

BASE DE DATOS DE LAS EXPLOTACIONES MINERAS

Astros	*(nombre_astro, tipo, od_astro)*
Satelites	*(nombre_satelite,* **nombre_planeta***)*
Ubicaciones	*(nombre_lugar,* **nombre_astro***, coordenadas, od_lugar)*
Factorias	*(nombre_factoria, ano_fundacion, od_factoria,* **nombre_lugar, nombre_astro***)*
Prospecciones	*(prospeccion#, fecha_prospeccion, coste_prospeccion, od_prospeccion,* **nombre_lugar, nombre_astro***)*
Minerales	*(nombre_mineral, od_mineral, nombre_centro)*

/ La tabla Centros no será directamente definida en el esquema, sino que será incluida como un objeto dentro de la tabla Minerales para representar la relación existente entre ambas */*

Centros	*(nombre_centro, ubicacion_centro, od_centro,* **nombre_mineral***)*

[38] En este ejemplo se ha tratado de una forma muy sencilla la problemática del personal de una empresa. Los sistemas de personal son, por sí mismos, suficientemente complejos como para necesitar una dedicación exclusiva en el estudio y análisis de los mismos.

Hallazgos	*(***prospeccion#, nombre_mineral***, ctdmin, rdtomin)*
Explotaciones	*(nombre_factoria, nombre_mineral)*
Producciones	*(***nombre_factoria, nombre_mineral***, fecha_produccion, ctd_produccion, rdto_produccion)*
Silos	*(***nombre_factoria, nombre_mineral***, ctd_almacen)*
Naves	*(nave#, od_nave)*
Viajes	*(vuelo#, fecha_ida, nave#, fecha_vuelta, od_vuelo, sr_viaje)*
Cargamentos	*(***nombre_factoria, nombre_mineral***, num_carga, fecha_carga, ctd_carga,* **vuelo#, fecha_ida***, fecha_reco)*
Comidas	*(alimento#, nombre_alimento, ctd_app, coste_alimento)*
Provisiones	*(***nombre_factoria***, num_provision, peso_provision, fecha_llegada,* **vuelo#, fecha_ida***)*
Viveres	*(***nombre_factoria, num_provision, alimento#***, ctd_alimento)*
Categorias	*(trabajo#, categoria, sueldo)*
Personal	*(id_personal, nombrec_personal, od_personal)*
Empleos	*(***id_personal, fecha_alta***,* **trabajo#***, od_empleo)*
Destinados	*(***id_personal, fecha_alta***,* **nombre_factoria***)*
Tripulantes	*(***id_personal, fecha_alta, vuelo#, fecha_ida***)*

- Las tablas *Astros* y *Satelites* por aplicación de la regla *PRTECAR-4*, eliminando el tipo de interrelación jerárquica, además de la tabla *Satelites* por aplicación de la regla *RTECAR-3.1*. Como se puede observar no se ha considerado una tabla para el tipo de entidad *Planetas* debido a que sería una representación redundante por no aportar información alguna. La relación existente entre los tipos de entidad *Satelites* y *Planetas* puede verse como una relación reflexiva entre el tipo de entidad *Astros*, lo que por aplicación de la regla *RTECAR-4* daría lugar a las tablas consideradas.
- La tabla *Ubicaciones* por aplicación de la regla *RTECAR-3.1* al tipo de interrelación *(A-L)*, *(F-L)* y *(R-L)*.
- La tabla *Factorias* por aplicación de la regla *RTECAR-1* al tipo de entidad de igual nombre, y la regla *RTECAR-3.1* al tipo de interrelación *(F-L)*.
- La tabla *Prospecciones* por aplicación de la regla *RTECAR-1* al tipo de entidad de igual nombre, y la regla *RTECAR-3.1* al tipo de interrelación *(R-L)*.
- La tabla *Hallazgos* por aplicación de la regla *RTECAR-4* al tipo de interrelación *(R-M)*.
- La tabla *Minerales* por aplicación de la regla *RTECAR-1* al tipo de entidad de igual nombre.

- La tabla *Centros* por aplicación de la regla *RTECAR-1* al tipo de entidad de igual nombre, y la regla *RTECAR-3.1* al tipo de interrelación *(C-M)*.
- La tabla *Explotaciones* por aplicación de la regla *RTECAR-4* al tipo de interrelación *(F-M)*.
- La tabla *Producciones* por aplicación de la regla *RTECAR-3.1* al tipo de interrelación *(N-O)*.
- La tabla *Silos* por aplicación de la regla *RTECAR-3.1* a los tipos de interrelación *(F-N)* y *(M-N)*.
- La tabla *Naves* por aplicación de la regla *RTECAR-1* al tipo de entidad del mismo nombre.
- La tabla *Viajes* por aplicación de la regla *PRTECAR-4* al tipo de interrelación jerárquica en la que participa el tipo de entidad del mismo nombre se eliminan los subtipos correspondientes, así como por aplicación de la regla *RTECAR-3.1* al tipo de interrelación *(V-J)*.
- La tabla *Cargamentos* por aplicación de la regla *RTECAR-3.1* a los tipos de interrelación *(N-C)* y *(C-G)*.
- La tabla *Provisiones* por aplicación de la regla *RTECAR-3.1* a los tipos de interrelación *(F-O)* y *(S-O)*.
- La tabla *Comidas* por aplicación de la regla *RTECAR-1* al tipo de entidad *Alimentos*.
- La tabla *Viveres* por aplicación de la regla *RTECAR-4* al tipo de interrelación *(I-O)*.
- La tabla *Categorias* por aplicación de la regla *RTECAR-1* al tipo de entidad del mismo nombre.
- La tabla *Personal* por aplicación de la regla *RTECAR-1* al tipo de entidad del mismo nombre.
- La tabla *Empleos* por aplicación de la regla *RTECAR-3.1* a los tipos de interrelación *(P-E)* y *(E-G)*.
- La tabla *Destinados* por aplicación de la regla *RTECAR-3.2* al tipo de interrelación *(F-E)*.
- La tabla *Tripulantes* por aplicación de la regla *RTECAR-4* al tipo de interrelación *(E-J)*.

Todas las tablas que forman parte del esquema relacional se encuentran normalizadas en *FNBC* y, por lo tanto, pueden entrar a formar parte del esquema. La razón de esta normalización directa es el hecho de simplificar en el esquema conceptual las relaciones complejas (*n-arias*) considerándolas como tipos de entidad débiles. Ésta es una práctica usual en el proceso de análisis de los problemas y facilita la posterior traducción de los esquemas conceptuales. Aunque no se pueden dar unas reglas generales para que se pueda aplicar este proceso de simplificación, sí se ha

creído conveniente introducirlo en éste y otros ejemplos propuestos en la obra con el fin de que el lector se familiarice con esta práctica.

15.4. CONSTRUCCIÓN Y USO DE LA BASE DE DATOS

Una vez normalizado el modelo relacional propuesto, procederemos a la definición sintáctica del esquema relacional correspondiente.

15.4.1. Definición sintáctica de las tablas

```
   /* Inicialmente se borran las tablas y los tipos de objetos
   definidos previamente */
DROP TABLE Tripulantes;
DROP TABLE Destinados;
DROP TABLE Empleos;
DROP TABLE Personal;
DROP TABLE Categorias;
DROP TABLE Viveres;
DROP TABLE Provisiones;
DROP TABLE Comidas;
DROP TABLE Cargamentos;
DROP TABLE Viajes;
DROP TABLE Naves;
DROP TABLE Producciones;
DROP TABLE Silos;
DROP TABLE Explotaciones;
DROP TABLE Hallazgos;
DROP TABLE Minerales;
DROP TYPE Tipo_Centros;
   /* Los tipos se borran cuando se procede a reconstruir
   completamente el esquema de la base de datos */
DROP TYPE Centro;
DROP TABLE Prospecciones;
DROP TABLE Factorias;
DROP TABLE Ubicaciones;
DROP TABLE Satelites;
DROP TABLE Astros;
   /* Se define un tipo de objeto Centro con los atributos
   correspondientes a la tabla Centros definida en el esquema
   relacional */
CREATE OR REPLACE TYPE Centro AS OBJECT  (
   nombre_centro VARCHAR2(15),
   ubicacion_centro VARCHAR2(20),
   od_centro VARCHAR2(2000) )
   /
   /* Se define un tipo de objeto Tipo_Centros con una
   estructura tabular (AS TABLE) de objetos Centro definidos
   previamente */
```

```
CREATE TYPE Tipo_Centros AS TABLE OF Centro
   /
   /* Se definen las tablas del esquema relacional */
CREATE TABLE Astros  (
   nombre_astro VARCHAR2(15) NOT NULL,
   tipo VARCHAR2(10),
   od_astro LONG,
    CONSTRAINT pk_ast
     PRIMARY KEY (nombre_astro),
    CONSTRAINT ck_nas
     CHECK (nombre_astro = INITCAP(nombre_astro)) );
CREATE TABLE Satelites  (
   nombre_satelite VARCHAR2(15) NOT NULL,
   nombre_planeta VARCHAR2(15) NOT NULL,
    CONSTRAINT pk_sat
     PRIMARY KEY (nombre_satelite),
    CONSTRAINT fk_sat_ast
     FOREIGN KEY (nombre_satelite)
     REFERENCES Astros(nombre_astro)
     ON DELETE CASCADE,
    CONSTRAINT fk_pla_ast
     FOREIGN KEY (nombre_planeta)
     REFERENCES Astros(nombre_astro)
     ON DELETE CASCADE,
    CONSTRAINT ck_nst
     CHECK (nombre_satelite = INITCAP(nombre_satelite)) );
CREATE TABLE Ubicaciones  (
   nombre_lugar VARCHAR2(15) NOT NULL,
   nombre_astro VARCHAR2(15) NOT NULL,
   coordenadas NUMBER(6),
   od_lugar LONG,
    CONSTRAINT pk_lug
     PRIMARY KEY (nombre_lugar, nombre_astro),
    CONSTRAINT fk_lug_ast
     FOREIGN KEY (nombre_astro)
     REFERENCES Astros(nombre_astro)
     ON DELETE CASCADE,
    CONSTRAINT ck_nlu
     CHECK (nombre_lugar = INITCAP(nombre_lugar)) );
CREATE TABLE Factorias  (
   nombre_factoria VARCHAR2(15) NOT NULL,
   ano_fundacion NUMBER(4),
   od_factoria LONG,
   nombre_lugar VARCHAR2(15) NOT NULL,
   nombre_astro VARCHAR2(15) NOT NULL,
    CONSTRAINT pk_fac
     PRIMARY KEY (nombre_factoria),
    CONSTRAINT fk_fac_lug
```



```
     FOREIGN KEY (nombre_lugar, nombre_astro)
     REFERENCES Ubicaciones(nombre_lugar, nombre_astro)
     ON DELETE CASCADE,
    CONSTRAINT ck_nfa
     CHECK (nombre_factoria = INITCAP(nombre_factoria)) );
CREATE TABLE Prospecciones  (
   prospeccion NUMBER(5) NOT NULL,
   fecha_prospeccion DATE NOT NULL,
   coste_prospeccion NUMBER(6),
   od_prospeccion LONG,
   nombre_lugar VARCHAR2(15) NOT NULL,
   nombre_astro VARCHAR2(15) NOT NULL,
    CONSTRAINT pk_pro
     PRIMARY KEY (prospeccion),
    CONSTRAINT fk_pro_lug
     FOREIGN KEY (nombre_lugar, nombre_astro)
     REFERENCES Ubicaciones(nombre_lugar, nombre_astro)
     ON DELETE CASCADE );
   /* Haciendo uso de las posibilidades de Oracle, se
   representa la relación existente entre Minerales y Centros a
   través de la inclusión del tipo de objeto Centros definido
   anteriormente en la tabla Minerales como una tabla anidada
   (Nested Table). Esta opción nos permite representar
   directamente la relación 1:N entre los tipos de entidades
   Minerales y Centros, ya que la información de los Centros se
   encuentra formando parte de la estructura de la tabla
   Minerales como una tabla anidada (un conjunto de tuplas),
   que representa el conjunto de centros en los cuales puede
   ser tratado cada uno de los minerales */
CREATE TABLE Minerales  (
   nombre_mineral VARCHAR2(15) NOT NULL,
   od_mineral LONG,
   Centros Tipo_Centros,
    CONSTRAINT pk_nmi
     PRIMARY KEY (nombre_mineral),
    CONSTRAINT ck_nmi
     CHECK (nombre_mineral = INITCAP(nombre_mineral)) )
    NESTED TABLE Centros STORE AS Centros_Tabla (
     (PRIMARY KEY (NESTED_TABLE_ID, nombre_centro))
     ORGANIZATION INDEX COMPRESS);
CREATE TABLE Hallazgos  (
   prospeccion NUMBER(5) NOT NULL,
   nombre_mineral VARCHAR2(15) NOT NULL,
   ctdmin NUMBER(6),
   rdtomin NUMBER(2,2),
    CONSTRAINT pk_hal
     PRIMARY KEY (prospeccion, nombre_mineral),
    CONSTRAINT fk_hal_pro
     FOREIGN KEY (prospeccion)
```

```
      REFERENCES Prospecciones(prospeccion)
      ON DELETE CASCADE,
    CONSTRAINT fk_hal_min
      FOREIGN KEY (nombre_mineral)
      REFERENCES Minerales(nombre_mineral)
      ON DELETE CASCADE );
CREATE TABLE Explotaciones  (
   nombre_factoria VARCHAR2(15) NOT NULL,
   nombre_mineral VARCHAR2(15) NOT NULL,
    CONSTRAINT pk_Explotaciones
      PRIMARY KEY (nombre_factoria, nombre_mineral),
    CONSTRAINT fk_exp_fac
      FOREIGN KEY (nombre_factoria)
      REFERENCES Factorias(nombre_factoria)
      ON DELETE CASCADE,
    CONSTRAINT fk_exp_min
      FOREIGN KEY (nombre_mineral)
      REFERENCES Minerales(nombre_mineral)
      ON DELETE CASCADE );
CREATE TABLE Silos  (
   nombre_factoria VARCHAR2(15) NOT NULL,
   nombre_mineral VARCHAR2(15) NOT NULL,
   ctd_almacen NUMBER(5),
    CONSTRAINT pk_Silos
      PRIMARY KEY (nombre_factoria, nombre_mineral),
    CONSTRAINT fk_alm_fac
      FOREIGN KEY (nombre_factoria)
      REFERENCES Factorias(nombre_factoria)
      ON DELETE CASCADE,
    CONSTRAINT fk_alm_min
      FOREIGN KEY (nombre_mineral)
      REFERENCES Minerales(nombre_mineral)
      ON DELETE CASCADE,
    CONSTRAINT ck_cta
      CHECK (ctd_almacen > = 0) );
CREATE TABLE Producciones  (
   nombre_factoria VARCHAR2(15) NOT NULL,
   nombre_mineral VARCHAR2(15) NOT NULL,
   fecha_produccion DATE NOT NULL,
   ctd_produccion NUMBER(5),
   rdto_produccion NUMBER(2,2),
    CONSTRAINT pk_Producciones
      PRIMARY KEY (nombre_factoria, nombre_mineral,
                   fecha_produccion),
    CONSTRAINT fk_prod_alm
      FOREIGN KEY (nombre_factoria, nombre_mineral)
      REFERENCES Silos(nombre_factoria, nombre_mineral)
      ON DELETE CASCADE,
    CONSTRAINT ck_cdp
```



```
      CHECK (ctd_produccion > 0) );
CREATE TABLE Naves  (
   nave NUMBER(4) NOT NULL,
   od_nave LONG,
    CONSTRAINT pk_Naves
      PRIMARY KEY (nave) );
CREATE TABLE Viajes  (
   vuelo NUMBER(4) NOT NULL,
   fecha_ida DATE NOT NULL,
   nave NUMBER(4) NOT NULL,
   fecha_vuelta DATE,
   od_vuelo LONG,
   sr_viaje VARCHAR2(1),
    CONSTRAINT pk_Viajes
      PRIMARY KEY (vuelo, fecha_ida),
    CONSTRAINT fk_via_nav
      FOREIGN KEY (nave)
      REFERENCES Naves(nave)
      ON DELETE CASCADE,
    CONSTRAINT ck_sr
      CHECK (sr_viaje IN ('S', 'R')),
    CONSTRAINT ck_fec
      CHECK (fecha_vuelta > fecha_ida) );
CREATE TABLE Cargamentos  (
   nombre_factoria VARCHAR2(15) NOT NULL,
   nombre_mineral VARCHAR2(15) NOT NULL,
   num_carga NUMBER(5) NOT NULL,
   fecha_carga DATE,
   ctd_carga NUMBER(4),
   vuelo NUMBER(4),
   fecha_ida DATE,
   fecha_reco DATE,
    CONSTRAINT pk_Cargamentos
      PRIMARY KEY (nombre_factoria, nombre_mineral, num_carga),
    CONSTRAINT fk_car_alm
      FOREIGN KEY (nombre_factoria, nombre_mineral)
      REFERENCES Silos(nombre_factoria,nombre_mineral)
      ON DELETE CASCADE,
    CONSTRAINT fk_car_via
      FOREIGN KEY (vuelo, fecha_ida)
      REFERENCES Viajes(vuelo, fecha_ida),
    CONSTRAINT ck_car
      CHECK (ctd_carga > 0) );
CREATE TABLE Comidas  (
   alimento NUMBER(5) NOT NULL,
   nombre_alimento VARCHAR(20) NOT NULL,
   ctd_app NUMBER(4),
   coste_alimento NUMBER(3),
```

```
    CONSTRAINT pk_Comidas
      PRIMARY KEY (alimento),
    CONSTRAINT ck_app
      CHECK (ctd_app > = 0),
    CONSTRAINT ck_cal
      CHECK (coste_alimento > = 0) );
CREATE TABLE Provisiones  (
   nombre_factoria VARCHAR2(15) NOT NULL,
   num_provision NUMBER(5) NOT NULL,
   peso_provision NUMBER(4),
   fecha_llegada DATE,
   vuelo NUMBER(4),
   fecha_ida DATE,
    CONSTRAINT pk_Provisiones
      PRIMARY KEY (nombre_factoria, num_provision),
    CONSTRAINT fk_pro_fac
      FOREIGN KEY (nombre_factoria)
      REFERENCES Factorias(nombre_factoria)
      ON DELETE CASCADE,
    CONSTRAINT fk_pro_via
      FOREIGN KEY (vuelo, fecha_ida)
      REFERENCES Viajes(vuelo, fecha_ida)
      ON DELETE CASCADE,
    CONSTRAINT ck_ppr
      CHECK (peso_provision > 0),
    CONSTRAINT ck_flle
      CHECK (fecha_llegada > fecha_ida) );
CREATE TABLE Viveres  (
   nombre_factoria VARCHAR2(15) NOT NULL,
   num_provision NUMBER(5) NOT NULL,
   alimento NUMBER(5) NOT NULL,
   ctd_alimento NUMBER(4),
    CONSTRAINT pk_Viveres
      PRIMARY KEY (nombre_factoria, num_provision, alimento),
    CONSTRAINT fk_viv_prov
      FOREIGN KEY (nombre_factoria, num_provision)
      REFERENCES Provisiones(nombre_factoria, num_provision)
      ON DELETE CASCADE,
    CONSTRAINT fk_viv_ali
      FOREIGN KEY (alimento)
      REFERENCES Comidas(alimento)
      ON DELETE CASCADE,
    CONSTRAINT ck_ato
      CHECK (ctd_alimento > 0) );
CREATE TABLE Categorias  (
   trabajo NUMBER(2) NOT NULL,
   categoria VARCHAR2(20),
   sueldo NUMBER(7),
    CONSTRAINT pk_Categorias
```



```
      PRIMARY KEY (trabajo),
    CONSTRAINT ck_sue
      CHECK (sueldo > 0) );
CREATE TABLE Personal  (
   id_personal VARCHAR2(12) NOT NULL,
   nombrec_personal VARCHAR2(30),
   od_personal LONG,
    CONSTRAINT pk_Personal
      PRIMARY KEY (id_personal) );
CREATE TABLE Empleos  (
   id_personal VARCHAR2(12) NOT NULL,
   fecha_alta DATE NOT NULL,
   trabajo NUMBER(2) NOT NULL,
   od_empleo LONG,
    CONSTRAINT pk_Em
      PRIMARY KEY (id_personal, fecha_alta),
    CONSTRAINT fk_emp_per
      FOREIGN KEY (id_personal)
      REFERENCES Personal(id_personal)
      ON DELETE CASCADE,
    CONSTRAINT fk_emp_cat
      FOREIGN KEY (trabajo)
      REFERENCES Categorias(trabajo)
      ON DELETE CASCADE );
CREATE TABLE Destinados  (
   id_personal VARCHAR2(12) NOT NULL,
   fecha_alta DATE NOT NULL,
   nombre_factoria VARCHAR2(15) NOT NULL,
    CONSTRAINT pk_dts
      PRIMARY KEY (id_personal, fecha_alta),
    CONSTRAINT fk_des_fac
      FOREIGN KEY (nombre_factoria)
      REFERENCES Factorias(nombre_factoria)
      ON DELETE CASCADE,
    CONSTRAINT fk_des_emp_fue
      FOREIGN KEY (id_personal, fecha_alta)
      REFERENCES Empleos(id_personal, fecha_alta)
      ON DELETE CASCADE );
CREATE TABLE Tripulantes  (
   id_personal VARCHAR2(12) NOT NULL,
   fecha_alta DATE NOT NULL,
   vuelo NUMBER(4) NOT NULL,
   fecha_ida DATE NOT NULL,
    CONSTRAINT pk_Tripulantes
      PRIMARY KEY (id_personal, fecha_alta, vuelo, fecha_ida),
    CONSTRAINT fk_tri_emp
      FOREIGN KEY (id_personal, fecha_alta)
      REFERENCES Empleos(id_personal, fecha_alta)
```

```
    ON DELETE CASCADE,
  CONSTRAINT fk_tri_vue
    FOREIGN KEY (vuelo, fecha_ida)
    REFERENCES Viajes(vuelo, fecha_ida)
    ON DELETE CASCADE );
```

15.4.2. Manipulación de la Base de Datos

Cap15ej01.sql

Obtener todas las fábricas existentes en astros donde se ha encontrado Plata.

De la tabla Hallazgos se obtiene el atributo prospeccion de todos los hallazgos que se han producido en los cuales su atributo nombre_mineral tiene el valor 'Plata'. El resultado de esta subconsulta se utiliza para obtener los atributos nombre_lugar y nombre_astro de todas las tuplas de la tabla Prospecciones que cumplan la condición de que el número de prospección sea el devuelto anteriormente. La información obtenida en esta subconsulta se emplea para obtener de la tabla Factorias el atributo nombre_factoria de todas las tuplas cuyos campos nombre_lugar y nombre_astro son los generados en la subconsulta anterior.

```
SELECT nombre_factoria "Factorías"
   FROM Factorias
   WHERE (nombre_lugar, nombre_astro) IN
     (SELECT nombre_lugar, nombre_astro
      FROM Prospecciones
      WHERE prospeccion IN
        (SELECT prospeccion
           FROM Hallazgos
           WHERE nombre_mineral = 'Plata'));
```

Resultado

```
Factorías
---------------
Factoria1
Factoria2
Factoria3
```

Cap15ej02.sql

Obtener todos los satélites de aquellos planetas donde existen prospecciones.

De la tabla Satelites se obtendrán los atributos nombre_satelite y nombre_planeta de todas aquellas tuplas que cumplen la condición de que existen tuplas en la tabla Prospecciones y que cumplen la condición de que el atributo nombre_astro es igual al atributo nombre_planeta de la tabla Satelites. Es decir, se garantiza que existen prospecciones en un planeta, y una vez cumplida esta condición, se obtienen todos sus satélites.

```
SELECT nombre_satelite, nombre_planeta
   FROM Satelites
   WHERE EXISTS
     (SELECT nombre_astro
      FROM Prospecciones
      WHERE nombre_astro = Satelites.nombre_planeta);
```


Resultado

```
NOMBRE_SATELITE NOMBRE_PLANETA
--------------- ---------------
Luna            Tierra
Metis           Júpiter
Europa          Júpiter
Lisitea         Júpiter
Atlas           Saturno
Pandora         Saturno
Mimas           Saturno
Ofelia          Urano
Julieta         Urano
Belinda         Urano
```

Cap15ej03.sql

Obtener el número de meses que transcurrieron desde que el vuelo 2111 partió de la tierra hasta que regresó.

En esta consulta se aplica la función MONTHS_BETWEEN a los atributos fecha_vuelta y fecha_ida de todas las tuplas de la tabla Viajes que cumplen la condición de que la nave es la 2378, obteniéndose todos los intervalos de días de los viajes del vuelo referenciado.

```
SELECT MONTHS_BETWEEN(fecha_vuelta, fecha_ida)
       "Meses de vuelo", TO_CHAR(fecha_ida,'DD/MM/YYYY')
       "Fecha_ida"
   FROM Viajes
   WHERE vuelo = 2111;
```

Resultado

```
Meses de vuelo Fecha_ida
-------------- ----------
             2 01/01/2067
```

Cap15ej04.sql

Obtener el nombre de los alimentos trasladados a la factoría 'Factoria1' el 01/01/2057.

De la tabla Provisiones se obtiene el atributo num_provision de todas las tuplas que cumplen la condición de que nombre_factoria tiene el valor 'Factoria1' y que fecha_ida es el '01/01/2057'. El resultado de esta subconsulta se utiliza para obtener el atributo alimento de todas las tuplas de la tabla Viveres que cumplen la condición de que num_provision tiene el valor devuelto por la consulta anterior. Por último, de la tabla Comidas se obtiene el atributo nombre_alimento de todas aquellas tuplas en las que el atributo alimento tiene el valor devuelto por la subconsulta anterior.

```
SELECT nombre_alimento
   FROM Comidas
   WHERE alimento IN
      (SELECT alimento
       FROM Viveres
       WHERE num_provision IN
         (SELECT num_provision
            FROM Provisiones
            WHERE nombre_factoria = 'Factoria1' AND
              fecha_ida = TO_DATE('01/01/2057','DD/MM/YYYY')));
```

Resultado

```
NOMBRE_ALIMENTO
--------------------
Aceite
Carne
Legumbres
Patatas
Bebidas
```

Cap15ej05.sql

Obtener la cantidad de mineral de Oro almacenado en Marte.

De la tabla Factorias se obtiene el atributo nombre_factoria de todas las tuplas que cumplen la condición de que nombre_astro tiene el valor 'Marte'. El resultado de esta subconsulta se utiliza para obtener el sumatorio del valor del atributo ctd_almacen de todas las tuplas de la tabla Silos que cumplan la condición de que el atributo nombre_mineral sea 'Oro'.

```
SELECT SUM (ctd_almacen) "Existencias Oro"
   FROM Silos
   WHERE nombre_mineral = 'Oro' AND nombre_factoria IN
     (SELECT nombre_factoria
        FROM Factorias
        WHERE nombre_astro = 'Marte');
```

Resultado

```
Existencias Oro
---------------
          31111
```

Cap15ej06.sql

Obtener el nombre e identificador del personal destinado a Marte después del 2011.

De la tabla Factorias se obtiene el atributo nombre_factoria de todas las factorías que cumplen la condición de que nombre_astro tenga el valor 'Marte'. El resultado de esta subconsulta se utiliza para obtener de la tabla Destinados el valor del atributo id_personal de todos los trabajadores destinados a una factoría en Marte cuyo atributo fecha_alta sea posterior al 31 de diciembre del 2011, y el resultado de esta subconsulta se utiliza para obtener de la tabla Personal el atributo nombrec_personal de todos estos trabajadores, así como su id_personal.

```
SELECT nombrec_personal, id_personal
   FROM Personal
   WHERE id_personal IN
     (SELECT id_personal
      FROM Destinados
      WHERE fecha_alta > TO_DATE('31/12/2011','DD/MM/YYYY')
            AND nombre_factoria IN
       (SELECT nombre_factoria
          FROM Factorias
          WHERE nombre_astro = 'Marte') );
```


Resultado

```
NOMBREC_PERSONAL                ID_PERSONAL
------------------------------- -----------
Pedro González González         10101016
Javier Muñoz Muñoz              10101020
Antonio Navarro Lozano          10101021
```

Cap15ej07.sql

Obtener el número de factorías existentes en cada astro.

De la tabla Factorias se obtiene el atributo nombre_astro y se contabiliza el número de apariciones de un valor distinto del atributo nombre_factoria para un mismo valor de nombre_astro. Para que esta consulta pueda realizarse, es necesario que el resultado se agrupe por el atributo nombre_astro.

```
SELECT nombre_astro "Astro", COUNT(nombre_factoria) "Factorías"
   FROM Factorias
   GROUP BY nombre_astro
   ORDER BY COUNT(nombre_factoria);
```

Resultado

```
Astro           Factorías
--------------- ---------
Marte                   1
Luna                    2
Tierra                  2
```

Cap15ej08.sql

Se desea consultar la estructura de la tabla anidada al campo centros de la tabla Minerales.

El mandato para ver la estructura de una tabla es: DESC nombre_tabla. Pero este mandato no solamente sirve para ver la estructura de una tabla sino que también sirve para ver la estructura de cualquier objeto, por tanto, la solución sería:

1. *Ver el tipo del campo Centros de la tabla Minerales, para ello se utiliza el mandato DESC con la tabla Minerales.*
2. *Utilizar el mandato DESC con el tipo del campo Centros.*

```
DESC Minerales
DESC Tipo_Centros
```

Resultado

```
Name                            Null?    Type
------------------------------- -------- ----
NOMBRE_MINERAL                  NOT NULL VARCHAR2(15)
OD_MINERAL                               LONG
CENTROS                                  TIPO_CENTROS
Tipo_Centros TABLE OF CENTRO
Name                            Null?    Type
------------------------------- -------- ----
NOMBRE_CENTRO                            VARCHAR2(15)
UBICACION_CENTRO                         VARCHAR2(20)
OD_CENTRO                                VARCHAR2(2000)
```

Cap15ej09.sql

Se ha descubierto en la tierra Oro. Por tanto, se va a abrir un nuevo centro para su extracción, denominado CenPlaTierra. Insertar dicho Centro en la tabla Minerales.

Para realizar esta inserción no se puede utilizar el formato usual del mandato INSERT, sino que se debe usar un formato especial en el cual se identifica al objeto (tabla) enlazado como una subconsulta precedida por las palabras reservadas THE o TABLE. De esta forma se obtiene una referencia a dicha tabla enlazada, pudiéndose insertar tuplas a través de la palabra reservada VALUES. Se ha de indicar que la subconsulta solamente puede devolver una fila, la cual referencia a la tabla en la que se desea insertar.

```
INSERT INTO TABLE
   (SELECT Centros
    FROM Minerales
    WHERE UPPER(nombre_mineral)=UPPER('oro'))
   VALUES ('CenPlaTierra','Tierra','');
```

Resultado

```
1 fila creada.
```

Cap15ej10.sql

Se ha acabado el mineral denominado Oro en el planeta Júpiter. Por tanto, se van a cerrar los centros en los cuáles se extraía dicho mineral en ese planeta. Borrar todos los centros de Júpiter donde se extraía Oro.

Para realizar el borrado de los centros de Júpiter donde se extrae Oro es preciso extraer la referencia a la tabla anidada utilizada para almacenar los centros donde se obtiene dicho mineral. Para ello se vuelve a utilizar la cláusula TABLE acompañada de una subconsulta en la cual se selecciona el campo Centros para el mineral Oro. De esta forma se cuenta con una referencia a dicha tabla la cual será utilizada en la cláusula FROM del mandato DELETE.

```
DELETE FROM TABLE
   (SELECT Centros
    FROM Minerales
    WHERE UPPER(nombre_mineral)=UPPER('Oro'))
   WHERE UPPER(ubicacion_centro)=UPPER('Júpiter');
```

Resultado

```
1 fila borrada.
```

Cap15ejP1.sql

Cree un procedimiento PL/SQL que para un determinado mineral (parámetro de entrada del procedimiento), muestre todos los centros que lo procesan.

Se crea un cursor "centro_mineral" para almacenar los atributos nombre_centro y ubicacion_centro de la tabla enlazada minerales —se utiliza el formato especial "TABLE(SELECT()" para referenciar este tipo de objeto—, posteriormente se utiliza una variable de tipo ROWTYPE y un bucle FOR para mostrar la información.

```
/* Este procedimiento presenta una muestra de la
manipulación de las tablas anidadas construidas haciendo uso
de los tipos de objeto, los cuales son manipulados por
Oracle bajo el modelo objeto-relacional */
```



```
CREATE OR REPLACE PROCEDURE lista_centro
   (n_mineral VARCHAR2)
AS
-- Se definen las variables de trabajo y cursores
   centro VARCHAR2(12);
   CURSOR centro_mineral IS
      SELECT C.nombre_centro, C.ubicacion_centro
       FROM Minerales M,
        TABLE(SELECT Centros
              FROM Minerales
              WHERE UPPER(Minerales.nombre_mineral) =
                    UPPER(n_mineral))C
       WHERE UPPER(M.nombre_mineral) = UPPER(n_mineral);
BEGIN
   DBMS_OUTPUT.ENABLE(100000);
   DBMS_OUTPUT.PUT_LINE ('Centros que procesan el mineral:
   '||n_mineral);
   DBMS_OUTPUT.PUT_LINE ('Centro'||'    '||'Lugar');
   FOR fila_del_cursor IN centro_mineral LOOP
      DBMS_OUTPUT.PUT_LINE (fila_del_cursor.nombre_centro||'
      '||fila_del_cursor.ubicacion_centro);
   END LOOP;
COMMIT;
-- Se comprueban los posibles errores del proceso
EXCEPTION
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

Prueba de ejecución

```
-- Se ejecuta el procedimiento
SET SERVEROUTPUT ON
    EXEC lista_centro ('Platino');
SET SERVEROUTPUT OFF
```

CAPÍTULO 16

ORACLE EN INTERNET

Los cambios tecnológicos experimentados en los últimos años, paralelamente con las nuevas necesidades de las empresas para mantener su competitividad en los mercados, han dado lugar a que surjan nuevas exigencias para las bases de datos y, por lo tanto, para los *SGBD* sobre los cuales están implementadas.

Ya no es suficiente, para la mayoría de los sistemas software que hacen uso de las bases de datos el que éstas satisfagan ese conjunto de características o propiedades que se describieron en el Capítulo 1. Los *SGBD* actuales además de permitir el mantenimiento y gestión de bases de datos con un alto desempeño, de forma privada, segura y fiable deben satisfacer el que puedan ser *accesibles*.

El término *accesible* debe entenderse en el sentido más amplio de su acepción. Los usuarios, de cualquier tipo, requieren además de poder acceder a las bases de datos mediante conexiones locales o remotas a la máquina servidora en la cual se encuentra instalada, de un acceso a través de Internet.

Internet ha transformado en los últimos años el uso, manipulación y acceso a la información. La divulgación de la información a través de la red y su importancia dentro de los procesos de negocio de las organizaciones, ha dado lugar a que se exija a los *SGBD* el que puedan dar un soporte efectivo a estas nuevas necesidades.

En este capítulo se presenta una introducción al lector del uso de las bases de datos en Internet, y cómo *Oracle* da soporte a esta integración entre las bases de datos e Internet a través de algunos de sus componentes.

El contenido del capítulo se complementa con un ejemplo, un caso práctico, el cual se resuelve, al igual que en los capítulos anteriores, desde su conceptualización hasta su diseño, solución e implementación bajo *Oracle*, y que consiste en el mantenimiento de todas y cada una de las bases de datos desarrolladas en esta obra, de los ejercicios prácticos resueltos y de un conjunto de ejercicios adicionales.

En este caso, en lugar de presentar una serie de procedimientos del uso y manipulación de la base de datos haciendo uso de *SQL* y *PL/SQL*, se incluye un ejemplo del acceso a la base de datos desde un navegador estándar (*Internet Explorer*, *Netscape*), el cual se incluye en el CD que acompaña la obra.

Este procedimiento desarrollado permite al lector consultar el conjunto de ejercicios prácticos incluidos en la obra (y otros adicionales) desde una página Web, ejecutarlos y visualizar el resultado de los mismos.

Como para poder ejecutar el procedimiento propuesto es necesario que el lector instale en su ordenador la base de datos de los ejercicios, se incluye en este capítulo, una guía rápida de instalación de *Oracle8i* y del *WebDB*.

16.1. LAS BASES DE DATOS Y LA RED

El acceso a la base de datos a través de Internet supone, entre otras muchas cosas, la comunicación entre:

- El *servidor de la base de datos*, el cual "*escucha*" y atiende todas las peticiones que el usuario realiza para acceder a la información de la base de datos.
- El *servidor Web*, el cual se encarga de la comunicación remota con el usuario.
- El *proceso cliente*, consistente en un conjunto de sentencias cuyo objetivo es acceder a la información almacenada en la base de datos.

El servidor Web es el encargado de comunicarse con el proceso cliente aportándole un soporte de red. Este proceso recibe las peticiones del cliente, y cuando éstas requieren del acceso a la base de datos, el servidor Web se comunica con el servidor de bases de datos transfiriendo las peticiones del cliente.

El servidor de bases de datos recoge las peticiones del cliente, a través del servidor Web, y resuelve las mismas. El resultado de las peticiones del usuario es devuelto al servidor Web, el cual se comunica de nuevo con el cliente, devolviéndole el resultado de sus peticiones a la base de datos que ha sido aportado por el servidor de bases de datos.

Bajo este modelo, cada servicio o proceso (servidor de bases de datos, servidor Web y cliente) puede ser ejecutado en máquinas diferentes (o no) descentralizando el consumo de recursos computacionales, por lo que aumenta el desempeño y las necesidades de los mismos, e independiza las plataformas en las cuales estos procesos se ejecutan.

Como puede observar el lector, existen dos tipos de comunicaciones bajo este modelo en tres capas:

1. Entre el servidor Web y el servidor de bases de datos
2. Entre el cliente (usuario) y el servidor Web.

La comunicación con el servidor Web debe realizarse a través de procedimientos desarrollados con lenguajes estándar en Internet, como por ejemplo *Java* o *HTML*, y la comunicación con el servidor de bases de datos debe realizarse a través de un lenguaje que reconozca el mismo, como por ejemplo *SQL* o *PL/SQL*.

De esta forma, el proceso cliente, el cual accede a la base de datos a través de Internet, incorpora código fuente escrito tanto en un lenguaje de Internet (*Java*), como en un leguaje de bases de datos (*SQL*), puesto que este proceso debe comunicarse con el servidor Web y, entre otras acciones, va a acceder a la base de datos mantenida en el servidor de bases de datos.

16.1.1. Acceso a bases de datos Oracle con Java

Java es un lenguaje orientado a objetos que por su potencia, independencia de la plataforma y simplicidad se ha convertido en un estándar en Internet. Bien a través de *applets* o incrustando sentencias *JavaScript* en páginas *HTML*, se pueden desarrollar aplicaciones Web muy potentes.

Para acceder a la base de datos a través de procedimientos desarrollados en *Java*, se pueden utilizar dos técnicas:

Conectividad de base de datos de Java (**JDBC**): es un estándar industrial que aporta una interfaz de programación de aplicaciones independiente de la base de datos y que permite el desarrollo de aplicaciones que manipulen las mismas. Cada fabricante de bases de datos debe implementar un controlador de *JDBC* que permite el acceso a la base de datos desde programas *Java*. Este controlador, basado en el estándar, es dependiente del *SGBD* por lo que el programa *Java* que incorpora *JDBC* también lo es. El acceso a la base de datos se realiza haciendo uso de las clases incluidas en el paquete *java.sql.** (incluido en el **JDK**, *Kit de Desarrollo de Java*).

La principal característica de *JDBC* además de la independencia de la base de datos, es que permite el uso de *SQL* dinámico; es decir, que se generen o construyan sentencias *SQL* en tiempo de ejecución. Esto es debido a que *Java* no realiza comprobaciones sintácticas, ni semánticas, de las sentencias *SQL* que realizan llamadas al *JDBC*. Esta ventaja puede ser un inconveniente, ya que los errores (de acceso a la base de datos) serán descubiertos en tiempo de ejecución y no en tiempo de compilación de las clases *Java*.

SQL-Java (**SQLJ**): es un intérprete que traduce las sentencias *SQL*, y otras sentencias como llamadas a procedimientos y funciones almacenadas en la base de datos,

incrustadas en los programas *Java* a sentencias *JDBC* antes de generar el archivo de clases *Java* asociado al programa.

Las sentencias son incrustadas en el programa fuente de la forma clásica al uso de lenguajes huésped para el acceso a la base de datos. Cada sentencia tiene una cabecera (por ejemplo: *#sql*) que informa al preprocesador de su naturaleza y, por tanto, que debe ser preprocesada.

Además de estas dos técnicas de acceso a bases de datos, *Oracle* también incluye una máquina virtual *Java* (*JVM*), lo que permite cargar y ejecutar clases dentro del propio servidor de bases de datos. La *JVM* de *Oracle* aporta una gran potencia al servidor de bases de datos puesto que los procedimientos *Java* pueden ser almacenados en el propio núcleo de la base de datos y cargar y ejecutar funciones y procedimientos *Java*.

16.1.2. Desarrollo de aplicaciones para Internet con Oracle

Además de permitir y de garantizar el acceso a las bases de datos a través de Internet de forma fiable y eficiente, *Oracle* cuenta con una serie de productos orientados al desarrollo de aplicaciones para Internet.

Productos como *Oracle®Designer* y *Oracle®Developer*, ampliamente conocidos en el diseño y desarrollo de aplicaciones cliente-servidor, han sido mejorados en las últimas versiones incorporando componentes que contemplan el uso de la base de datos a través de Internet.

Productos de *Oracle* como *Oracle Application Server* (**OAS**), *Oracle Web Application* (**OWA**), *Internet Application Server* (**IAS**), permiten una conectividad total de los usuarios, a través de Internet, a bases de datos *Oracle* y contenidos Web.

Pero hoy en día, las necesidades de negocio requieren que las organizaciones utilicen la red para algo más que acceder a la información almacenada en la base de datos. La red es un medio idóneo de difusión de información tanto al exterior de la organización, como al interior o los miembros de la misma. Por ello, las organizaciones desarrollan sus "*portales*" a través de los cuales la información es difundida; información que debe ser mantenida y que en muchos casos forma parte de la base de datos utilizada en las operaciones de negocio de estas organizaciones, lo que requiere una integración entre la tecnología Internet y la de bases de datos.

Estas nuevas necesidades del negocio suponen un esfuerzo añadido de gestión y administración tanto de la información, como del software o herramientas utilizadas para su tratamiento. *Oracle* cuenta con un producto denominado *WebDB* (*Oracle Portal* a partir de la versión 3.0) que intenta abordar una solución global a la gestión y uso de las bases de datos a través de Internet, y de la administración de los portales o sitios Web.

WebDB permite crear una taxonomía para la clasificación y la organización del contenido de un sitio Web, teniendo una estructura incorporada para organizar, clasificar y realizar las referencias cruzadas de los elementos del mismo. Mientras que en los sitios Web clásicos el componente más pequeño es una página, con *WebDB* es un elemento (componente), generándose las páginas de forma dinámica a partir de los componentes.

WebDB organiza la información en carpetas, similares a las del sistema de archivo, que contienen elementos clasificados en categorías y que tienen asociados atributos. Cada elemento puede clasificarse en más de una categoría a través de la definición de perspectivas, las cuales pueden ser invocadas en tiempo de ejecución por páginas dinámicas. *Oracle WebDB* es una aplicación basada en navegadores que permite desarrollar y usar portales Web. Tanto los usuarios finales, como los administradores del portal, en forma conjunta obtienen, publican y gestionan la información que se presenta en el portal a través de un navegador estándar.

En este capítulo se incluye una sección destinada al proceso de instalación de *Oracle WebDB*, el cual ha sido utilizado (evidentemente, sólo en una mínima parte) para el desarrollo de la aplicación que manipula a través de una página Web el contenido de las bases de datos de los problemas resueltos en este texto.

16.2. EJERCICIOS DE MANIPULACIÓN DE BASES DE DATOS

Para incluir un ejemplo de la integración y uso de las bases de datos en un entorno Web se ha propuesto el desarrollo de una base de datos que mantenga una colección de ejercicios de manipulación de las bases de datos correspondientes a los ejemplos incluidos en este libro.

De esta forma, se plantea el problema de representar la información correspondiente a un conjunto de ejercicios consistentes en un enunciado, una propuesta de solución y un conjunto de sentencias *SQL* o *PL/SQL* correspondientes a un conjunto de ejemplos, diez hasta el momento, que representan la solución a diferentes problemas del mundo real.

Este problema se puede especificar detalladamente mediante los siguientes supuestos:

SUPUESTO 1: Se desea mantener información de los ejercicios correspondientes a un conjunto de bases de datos (ejemplos) diferentes. Actualmente, se cuenta con diez bases de datos sobre las que se tiene una colección de ejercicios.

SUPUESTO 2: Los ejercicios para cada una de las bases de datos o ejemplos se desean clasificar en diferentes categorías: los que están incluidos en el texto del libro, y ejercicios adicionales no incluidos en el texto.

SUPUESTO 3: Cada ejercicio viene representado por la siguiente información textual: una descripción o enunciado del ejercicio, una descripción o comentario del proceso de solución, y un conjunto de sentencias realizadas en *SQL* o *PL/SQL* consistentes en la solución al ejercicio propuesto.

16.2.1. Modelo Conceptual

La simplicidad del problema nos permite que con estos supuestos puedan ser extraídos los elementos conceptuales que caracterizan el mismo.

Tipo de entidad *Problema*: representando a "*las diferentes bases de datos o problemas sobre los cuales se desea mantener la información correspondiente a ejercicios de uso y manipulación del mismo*" (SUPUESTO 1).

Para este tipo de entidad se van a considerar los atributos: *capitulo*, como identificador del tipo de entidad y representando el capítulo del libro que trata sobre un problema y para el cual se ha construido una base de datos, y *descripcion* para representar el nombre del problema o la base de datos para la cual se desea representar una colección de ejercicios.

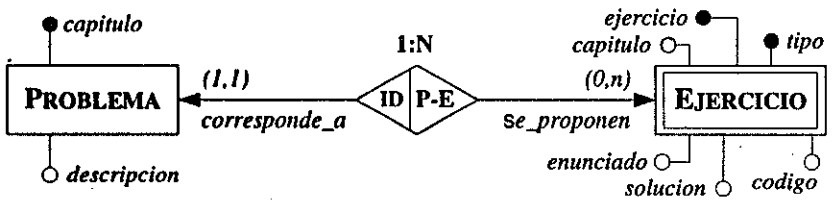


Figura 16.1 Diagrama Entidad-Interrelación del problema propuesto

Tipo de entidad *Ejercicio*: representando a "*los diferentes ejercicios que se desean mantener sobre cada uno de los problemas o bases de datos diferentes*" (SUPUESTOS 2 y 3).

Como los ejercicios deben su existencia a los problemas sobre los cuales se desarrollan, se trata de un tipo de entidad débil por identificación con respecto al tipo de entidad *Problema*. Para este tipo de entidad se van a considerar los atributos: ***ejercicio***, representando a cada uno de los ejercicios; ***tipo***, representando el tipo o categoría del ejercicio; ***enunciado***, para representar el enunciado o propuesta del ejercicio; ***solucion***, representando a la descripción de la solución desarrollada para el ejercicio, y ***codigo***, representando el código *SQL* o ***PL/SQL*** que forma parte de la solución desarrollada para cada ejercicio (SUPUESTO 3).

El identificador de este tipo de entidad estará formado por la agregación de los atributos ***capitulo*** (heredado del tipo de entidad *Problema* por la debilidad de identificación), ***ejercicio*** y ***tipo***, lo que satisface los requisitos impuestos al problema.

Los ejercicios pueden ser de dos categorías diferentes: los incluidos en el texto, y los adicionales no incluidos en el texto pero sí incluidos en el material de este libro. Por ello, el atributo *tipo* sólo podrá tomar dos valores "T" o "A", para representar las dos categorías de ejercicios permitidas.

Estos dos tipos de entidades están relacionados a través de un tipo de interrelación que representa la relación existente entre cada uno de los ejercicios y la base de datos o ejemplo para el cual se han desarrollado. Este tipo de interrelación se puede definir de la forma:

Tipo de interrelación *Problema/Ejercicio* **(P-E)**: el cual relaciona los tipos de entidad *Problema* y *Ejercicio*. Es evidente que cada ejercicio se desarrolla para uno y sólo un problema puesto que hace uso de los elementos de la base de datos desarrollada para ese problema, y que para cada base de datos o problema se pueden desarrollar cero o muchos ejercicios. Por tanto se trata de un tipo de interrelación uno a muchos que representa la debilidad por identificación entre ambos tipos de entidad.

La figura 16.1 muestra el esquema conceptual que describe la solución del problema, apreciándose en el mismo cada uno de los objetos descritos.

16.2.2. Modelo Relacional

El modelo conceptual propuesto para este problema y representado en la figura 16.1 puede ser fácilmente derivado utilizando las reglas de transformación propuestas en el Capítulo 5 para construir el siguiente esquema relacional.

Problema *(capitulo, descripcion)*

Ejercicio *(**capitulo**, ejercicio, tipo, enunciado, solucion, codigo)*

Como se puede apreciar, la tabla *Problema* se deriva por aplicación de la regla *RTECAR-1* al tipo de entidad del mismo nombre, y la tabla *Ejercicio*, se obtiene por aplicación de la regla *RTECAR-1* al tipo de entidad del mismo nombre y la regla *RTECAR-3.1* al tipo de interrelación (*P-E*).

Ambas tablas se encuentran normalizadas, por lo que se puede definir sintácticamente el esquema relacional de la forma siguiente:

```
   /* Inicialmente, se borran las tablas */
DROP TABLE Ejercicio;
DROP TABLE Problema;
   /* Se crean las tablas que forman parte del esquema */
CREATE TABLE Problema (
   capitulo NUMBER(2) NOT NULL,
   descripcion VARCHAR2(30),
    CONSTRAINT pk_pro
      PRIMARY KEY (capitulo) );
```

```
CREATE TABLE Ejercicio (
   capitulo NUMBER(2) NOT NULL,
   ejercicio CHAR(2) NOT NULL,
   tipo CHAR(1) NOT NULL,
   enunciado VARCHAR2(2000),
   solucion VARCHAR2(4000),
   codigo CLOB,
    CONSTRAINT pk_eje
     PRIMARY KEY (capitulo, ejercicio, tipo),
    CONSTRAINT fk_eje_pro
     FOREIGN KEY (capitulo)
     REFERENCES Problema(capitulo)
     ON DELETE CASCADE,
    CONSTRAINT ck_tip
     CHECK (tipo IN ('T', 'A')) );
```

En lugar de representar lógicamente el problema mediante un esquema relacional clásico, puede hacerse también mediante una representación objeto-relacional. Observe el lector, que los capítulos o diferentes problemas pueden verse como formados por la agregación de un conjunto de ejercicios, los cuales forman parte de las características de los problemas y, por tanto, pueden ser tratados como un atributo más. En este caso un atributo complejo, puesto que cada ejercicio está formado por un conjunto de atributos que lo caracterizan, y cada problema puede contener muchos ejercicios.

La representación del problema haciendo uso del paradigma orientado a objetos y su implementación en un esquema objeto-relacional, permite la construcción de una base de datos simple, en la que se hace uso de una única tabla (para representar los problemas), que incorpora una tabla anidada (para representar los ejercicios).

```
   /* Se borran los tipos de objeto y la tabla */
DROP TABLE Problema;
DROP TYPE Problema_Tipo;
DROP TYPE Ejercicio_Tipo;
DROP TYPE Tipo_Ejercicios;
   /* Se crean los tipos de objetos existentes */
CREATE TYPE Ejercicio_Tipo AS OBJECT (
   ejercicio CHAR(2),
   tipo CHAR(1),
   enunciado VARCHAR2(2000),
   solucion VARCHAR2(4000),
   codigo CLOB )
   /
CREATE TYPE Tipo_Ejercicios AS TABLE OF Ejercicio_Tipo
   /
CREATE OR REPLACE TYPE Problema_Tipo AS OBJECT (
   capitulo NUMBER(2),
   descripcion VARCHAR2(30),
   ejercicios Tipo_Ejercicios )
   /
   /* Se crea la tabla para los problemas, la cual contiene la
   tabla anidada para los ejercicios */
CREATE TABLE Problema OF Problema_Tipo (
   capitulo NOT NULL )
   NESTED TABLE Ejercicios
      STORE AS Ejercicios_Tabla (
      (PRIMARY KEY (NESTED_TABLE_ID, ejercicio, tipo) );

   /* Se modifica la definición de la tabla principal y la
   anidada para representar las restricciones */
ALTER TABLE Problema (
   ADD CONSTRAINT pk_prob
    PRIMARY KEY (capitulo);
ALTER TABLE Ejercicios_Tabla (
   ADD CONSTRAINT ck_tipo
    CHECK (tipo IN ('T', 'A')) );
```


Proponemos como práctica para el lector, el que implemente este esquema objeto-relacional y, como cuenta en el CD que acompaña la obra con los ficheros con el código fuente de los esquemas y ejercicios de los problemas resueltos, construya las correspondientes bases de datos y practique con la manipulación de las mismas bajo este nuevo esquema.

16.2.3. Manipulación de la Base de Datos

Como puede observar el lector, la base de datos que se describe en este capítulo, tiene como objetivo el de almacenar el conjunto de ejercicios *SQL* y *PL/SQL* descritos en la obra (y otros más) para cada uno de los problemas planteados en los Capítulos 6 al 15 y, generalizando, cualquier secuencia de mandatos que pueda ser interpretada por *Oracle*.

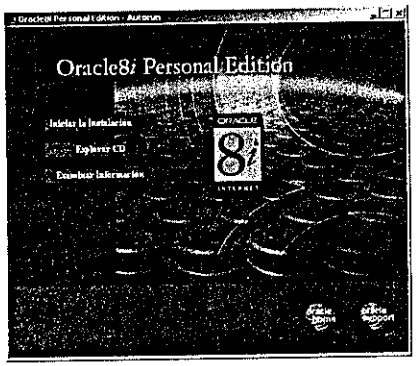
Así, en la base de datos diseñada se puede almacenar, tanto la definición de un esquema (por ejemplo, el correspondiente al Capítulo 6 —*El Catastro Municipal*—), como las sentencias de inserción, borrado, modificación y consulta que se desee sobre un determinado esquema.

Por ello, se ha utilizado este esquema para almacenar todos y cada uno de los esquemas descritos en la obra, así como los ejercicios correspondientes, para que el lector pueda acceder, modificar, y ejecutar todos y cada uno de los ejemplos descritos en el libro. Esta operación la puede realizar bien haciendo uso del *SQL*PLUS* de *Oracle*, o a través de una aplicación desarrollada en *WebDB* que se incluye en el CD que acompaña la obra.

Para ello, describiremos brevemente a continuación cómo realizar la instalación de los productos *Oracle Personal Edition* y *Oracle WebDB*.

16.3. INSTALACIÓN DE ORACLE PERSONAL EDITION

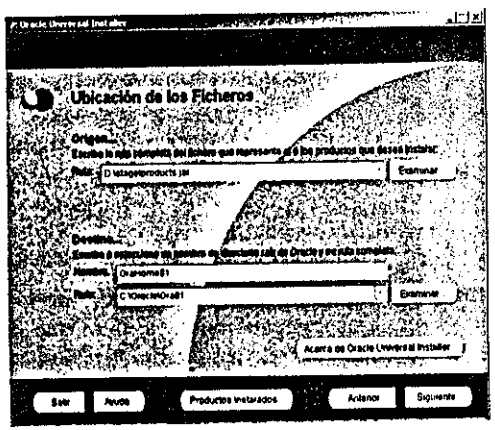
La instalación de *Oracle8i Personal Edition* se realiza a través de la sesión programada que se encuentra en el CD que acompaña al producto. Este CD cuenta con un auto-arranque que nos presenta una pantalla inicial a través de la cual el usuario puede:



- *Iniciar la instalación* de los productos proporcionados.
- *Explorar el CD,* recorriendo cada uno de los directorios en los cuales está organizado el CD.
- *Examinar Información* acerca de los productos *Oracle*.

La instalación del producto comienza mediante la elección de la opción *iniciar la instalación*, o bien *explorando* el CD, accediendo al directorio en el que se encuentra y ejecutando el fichero *setup.exe*.

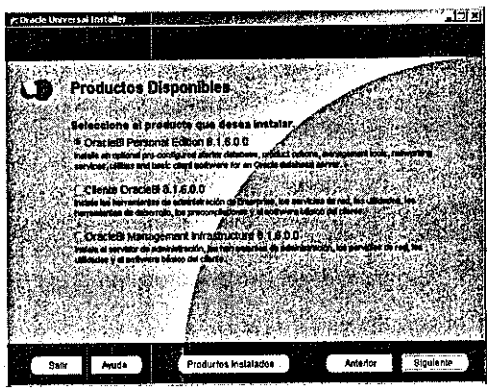
Inicialmente aparece una ventana en la cual se presentan las opciones de instalación típica bajo *Windows* de cualquier producto, a través de la que el usuario puede instalar, desinstalar y comprobar el estado de los productos instalados. En el proceso de instalación se van presentando ventanas sucesivas en las que el usuario debe completar algunos formularios requeridos por el proceso para los cuales *Oracle* proporciona unos valores por defecto.



La instalación comienza solicitando información para poder iniciar el proceso de copia de ficheros desde la unidad elegida como origen hasta la unidad elegida como destino. La parte encabezada por la etiqueta *Origen* se dejaría tal como la presenta el programa de instalación. La parte encabezada con la etiqueta *Destino* es un poco más delicada, ya que a través de ella se elige el lugar de almacenamiento de los ficheros que van a ser utilizados por *Oracle Personal Edition*. En ella se solicita la siguiente información:

Nombre: o *HOME* utilizado para almacenar *Oracle8i Personal Edition*. El *HOME* es un nombre simbólico que representa el lugar de almacenamiento de los ficheros necesitados por *Oracle8i Personal Edition*. De esta forma se pueden instalar diferentes versiones de un mismo producto *Oracle* sin que interfieran entre sí en su ejecución. Hay que tener cuidado con la asignación del *HOME*, ya que hay productos *Oracle* cuya instalación es incompatible. Éste sería el caso a tener en cuenta entre *Oracle Personal Edition* y *WebDB*, como se verá más adelante. Se aconseja al usuario dejar el valor que aparece por defecto.

Ruta: o directorio en el espacio de almacenamiento del usuario al cual equivale el *HOME* elegido anteriormente. Si el usuario proporciona un *HOME* diferente al proporcionado por *Oracle*, la *Ruta* aparece vacía, lo que significa que el usuario debe elegir un directorio existente, o bien un directorio que será creado por el programa de instalación.

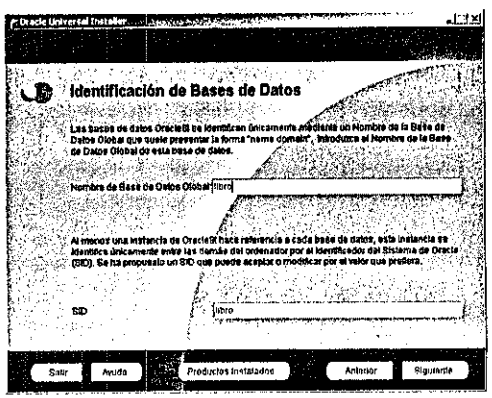


Una vez determinado el *HOME* y su ubicación, se procede a instalar el servidor de bases de datos *Oracle* el cual va a ser utilizado para el mantenimiento de la base de datos problema, y que va a ser utilizado por *WebDB*. Los otros productos pueden ser instalados posteriormente por el usuario si va a trabajar con *Oracle*:

Oracle Client 8.1.6.0.0. necesario para realizar la conexión al servidor *Oracle Personal Edition 8.1.6.0.0*, instalado en la máquina local o una máquina remota.

Oracle Management Infraestructure 8.1.6.0.0. necesario para realizar labores de administración (incluyendo el software de la parte cliente).

Una vez elegido el producto a instalar, se debe elegir el tipo de instalación (Típica, Mínima o Personalizada). Recomendamos al lector que elija el tipo por defecto (Typical) a no ser que ya haya instalado previamente el producto y sólo desee reinstalar parte del mismo o instalar nuevos componentes.



A continuación se solicita el nombre de la base de datos que va a ser utilizada por el usuario. En este formulario se deberá introducir la cadena **libro**, tanto para el *Nombre de la Base de Datos Global*, como para el *Identificador del Sistema de Oracle* (*SID*).

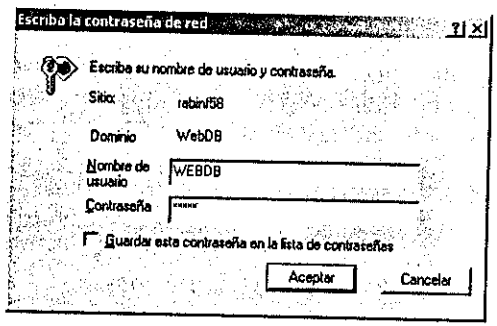
Tablespaces del usuario dueño de los procedimientos que se utilizarán para el desarrollo de aplicaciones *Web* con *WebDB*. Se recomienda dejar la información que el programa instalador proporciona como defecto (*OAS_PUBLIC, TOOLS, TEMP*).

Después de terminar la carga de los paquetes *WebDB-PL/SQL Web Toolkit*, se procede a instalar el esquema de nombre *WEBDB*, dejando los valores por defecto para los *Tablespaces*. A partir de este momento el proceso de instalación presenta una serie de ventanas texto *MSDOS* en las que se ejecuta el producto *SQL*Plus* para cargar los paquetes del *WebDB*. Estas ventanas se quedan bloqueadas y deben ser terminadas manualmente por el usuario para que el proceso de instalación continúe.

Una vez concluido el proceso, se pulsa el botón *Salir* desde la ventana de instalación de componentes para terminar.

16.5. CREACIÓN DE USUARIOS WebDB

Una vez que se ha instalado el *Oracle8i Personal Edition* y *Oracle WebDB*, el siguiente paso es la creación de un usuario que pueda utilizar el producto *WebDB*. Debido a las características de este producto, el usuario no puede ser creado desde las opciones de administración de *Oracle*, sino que debe crearse desde el mismo *WebDB*.



La creación de un usuario de *WebDB* se realiza conectándose al *WebDB* con el *login* por defecto del producto (*login*: WEBDB, *password*: WEBDB), a no ser que haya sido modificado.

La conexión a *WebDB* se realiza a través de un navegador estándar, ejecutando la siguiente instrucción:

URL: *http://equipo_local: puerto/*,

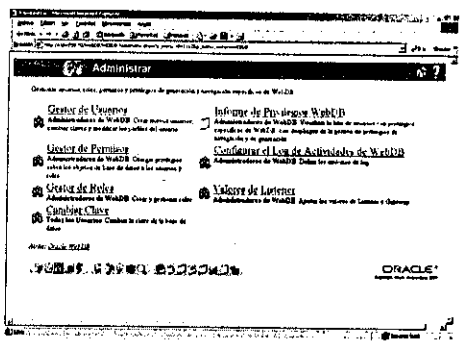
donde: *equipo_local*, es el nombre de la máquina en donde se encuentra instalado *WebDB*, y *puerto*, es el número de puerto que se asignó en la instalación del producto.

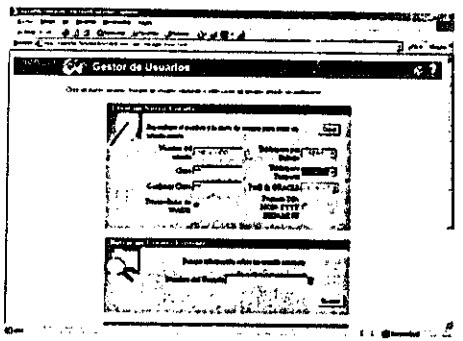
Una vez introducido el *login* y *password* y pulsado el botón *Aceptar*, se realiza la conexión como usuario con privilegios de administración, lo que le permite la definición de nuevos usuarios.

Para crear un nuevo usuario se avanza por los menús que presenta *WebDB* por las opciones: *Administrar*, *Gestor de Usuarios*, hasta alcanzar el formulario de creación de nuevos usuarios, el cual se completará con la siguiente información:

Nombre de usuario	CHENCODD	*Clave*	C2C
Desarrollador de WEBDB	ACTIVADO	*Tablespace por defecto*	USERS
Tablespace temporal	TEMP		

El siguiente paso es asignar permisos a este usuario. Estos permisos serán asignados de forma automática en el proceso de instalación de la base de datos con los ejercicios que se incluye en el CD.





16.6. CARGA DE LA EXTENSIÓN DE LA BASE DE DATOS

Una vez que se han completado los pasos anteriores, el siguiente paso consiste en almacenar en la base de datos que ha sido previamente creada, la extensión correspondiente a la misma, la cual consiste en todos los ejercicios que se describen en el texto, más un conjunto de ejercicios adicionales.

Puesto que en dichos ejercicios se ha utilizado el paquete UTL_FILE para poder manipular ficheros de texto en el lado del servidor de la base de datos, se deben tener en cuenta los mecanismos de seguridad para restringir el acceso a los directorios del mismo. Esta restricción se realiza a través del parámetro UTL_FILE_DIR del fichero de inicialización de la base de datos que se encuentra en la siguiente localización:

```
C:\Oracle\admin\libro\pfile\INIT.ORA.
```

Cada uno de los directorios accesibles se indica a través de una línea en dicho fichero de la forma siguiente:

```
UTL_FILE_DIR= directorio
```

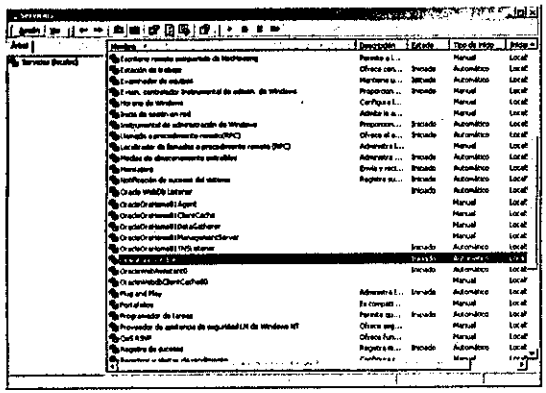
donde: *directorio* representa la localización (ruta de directorios) donde se encuentran los ficheros a los cuales se va a acceder.

Como la ejecución de cada uno de los ejercicios se va a realizar de forma local se ha optado por asignar a dicho parámetro el siguiente valor:

```
UTL_FILE_DIR= *
```

con lo cual se indica que se tiene acceso a todos los directorios del disco.

El cambio realizado en el punto anterior implica una modificación en el fichero de inicialización de la base de datos. Por tanto, se debe reiniciar la instancia que



representa a dicha base de datos. *Windows 2000* maneja la instancia de la base de datos como un servicio automático denominado:

OracleServiceNombrebasededatos

que se arranca al iniciar el sistema —el cual y para la base de datos de nuestro ejemplo se denominará *OracleServiceLibro*—.

Para reiniciar la instancia de la base de datos primero se debe parar, y después volver a arrancar dicho servicio. La forma de reiniciar dicho servicio es a través de la utilidad *Servicios* del *Windows 2000*, a la cual se puede acceder de la forma:

```
Panel de Control:Herramientas administrativas:Servicios
```

Si se pulsa dos veces sobre el icono *Servicios*, aparece una nueva ventana con todos los servicios de *Windows 2000*, y una barra de herramientas con su barra de botones que permitirá iniciar y detener el servicio correspondiente a la base de datos.

Una vez reinicializado el servicio correspondiente a la base de datos, se puede proceder a la carga de la información de la misma. Esta carga se realiza de forma automática ejecutando el fichero *instalación.bat* que se encuentra en el directorio *raíz* del CD.

16.7. DESINSTALACIÓN

Si el usuario desea desinstalar la información correspondiente a la base de datos, en el CD se incluye el fichero *desinstalación.bat*, cuya ejecución realiza esta tarea de forma automática. Este fichero se encuentra ubicado en el mismo directorio *raíz* del CD.

16.8. ORGANIZACIÓN Y CONTENIDO DEL CD

El CD que acompaña la obra incluye una serie de ficheros de datos y por lotes que facilitará al lector la comprensión de los conceptos y ejercicios prácticos propuestos en la misma, así como el proceso de instalación de la base de datos descrita en este capítulo. El contenido del CD se ha organizado en una serie de directorios de la forma siguiente:

Directorio Raíz: en el cual se encuentran los siguientes ficheros:

- *Instalación.bat*: cuya ejecución da lugar a la carga automática de los datos correspondiente a la base de datos descrita en este capítulo.

- *Desinstalación*.bat: cuya ejecución da lugar a la desinstalación, de forma automática de los datos, y la base de datos, descrita en este capítulo.
- *Index*.html: es la página principal de la interfaz que se incluye en el CD, y mediante la cual el usuario puede acceder al contenido del mismo y ejecutar la aplicación desarrollada en *WebDB* que hace uso de la base de datos descrita en este capítulo.

Directorio Ejercicios: en el cual se encuentran una serie de subdirectorios denominados *CapítuloXX*, donde *XX* es el número del capítulo correspondiente en el texto, y en donde se encuentran los siguientes ficheros:

- *CapXXgees.bat*: cuya ejecución da lugar al borrado de la base de datos y generación del esquema del ejemplo descrito en el capítulo correspondiente (por ejemplo si XX=06, se borrará de la base de datos el ejemplo del *Catastro Municipal* y se generará su esquema de nuevo).
- *CapXXgein.bat*: cuya ejecución da lugar al borrado y nueva carga de los datos del ejemplo descrito en el capítulo correspondiente. Este y/o anterior fichero por lotes, lo deberá ejecutar el lector cuando quiera restaurar las bases de datos de los ejemplos a su estado inicial. Estas operaciones se recomiendan después de que el usuario haya ejecutado cualquier operación que de lugar a una modificación de la mismas, si desea que los resultados de las nuevas manipulaciones de las bases de datos coincidan con los que se incluyen en el texto[39].
- *Directorio Texto*: en el que se incluyen una serie de ficheros que incluyen los ejercicios descritos en el libro para cada uno de los problemas resueltos. Los ficheros tienen una denominación de la forma: *CapXXejYZ.sql*, donde:
  - *YZ* toma valores numéricos para los ejercicios de *SQL*.
  - *Y toma el valor* "P", *y Z valores numéricos para los ejercicios de PL/SQL*.
  - *YZ* toma el valor "*es*" para el *script* de definición del esquema.
  - *YZ* toma el valor "*in*" para el *script* de inserción de la extensión de la base de datos.
- *Directorio Adicionales*: en el que se incluyen una serie de ficheros que incluyen los ejercicios adicionales tanto de *SQL*, como de *PL/SQL*. Los ficheros tienen una denominación de la forma: *CapXXadYZ.sql*.

Directorio Apéndices: en este directorio se encuentra una colección de documentos *Microsoft Word* en los que se presentan una serie de ejercicios para que el lector practique en la resolución de los mismos. En estos archivos se plantean diez problemas diferentes para que el lector realice el modelo conceptual y su correspondiente derivación al esquema relacional. Cada problema se acompaña de

[39] Después de la ejecución de los ficheros *CapXXgees.bat* o *CapXXgein.bat*, el usuario deberá teclear *EXIT* para abandonar la ventana de ejecución.

una serie de guías que facilitarán al lector la toma de decisiones en el proceso de análisis y resolución de los mismos.

Directorio Frames: en este directorio se encuentra un conjunto de archivos que conforman la interfaz desarrollada para que el lector pueda acceder y manipular todo el contenido del CD. Se trata, principalmente, de archivos .HTML que son accedidos a través de la página principal (*index.html*) de la interfaz.

Directorio Images: en este directorio se encuentran las imágenes que son utilizadas en la interfaz que se incluye en el CD.

Directorio bin: se trata de un directorio oculto, al cual se recomienda no acceder a usuarios no expertos. En este directorio se encuentra un conjunto de archivos y procedimientos *SQL* y *PL/SQL* que son utilizados por las macros de instalación, desinstalación y acceso a la base de datos descrita en este capítulo a través del *WebDB*.

El lector podrá acceder y hacer uso de la información contenida en el CD de cualquiera de las formas siguientes:

- Directamente a través del explorador de *Windows*, puede acceder a los ficheros texto que almacenan los ejercicios (desarrollados en la obra y los adicionales) para cada uno de los problemas resueltos. Además, si el lector tiene instalado el servidor de bases de datos *Personal Oracle 8i* (o superior), puede ejecutar estos archivos a través de la interfaz de *SQL*PLUS*, siempre que haya instalado previamente la base de datos descrita en este capítulo.
- Ejecutando el archivo *Index.html*, que se encuentra en el directorio raíz del CD. Esta operación se puede realizar haciendo uso de cualquier navegador estándar, y da lugar a la presentación de una interfaz a través de la cual el lector puede acceder a:
  - El contenido de los archivos con los ejercicios resueltos en el texto y los adicionales para los Capítulos 4, y 6 a 15.
  - El contenido de los apéndices en los que se proponen una serie de ejercicios para que el lector practique en su resolución.
  - La aplicación desarrollada con *WebDB*, que manipula la base de datos descrita en este capítulo[40]. Esta base de datos, almacena todos los esquemas correspondientes a los problemas resueltos en la obra, así como los ejercicios descritos y los adicionales. Esta aplicación permite al lector ejecutar todos y cada uno de estos ejercicios, y comprobar el resultado fácilmente.
  - Información adicional sobre la obra y sus autores.

[40] En el primer acceso a la aplicación el usuario deberá indicar el nombre del host y el puerto (*localhost:70*) que han declarado en la instalación del *WebDB*.

BIBLIOGRAFÍA

[BATI94] *Diseño Conceptual de Bases de Datos*, **C. Batini, S. Ceri, S. Navathe**, Addison-Wesley, 1994.

[BOBR00] *Oracle 8i para Windows NT*, **S. Bobrowski**, Osborne-McGraw-Hill, 2000.

[DATE89] *A Guide to the SQL Standard*, **C. J. Date**, Addison-Wesley, 1989.

[DATE93] *Introducción a los Sistemas de Bases de Datos*, **C. J. Date**, Addison-Wesley, 1993.

[DORS99] *Oracle8. Design Using UML Object Modeling*, **P. Dorsey, J.R. Hudicka**, Oracle Press. Osborne-McGraw-Hill, 1999.

[ELMA97] *Sistemas de Bases de Datos: Conceptos Fundamentales*, **R. Elmasri, S. Navathe**, Addison Wesley, 1997.

[GARD87] *Bases de Datos*, **G. Gardarin**, Paraninfo, 1987.

[GARD94] *Dominar las Bases de Datos*, **G. Gardarin**, Ediciones Gestión 2000-Eyrolles, 1994.

[GILL88] *Introducción a las Bases de Datos*, **M. L. Gillenson**, McGraw-Hill, 1988.

[HANS97] *Diseño y Administración de Bases de Datos*, **G.W. Hansen, J.V. Hansen**, Prentice Hall, 1997.

[HAWR90] *Introducción al Análisis y Diseño de Sistemas con Ejemplos Prácticos*, **I. Hawryszkiewycz**, Anaya, 1990.

[KOCH00] *Oracle8i: The Complete Reference, 10th Edition*, **G. Koch, K. Loney**. Oracle Press. Osborne-McGraw-Hill, 2000.

[KROE96] *Fundamentos de Bases de Datos*, **D. Kroenke**, Prentice Hall, 1996.

[LUQU97] *Diseño y Uso de Bases de Datos Relacionales*, **I. Luque Ruiz, M.A. Gómez-Nieto**, RaMa, 1997.

[LUQU98] *Ficheros: Organizaciones Clásicas para el Almacenamiento de la Información*, **I. Luque Ruiz, J.A. Romero del Castillo, M.A. Gómez-Nieto**, Servicio de Publicaciones de la Universidad de Córdoba, 1998.

[LUQU99] *Ingeniería del Software: Fundamentos para el Desarrollo de Sistemas Informáticos*, **I. Luque Ruiz, M.A. Gómez-Nieto**, Servicio de Publicaciones de la Universidad de Córdoba, 1998.

[MART82] *Organización de las Bases de Datos*, **J. Martin**, Prentice-Hall, 1982.

[ORA99-1] *Oracle Application Server. PL/SQL Web Toolkit Reference. Release 4.0.8.* Oracle, 1999.

[ORA99-2] *Oracle Webdb. Creating and Managing Components. Release 2.2.* Oracle. 1999.

[PRAT94] *DataBase Systems Management and Design*, **P.J. Pratt, J.J. Adamski**, Int. Tomson Publishing, 1994.

[RYAN95] *Data Base Systems Engineering*, **N. Ryan, D. Smith**, Int. Thomson Publishing, 1995.

[SAND94] *Data Modeling*, **G. L Sanders**, Int. Tomson Publishing, 1994.

[SILB98] *Procesamiento de Bases de Datos*, **A. Silberschatz, H. F. Korth, S. Sudarshan**, McGraw-Hill, 1998.

[TEOR82] *Design of Database Structures*, **T.J. Teorey, J.P. Fry**, Prentice-Hall, 1982.

[TSIC82] *Data Models*, **D.C. Tsichritzis, F.H. Lochovsky**, Prentice-Hall, 1982.

[ULLM83] *Principles of DataBase Systems*, **J.D. Ullman**, Computer Science Press, 1983.

[URMA99] *Oracle8. Programación PL/SQL.* **S. Urman**, Oracle Press. Osborne-McGraw-Hill, 1999

[WIED86] *Diseño de Bases de Datos*, **G. Wiederhold**, McGraw-Hill, 1986.

ÍNDICE ALFABÉTICO

D

E

F

G

I

J

L

M

N

O

P

R

S

T

NOTAS

Bases de Datos

Desde Chen hasta Codd con ORACLE®

Usuarios de:	Nivel del libro:
Bases de Datos	Iniciación ✓ *Medio* Avanzado

La tecnología de *Bases de Datos* se ha extendido en los últimos años alcanzando a todo tipo de sistemas y usuarios. El avance tecnológico, el abaratamiento del hardware y software, y la implantación de las bases de datos relacionales como un estándar son algunas de las razones de este hecho. Pero recientemente nuevos desafíos han surgido en el uso de la tecnología de bases de datos. La implantación de Internet y la necesidad de expansión y difusión de los sistemas de negocio a través de la red obliga a que las bases de datos deban dar soporte a dominios complejos de información que representan cualquier elemento del dominio del problema al que dan solución. Es por ello que los modernos *Sistemas de Gestión de Bases de Datos*, como es el caso de Oracle a partir de la versión 8i, además de estar basados en el modelo de datos relacional clásico, permiten la representación de la información bajo el paradigma de objetos, originando lo que se denomina como una representación o esquema objeto.relacional, mediante la cual pueden ser manipulados elementos de información complejos (imágenes, ficheros, tablas, arrays, etc.) como si de elementos simples se tratase. Además, estos SGBD pueden ser operados a través de la red haciendo uso de los browsers estándar y lenguajes como Java, de una forma similar a como se hacía tradicionalmente con los lenguajes de tercera generación.

Este libro presenta una revisión actualizada del texto "*Diseño y Uso de Bases de Datos Relacionales*", en la que se ha incluido la visión objeto-relacional de los datos, además del uso de la las bases de datos a través de Internet. En la primera parte del texto se muestran los principios básicos para representar los problemas del mundo real haciendo uso del modelo Entidad-Interrelación(Chen), los fundamentos en que se basa el modelo de datos Relacional(Codd), y una serie de reglas a aplicar en el proceso de traducción desde la representación conceptual del problema a la representación relacional. Además se presenta una descripción del SQL, lenguaje estándar en las bases de datos relacionales, y del lenguaje PL/SQL, un lenguaje procedimental estructurado propio de Oracle, que permite el desarrollo de aplicaciones.

En la segunda parte se presentan de menor a mayor complejidad una serie de problemas completamente resueltos. Para cada problema se realiza paso a paso el análisis del mismo hasta obtener el modelo conceptual para, posteriormente, derivar el correspondiente esquema relacional. Además, cada uno de los diez problemas se acompaña con una solución de ejercicios completamente resueltos haciendo uso de SQL o PL/SQL. Finalmente se incluye un último ejemplo que es utilizado para presentar la interacción de Oracle con Internet. Este ejemplo se acompaña con la descripción del producto WebDB de Oracle, el cual es utilizado en el desarrollo de los ejercicios que se incluyen en esta parte.

El libro incluye un CD-ROM que contiene un gran número de ejercicios resueltos para cada uno de los problemas planteados, y además una aplicación desarrollada con WebDB de Oracle mediante la cual el lector podrá ejecutar todos los ejercicios presentados, lo que aporta un valioso poder pedagógico a la obra. También contiene una interfaz que se ejecuta a través de cualquier navegador estándar que permite consultar toda la información.

Incluye CD-ROM con los ejercicios del libro

ISBN 84-7897-478-4

Ra-Ma®