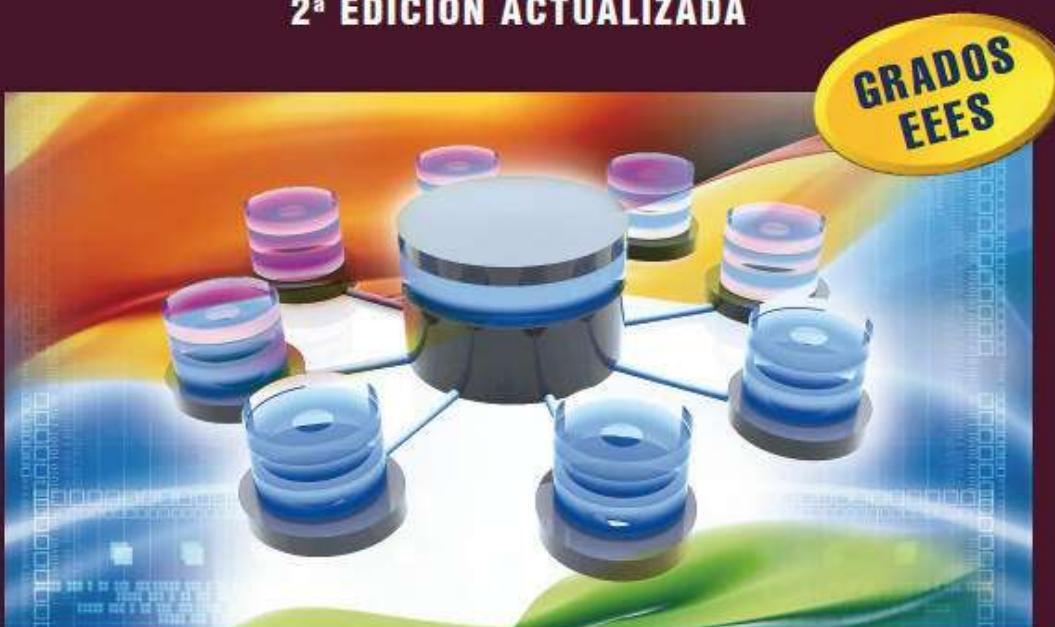


Desarrollo de Bases de Datos

**Casos prácticos desde
el análisis a la implementación**

2^a EDICIÓN ACTUALIZADA



www.ra-ma.es

Desde www.ra-ma.es podrá
descargarse material adicional.

**Dolores Cuadra • Elena Castro • Ana M^a Iglesias
Paloma Martínez • Fco. Javier Calle • César de Pablo
Harith Al-Jumaily • Lourdes Moreno • Sonia García Manzano
José Luis Martínez • Jesica Rivero • Isabel Segura**



Ra-Ma®

Desarrollo de Bases de Datos

**Casos Prácticos desde el Análisis a la
Implementación**

2.^a Edición actualizada

Descarga de Material Adicional

Este E-book tiene disponible un material adicional que complementa el contenido del mismo.

Este material se encuentra disponible en nuestra página Web www.ra-ma.com.

Para descargarlo debe dirigirse a la ficha del libro de papel que se corresponde con el libro electrónico que Ud. ha adquirido. Para localizar la ficha del libro de papel puede utilizar el buscador de la Web.

Una vez en la ficha del libro encontrará un enlace con un texto similar a este:

“Descarga del material adicional del libro”

Pulsando sobre este enlace, el fichero comenzará a descargarse.

Una vez concluida la descarga dispondrá de un archivo comprimido. Debe utilizar un software descompresor adecuado para completar la operación. En el proceso de descompresión se le solicitará una contraseña, dicha contraseña coincide con los 13 dígitos del ISBN del libro de papel (incluidos los guiones).

Encontrará este dato en la misma ficha del libro donde descargó el material adicional.

Si tiene cualquier pregunta o duda en ponerse en contacto con nosotros en la siguiente dirección de correo:ebooks@ra-ma.com

Desarrollo de Bases de Datos

**Casos Prácticos desde el Análisis a la
Implementación**

2.^a Edición actualizada

Dolores Cuadra

Elena Castro

Ana M.^a Iglesias

Paloma Martínez

Francisco Javier Calle

César de Pablo

Harith Al-Jumaily

Lourdes Moreno

Jesica Rivero

José Luis Martínez

Isabel Segura

Sonia García





DESARROLLO DE BASES DE DATOS: CASOS PRÁCTICOS DESDE EL ANÁLISIS A LA IMPLEMENTACIÓN

© Dolores Cuadra, Elena Castro, Ana M. Iglesias, Paloma Martínez, Francisco Javier Calle, César de Pablo, Harith Al-Jumaily, Lourdes Moreno, Jesica Rivero, José Luis Martínez, Isabel Seguray, Sonia García.

© De la Edición Original en papel publicada por Editorial RA-MA

ISBN de Edición en Papel: 978-84-9964-124-9

Todos los derechos reservados © RA-MA, S.A. Editorial y Publicaciones, Madrid, España.

MARCAS COMERCIALES. Las designaciones utilizadas por las empresas para distinguir sus productos (hardware, software, sistemas operativos, etc.) suelen ser marcas registradas. RA-MA ha intentado a lo largo de este libro distinguir las marcas comerciales de los términos descriptivos, siguiendo el estilo que utiliza el fabricante, sin intención de infringir la marca y solo en beneficio del propietario de la misma. Los datos de los ejemplos y pantallas son ficticios a no ser que se especifique lo contrario.

RA-MA es una marca comercial registrada.

Se ha puesto el máximo esfuerzo en ofrecer al lector una información completa y precisa. Sin embargo, RA-MA Editorial no asume ninguna responsabilidad derivada de su uso ni tampoco de cualquier violación de patentes ni otros derechos de terceras partes que pudieran ocurrir. Esta publicación tiene por objeto proporcionar unos conocimientos precisos y acreditados sobre el tema tratado. Su venta no supone para el editor ninguna forma de asistencia legal, administrativa o de ningún otro tipo. En caso de precisarse asesoría legal u otra forma de ayuda experta, deben buscarse los servicios de un profesional competente.

Reservados todos los derechos de publicación en cualquier idioma.

Según lo dispuesto en el Código Penal vigente ninguna parte de este libro puede ser reproducida, grabada en sistema de almacenamiento o transmitida en forma alguna ni por cualquier procedimiento, ya sea electrónico, mecánico, reprográfico, magnético o cualquier otro sin autorización previa y por escrito de RA-MA; su contenido está protegido por la Ley vigente que establece penas de prisión y/o multas a quienes, intencionadamente, reprodujeren o plagiaren, en todo o en parte, una obra literaria, artística o científica.

Editado por:

RA-MA, S.A. Editorial y Publicaciones

Calle Jarama, 33, Polígono Industrial IGARSA 28860 PARACUELLOS DE JARAMA, Madrid

Teléfono: 91 658 42 80. Fax: 91 662 81 39

Correo electrónico: editorial@ra-ma.com

Internet: www.ra-ma.es y www.ra-ma.com

Maquetación: Gustavo San Román Borrueto

Diseño Portada: Antonio García Tomé

ISBN: 978-84-9964-425-7

ookdesarrolladoenEspañaenseptiembrede2014.

La transformación a libro electrónico del presente título fue realizada por Sextil Online, S.A. de C.V./ Ink it
® 2017.

+52 (55) 52 54 38 52

contacto@ink-it.ink

www.ink-it.ink

Deseamos expresar nuestro agradecimiento a todos nuestros alumnos de la Universidad Carlos III de Madrid, que han sido nuestra motivación para la realización de este libro.

ÍNDICE

PREFACIO

CAPÍTULO 1. DISEÑO CONCEPTUAL

1.1 GUÍAS METODOLÓGICAS PARA ABORDAR LA RESOLUCIÓN DE LOSPROBLEMAS

- 1.1.1 Algunas heurísticas para la identificación de los distintosconstructores
- 1.1.2 Entidades frente aatributos
- 1.1.3 Entidadesfrenteatributostmultivaluados
- 1.1.4 Entidadesfrenteinterrelaciones

1.2 NOTACIONES

1.3 CÓMO SE ESTRUCTURAN LOSPROBLEMAS

PROBLEMA 1.1: GESTIÓN DEALQUILERES

PROBLEMA 1.2: ADMINISTRACIÓN DE FINCAS ENUNCIADO

PROBLEMA 1.3. MEDICAMENTOS

PROBLEMA 1.4: PROYECTOS DEINVESTIGACIÓN

PROBLEMA 1.5: VIAJES DE INVESTIGACIÓN

PROBLEMA 1.6: GESTIÓN DE PROYECTOS INFORMÁTICOS

PROBLEMA 1.7: MEDIO AMBIENTE

PROBLEMA 1.8: CLÍNICA OFTALMOLÓGICA ENUNCIADO

PROBLEMA 1.9: LONJA DE PESCADO

PROBLEMA 1.10: VIVEROS

CAPÍTULO 2. DISEÑO LÓGICO

2.1 RECORDATORIO DE LA ESTÁTICA DEL MODELO RELACIONAL

2.1.1 Notación

2.2.1 REGLAS DE TRANSFORMACIÓN DE UN ESQUEMA E/R A UN ESQUEMA RELACIONAL

2.2.1 Transformación de entidades, atributos y dominios

2.2.2 Transformación de interrelaciones N:M

2.2.3 Transformación de interrelaciones 1:N

2.2.4 Transformación de otros elementos del Modelo E/R Extendido

2.2.5 Transformación de Dependencias en Existencia y en Identificación

2.2.6 Transformación de Generalizaciones

2.2.7 Transformación de Interrelaciones de grado superior a dos

2.2.8 Transformación de Interrelaciones Exclusivas

2.2.9 Algunos aspectos sobre la pérdida de semántica en la transformación al Modelo Relacional

2.3 DINÁMICA DEL MODELO RELACIONAL: ÁLGEBRA RELACIONAL

2.4 CÓMO SE ESTRUCTURAN LOS PROBLEMAS

PROBLEMA 2.1: ALOJAMIENTOS RURALES

PROBLEMA 2.2: LA TIENDA DE REGALOS

PROBLEMA 2.3: COMPAÑÍA TEATRAL

PROBLEMA 2.4: CAMPEONATO DE AJEDREZ

PROBLEMA 2.5: FÁBRICA DE MUEBLES

PROBLEMA 2.6: OBSERVACIÓN DE AVES
PROBLEMA 2.7: GESTIÓN DE EMPRESA DE OCIO
PROBLEMA 2.8: GESTIÓN DE INCENDIOS
PROBLEMA 2.9: GRAN PREMIO DE FÓRMULA 1
PROBLEMA 2.10: FEDERACIÓN DE TAXIS
PROBLEMA 2.11: VIAJES DE INVESTIGACIÓN
PROBLEMA 2.12: GESTIÓN DE ALQUILERES
PROBLEMA 2.13: PROYECTOS DE INVESTIGACIÓN
PROBLEMA 2.14: GESTIÓN DE PROYECTOS INFORMÁTICOS
PROBLEMA 2.15: MEDICAMENTOS
PROBLEMA 2.16: PROYECTOS I+D
PROBLEMA 2.17: VUELTA CICLISTA
PROBLEMA 2.18: TARJETAS DESCUENTO
PROBLEMA 2.19: AGENCIA DE CASTINGS
PROBLEMA 2.20: COMPAÑÍA TELEFÓNICA

CAPÍTULO 3. BASES DE DATOS DISTRIBUIDAS

3.1 INTRODUCCIÓN

3.2 CLASES Y ARQUITECTURAS DE LOS SGBDD

3.3 DISEÑO DE BDD

3.3.1 Esquema de fragmentación

3.3.2 Esquema de asignación y replicación

PROBLEMA 3.1: BDD SOCIEDADES MÉDICAS

PROBLEMA 3.2: BDD UNIVERSIDAD CARLOS III

PROBLEMA 3.3: BDD SERVICIOS INFORMÁTICOS

PROBLEMA 3.4: BDD OFICINAS DEL INEM

PROBLEMA 3.5: BDD OFICINAS DE SEGUROS

PROBLEMA 3.6: BDD EMPRESA DE COSMÉTICOS

CAPÍTULO 4. ORGANIZACIONES DE FICHEROS

4.1 CONCEPTOS BÁSICOS

4.2 ORGANIZACIONES BASE

4.2.1 Organización serial

4.2.2 Organización secuencial

4.2.3 Organizaciones no consecutivas

4.2.4 Organizaciones direccionadas

4.3 ORGANIZACIONES AUXILIARES

4.3.1 Índices primarios, secundarios y clusters

4.3.2 Organizaciones indizadas multinivel

4.4 ACCESO MULTICLAVE

4.4.1 Acceso invertido

PROBLEMA 4.1: ORGANIZACIONES BASE E INDIZADAS I

PROBLEMA 4.2: ORGANIZACIONES BASE E INDIZADAS II

PROBLEMA 4.3: COMPARACIÓN DE ORGANIZACIONES

PROBLEMA 4.4: ORGANIZACIÓN INDIZADA I

PROBLEMA 4.5: ORGANIZACIÓN INDIZADA II

**PROBLEMA 4.6: DISEÑO Y COMPARACIÓN DE
ORGANIZACIONES**

PROBLEMA 4.7: ACCESO MULTICLAVE

CAPÍTULO 5. ALMACENAMIENTO Y DISEÑO FÍSICO EN ORACLE

5.1 ESPACIOS DE TABLAS

- 5.1.1 Segmentos y extensiones
- 5.1.2 Bloques
- 5.1.3 Tablas

5.2 ESTRUCTURAS AUXILIARES

- 5.2.1 Índices en árbol B+
- 5.2.2 Índices en mapa de bits
- 5.2.3 Tablas organizadas como índices
- 5.2.4 Agrupaciones y agrupaciones asociativas
- 5.2.5 Partición de tablas

5.3 DISEÑO FÍSICO

- 5.3.1 Selección de índices

PROBLEMA 5.1: ESTIMACIÓN DEL TAMAÑO DE UNA BASE DE DATOS

PROBLEMA 5.2: ESTIMACIÓN DEL VOLUMEN DE UNA TABLA

PROBLEMA 5.3: CREACIÓN DE ÍNDICES

PROBLEMA 5.4: SELECCIÓN DE ÍNDICES EN UNA BASE DE DATOS

CAPÍTULO 6. CASOS PRÁCTICOS: DISEÑO E IMPLEMENTACIÓN EN EL SGBD ORACLE

6.1 CASO PRÁCTICO 1: GESTIÓN FONDOS DE UN MUSEO

- 6.1.1 Diseño conceptual: Esquema E/R
- 6.1.2 Diseño lógico: Transformación al Esquema Relacional

6.1.3 Diseño Lógico Específico

6.2 CASO PRÁCTICO 2: EDITORIAL DE LIBROS DE TEXTO ESCOLAR

6.2.1 Diseño conceptual: Esquema E/R

6.2.2 Diseño Lógico: Transformación al Esquema Relacional

6.2.3 Diseño Lógico Específico

BIBLIOGRAFÍA

MATERIAL ADICIONAL

ÍNDICE ALFABÉTICO

PREFACIO

La obra se centra en el diseño de bases de datos desde un punto de vista eminentemente práctico. Nuestra experiencia docente en la universidad ha puesto de manifiesto la dificultad a la que se enfrentan los alumnos cuando realizan tareas de análisis y diseño de bases de datos. Por ello, los objetivos que nos hemos propuesto al escribir este libro son:

- Proporcionar una colección de problemas que sirva al lector para poner en práctica los conceptos teóricos de análisis, diseño e implementación de bases de datos.
- Abordar la resolución de los problemas de una forma comprensible justificando en todo momento las decisiones de diseño.
- Emplear unos principios metodológicos que ayuden a los analistas y diseñadores de bases de datos a elaborar esquemas conceptuales y lógicos intentando perder la mínima semántica.
- Dar a conocer las organizaciones básicas de ficheros para poder realizar un buen diseño físico.
- Algunos de los ejercicios propuestos provienen de prácticas y exámenes propuestos en nuestras asignaturas de Ficheros y Bases de Datos en el Grado de Ingeniería Informática, así como en las titulaciones de Ingeniería Técnica en Informática de Gestión e Ingeniería Informática de la Universidad Carlos III de Madrid, en la que los autores son profesores. La idea fue elaborar un texto que fuera el complemento práctico a los libros de Diseño de Bases de Datos, así como un material con el que el profesor de cualquier asignatura de Diseño y Organización de Ficheros puede contar para elaborar sus clases.

CONTENIDOS

El libro está estructurado en cuatro grandes bloques: Diseño Conceptual, Diseño Lógico, Bases de Datos Distribuidas y Diseño Físico (Organización de Ficheros y Administración en ORACLE 10g). Al finalizar estos bloques

temáticos se presenta un caso práctico que aplica las fases de una metodología para el diseño de una base de datos. La razón de esta división viene dada porque creemos que esta estructura es la que mejor se ajusta a las fases metodológicas y aspectos cruciales del Diseño de Bases de Datos.

El Capítulo 1 de Diseño Conceptual contiene una parte introductoria en la que se recuerdan brevemente los conceptos teóricos y notación del modelo E/R así como algunas guías metodológicas para la elaboración de las soluciones a los casos propuestos. El resto de este capítulo lo forman 10 problemas desglosados en varios pasos; en cada paso se estudiará un conjunto de supuestos semánticos que darán lugar a un subesquema E/R de forma que se irán añadiendo elementos al subesquema obtenido en el paso anterior y así sucesivamente hasta completar el estudio de todos los supuestos semánticos contemplados en el enunciado del problema. Supondremos que el enunciado constituye una descripción correcta (y casi siempre completa) del Universo del Discurso.

El Capítulo 2 de Diseño Lógico contiene una introducción a modo de recordatorio del modelo relacional y su notación, reglas de transformación de un esquema E/R a un esquema relacional y álgebra relacional. A continuación se presentan dos bloques de problemas. El primer bloque engloba 10 problemas de modelado relacional en los que se obtiene el esquema lógico estándar de la BD (grafo relacional) a partir de un conjunto de supuestos semánticos que describen el problema. Además, se muestran varias consultas en SQL3 y álgebra relacional. El segundo bloque lo forman 10 problemas (cuyos enunciados se resolvieron en capítulo de Diseño Conceptual) en los que se obtiene un esquema relacional a partir de un esquemaE/R.

El Capítulo 3, dedicado a las Bases de Datos Distribuidas, contiene una introducción en la que se describen las características y arquitecturas de las Bases de Datos Distribuidas así como los algoritmos de diseño de este tipo de bases de datos. Se muestran 6 problemas de diseño en los que se construyen los esquemas de fragmentación y asignación a partir de las especificaciones del problema.

El Diseño Físico se ha abordado en dos capítulos (4 y 5). El capítulo Organizaciones de Ficheros presenta una introducción teórica sobre la distribución en almacenamiento secundario de la información y cómo ganar eficiencia con estructuras auxiliares unidas a las organizaciones base para disminuir los accesos a soporte secundario. El capítulo de Almacenamiento y diseño físico en Oracle muestra las distintas organizaciones que el Sistema Gestor Oracle 10g proporciona para que realizar un buen diseño físico de la base

de datos. A través de cuatro problemas se muestra la utilidad de una buena gestión física para ganar eficiencia en el sistema.

Finalmente, el último capítulo (6) incluye dos casos prácticos en los que se aplican las fases de una metodología de diseño de bases de datos. Por tanto, recapitula todo lo aprendido en los bloques de Diseño Conceptual y Lógico.

ORIENTACIÓN A LOS LECTORES

La audiencia a la que va dirigida el libro es muy amplia: alumnos de grado en las asignaturas de Bases de Datos, Diseño de Bases de Datos y Organización de Ficheros en facultades, escuelas universitarias y escuelas de Formación Profesional, así como profesionales informáticos que trabajen en el área de bases de datos (analistas, programadores,etc.).

ESTRUCTURA DEL LIBRO

Para que al lector le resulte cómodo el manejo del libro de problemas presentamos (Figura P.1) las interrelaciones entre los distintos capítulos y los conocimientos previos que debería poseer para poder acometer la resolución de los problemas propuestos. El primer capítulo no necesita de otros para poder resolver los problemas presentados en el mismo. Para poder abordar los problemas del Capítulo 3 (Bases de Datos Distribuidas) es necesario tener conocimiento teórico de Bases de Datos Relacionales. El primer apartado del Capítulo 2 resuelve problemas de diseño relacional sin pasar por la fase conceptual del diseño, por lo que con el conocimiento del modelo relacional es suficiente para comprender los problemas presentados en este apartado. Los conceptos teóricos que muestra el Capítulo 4 son necesarios para entender el Capítulo 5 (al ser un caso particular de diseño físico en un SGBD). Las fases de la metodología de diseño de bases de datos pueden seguirse desde el Capítulo 1, pasando por el apartado 2.2 del Capítulo 2 y, por último, el Capítulo 5. Para terminar, el Capítulo 6 necesita de los capítulos mencionados anteriormente para poder entender los casos prácticos mostrados en el mismo.

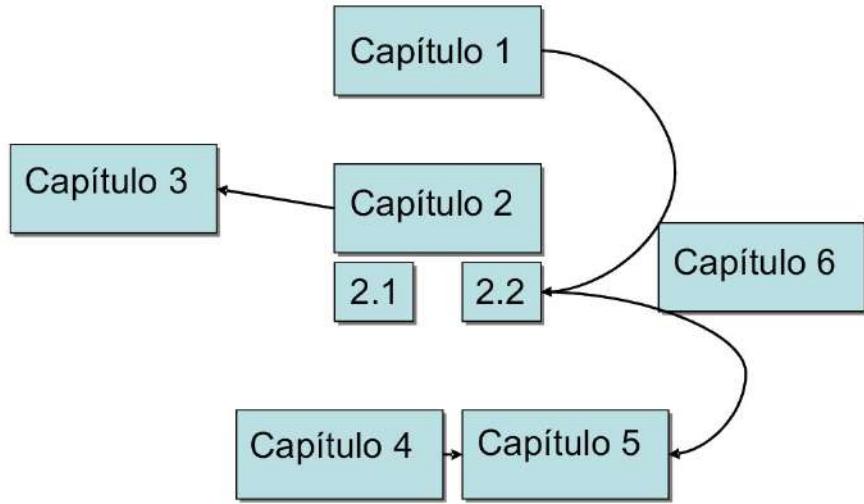


Figura P.1. Relaciones entre los capítulos presentados en el libro

TESTIMONIO DE RECONOCIMIENTO

Deseamos expresar nuestro agradecimiento a nuestras familias y amigos que han soportado pacientemente nuestras horas extras de trabajo y que nos han ofrecido en todo momento su apoyo incondicional.

Los autores

Madrid, enero de 2013

Capítulo 1

DISEÑO CONCEPTUAL

1.1 GUÍAS METODOLÓGICAS PARA ABORDAR LA RESOLUCIÓN DE LOSPROBLEMAS

La elaboración de un esquema E/R que recoja la semántica de un determinado Universo del Discurso es un proceso creativo para el que no existe un procedimiento definido. Sin embargo, sí es posible seguir una serie de recomendaciones o heurísticas que nos ayuden en el diseño. Estas recomendaciones no son reglas que siempre funcionen, sino que en algunos casos son adecuadas y en otrosno.

1.1.1 Algunas heurísticas para la identificación de los distintosconstructores

En la primera propuesta del Modelo E/R, Chen (1976), se distinguen tan solo tres clases de objetos: entidades, atributos e interrelaciones. El modelo que se utilizará en la fase de modelado conceptual en los ejercicios de este libro es el Modelo E/R extendido que añade al modelo básico un conjunto de constructores que ayudan a recoger mayor semántica del universo del discurso (cardinalidades mínimas y máximas en las interrelaciones, generalizaciones, dependencia en existencia y en identificación, etc.). Realmente no se trata de un único modelo sino de una familia de modelos, dado que cada autor lo especializa y le añade distintos constructores. En este apartado se expondrán brevemente cuáles son aquellos de los que consta el modelo E/R extendido con el fin de facilitar la comprensión de los problemas.

Las **entidades** son los objetos principales sobre los que debe recogerse información y generalmente denotan personas, lugares, cosas o eventos de interés. Las entidades aparecerán reflejadas en el enunciado habitualmente como nombres. A cada una de las posibles ocurrencias (cada persona, lugar, cosa o evento concreto) de la entidad se le denomina *ejemplar*.

Los **atributos** se utilizan para detallar las entidades asignándoles propiedades descriptivas tales como nombre, color y peso. Existen dos tipos de atributos: *identificadores* y *descriptores*. Los primeros se utilizan para distinguir de manera única cada una de las ocurrencias de una entidad (distinguiéndose entre *identificadores principales* e *identificadores alternativos*), mientras que los descriptores se utilizan para describir una ocurrencia de entidad. No solo es

possible especificar atributos en las entidades, sino también en las interrelaciones (en este caso solo tiene sentido hablar de atributos descriptores y no de identificadores). Los atributos también aparecerán reflejados en el enunciado, generalmente, como nombres.

En relación con los atributos también existe el concepto de *dominio* (conjunto de valores sobre los que se define el atributo). Aunque se pueden representar explícitamente en los diagramas E/R (como se muestra en el siguiente apartado donde se reflejan las notaciones) en los problemas de este libro se considerará que el dominio toma el mismo nombre que el del atributo. Por ejemplo, la entidad *EMPLEADO* puede tener el atributo *Estado civil* definido sobre el dominio Estados civiles (compuesto por los valores: soltero, casado, viudo, divorciado) y cuyas dos posibles representaciones en el diagrama E/R se muestran en la Figura 1.1.

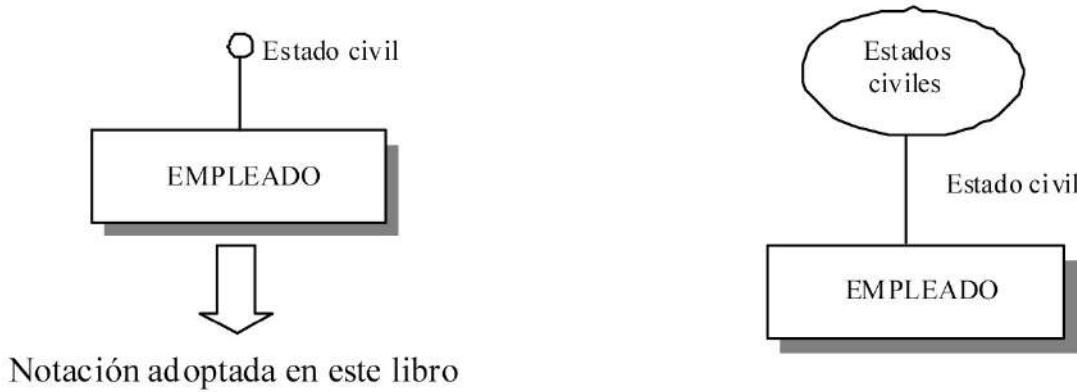


Figura 1.1. Representación de dominios

También es posible recoger otras restricciones semánticas sobre los atributos, aparte de las ya mencionadas de atributos identificadores principales y alternativos. Así, hablamos de atributos *obligatorios/opcionales* (si un atributo debe tomar o no un valor), atributos *univaluados/multivaluados* (si un atributo toma un único valor o varios), atributos *derivados* (si su valor se obtiene a partir de otros elementos del esquema E/R) y atributos *compuestos/simples* (dependiendo de si un atributo es o no un agregado de otros atributos). A su vez, estas restricciones se pueden combinar entre sí (pueden existir en un esquema E/R atributos multivaluados simples opcionales, univaluados compuestos opcionales, multivaluados obligatorios, multivaluados compuestos, etc.).

Las entidades pueden clasificarse por la fuerza de sus atributos identificadores, es decir, por su dependencia o no dependencia respecto a otras

entidades. Las entidades *fuertes* tienen existencia propia, es decir, poseen identificadores internos que determinan de manera única la existencia de sus ocurrencias. Las entidades *débiles* pueden serlo por dos motivos: bien porque su existencia en la BD depende de una entidad fuerte, bien porque requieran para su identificación de los atributos identificadores (algunas veces llamados atributos externos) de otra entidad. Por ejemplo, no poseen atributos identificadores internos que permitan la identificación de cada una de sus ocurrencias y requieren la presencia de atributos externos. En el primero de los casos se habla de *Dependencia en Existencia* y en el segundo de *Dependencia en Identificación*¹.

Finalmente, las **interrelaciones** representan asociaciones del mundo real entre una o más entidades. Las interrelaciones se caracterizan por su nombre, el grado (número de entidades que participan en la interrelación), el tipo de correspondencia (número máximo de ejemplares de una entidad asociados a una combinación de ejemplares de las otras entidades en la interrelación, que puede ser 1 ó N). Así, en el ejemplo de la Figura 1.2 se observa que el tipo de correspondencia de la interrelación **Participar** es 1:N, es decir, un Empleado participa como máximo en un proyecto y en un proyecto participan como máximo N empleados. Al igual que en las entidades, se denomina ejemplar de la interrelación a cada combinación de ejemplares de las entidades interrelacionadas que constituyen una ocurrencia en la interrelación.

Un constructor que amplía la semántica recogida en una interrelación es la *restricción de cardinalidad*. Se definen las cardinalidades máximas y mínimas de las entidades que participan en una interrelación como el número máximo y mínimo de ejemplares de una entidad que puede relacionarse con un único ejemplar de la otra, u otras entidades que participan en la interrelación.

Gráficamente, las restricciones de cardinalidad se representan por una etiqueta, (0,1), (1,1), (0,N) o (1,N)², situada en la línea que conecta la entidad con el rombo que representa el tipo de interrelación (ver Figura1.2).

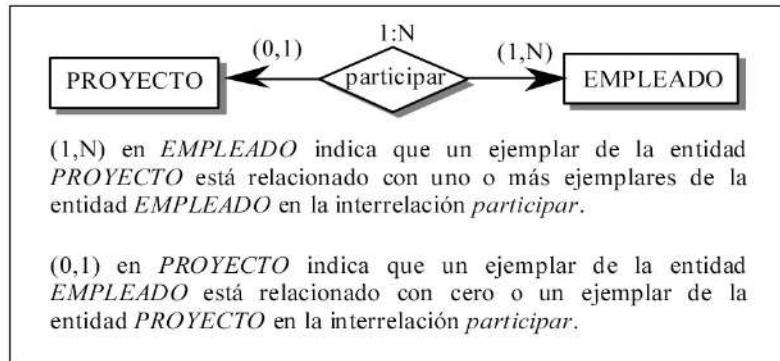


Figura 1.2. Ejemplo de cardinalidades mínimas y máximas

En los enunciados, la aparición de verbos podrá indicarnos, en algunos casos, la existencia de una interrelación en el esquema E/R.

En cuanto a las **generalizaciones**, nos proporcionan un mecanismo de abstracción que permite especializar una entidad (que se denominará *supertipo*) en *subtipos*, o lo que es igual, generalizar los subtipos en el supertipo. De esta forma vemos un conjunto de ocurrencias de una entidad como ocurrencias de otra entidad (como sucede también en las jerarquías *es_un* de las redes semánticas). Por ejemplo, una *Persona* es un *Animal* y un *Reptil* es un *Animal*; en este caso, *Animal* puede considerarse el supertipo y *Persona* y *Reptil* son subtipos de *Animal*. Las ocurrencias o ejemplares de *Persona* lo son también de *Animal*, e igual sucede con las de *Reptil*.

Podremos identificar generalizaciones si encontramos una serie de atributos comunes a un conjunto de entidades. Esta situación se da con frecuencia si se está realizando un proceso de integración de vistas. Estos atributos comunes describirán al supertipo y los atributos particulares permanecerán en los subtipos. Puede ocurrir que los subtipos no tengan atributos propios; en ese caso, solo existirán subtipos (aun cuando por recoger mejor la semántica, estos puedan reflejarse) si estos van a participar en interrelaciones (aparte de las interrelaciones en las que participe el supertipo).

Por ejemplo, supongamos las siguientes entidades identificadas en un universo del discurso de una empresa: *EMPLEADO* (con identificador *Nº-Emp* y los descriptores *Nombre-Emp*, *Dirección-Familiar*, *Fecha-Nacimiento*, *Descripción-Puesto*, *Salario*, *Experiencia*), *INGENIERO* (con identificador *Nº-Emp* y descriptores *Nombre-Emp*, *Dirección-Familiar*, *Especialidad*), *SECRETARIO* (con identificador *Nº-Emp* y descriptores *Nombre-Emp*, *Fecha-Nacimiento*, *Salario*, *Pulsaciones*) y *TÉCNICO* (con identificador *Nº-Emp* y

descriptores *Nombre-Emp*, *Experiencia*, *Años-de-Experiencia*).

Identificamos que *EMPLEADO* es una generalización de *INGENIERO*, *SECRETARIO* y *TÉCNICO*. Entonces redistribuimos los atributos entre las entidades. Situamos el identificador *Nº-Emp* y los descriptores genéricos *Nombre-Emp*, *Dirección-Familiar*, *Fecha-Nacimiento*, *Descripción-Puesto* y *Salario* en la entidad supertipo *EMPLEADO*, ponemos el descriptor específico *Especialidad* en la entidad *INGENIERO*; ponemos el descriptor específico *Pulsaciones* en la entidad *SECRETARIO* y, por último, ponemos los descriptores específicos *Experiencia* y *Años-Experiencia* en la entidad *TÉCNICO*. La Figura 1.3 muestra el diagrama E/R resultante.

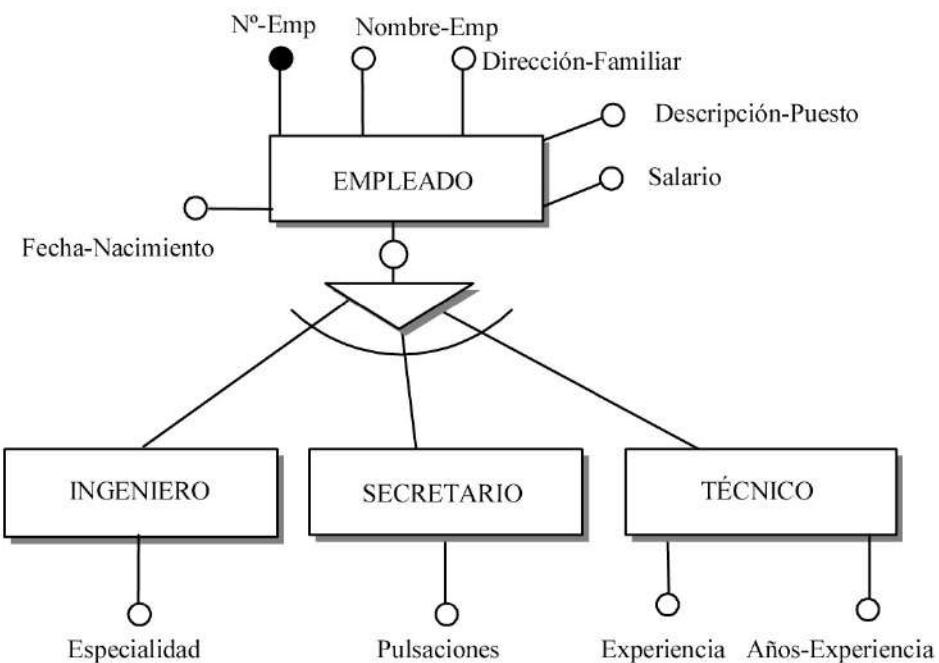


Figura 1.3. Ejemplo de generalización

Existen otras restricciones semánticas relacionadas con las generalizaciones como son la *totalidad/parcialidad* y la *exclusividad/solapamiento*. Si las ocurrencias de los subtipos de una generalización cubren al supertipo (es decir, no hay ocurrencias en el supertipo que no pertenezcan a ninguno de los subtipos) entonces se trata de una generalización total y en caso contrario parcial. Por otro lado, si puede haber ocurrencias que pertenezcan a más de uno de los subtipos entonces se trata de generalizaciones con solapamiento; en caso de que los subtipos sean disjuntos se habla de generalizaciones exclusivas.

En el ejemplo de la Figura 1.3 se muestra una generalización total, es decir,

no existen *EMPLEADOS* que no sean ni *INGENIEROS*, ni *SECRETARIOS* ni *TÉCNICOS*; además, es una generalización exclusiva, es decir, un *EMPLEADO* o es *INGENIERO* o *SECRETARIO* o *TÉCNICO*, pero no puede ser varias cosas a la vez.

Aunque es sencillo definir los constructores de entidad, atributo e interrelación, no es tan sencillo distinguir su papel en el modelado de BD. ¿Qué es lo que hace que un concepto sea un atributo, una entidad o una interrelación?; en este caso únicamente se podrá recurrir también a una serie de heurísticas que nos ayuden a decidir entre uno de los constructores que pueden reflejar la semántica de un determinado concepto. Algunas de ellas se describen a continuación.

1.1.2 Entidades frente aatributos

Los atributos no tienen existencia por sí mismos, sino que tienen sentido en cuanto a que pertenecen a una determinada entidad o interrelación. Una entidad debe estar caracterizada por algo más que su identificador principal. Si existe información descriptiva sobre un concepto u objeto, entonces debería clasificarse como entidad. Si solo se necesita un identificador para un objeto, el objeto debería clasificarse como un atributo. Así,

las entidades poseen información descriptiva y los atributos no.

Por ejemplo, en el supuesto “*los Almacenes se localizan en Ciudades*”. Si existe alguna información descriptiva sobre el Estado y la Población de las Ciudades, entonces Ciudad debería clasificarse como entidad. Si solo se necesita el atributo *Nombre_Ciudad* para identificar una ciudad, entonces debería clasificarse como atributo.

Por otro lado, podría ocurrir que aun teniendo un concepto para el que solo existe un identificador principal, éste se relacione con más de una entidad. En ese caso podría aparecer como una entidad en el esquema E/R. En el ejemplo anterior, si existiera otro supuesto “*los proveedores tienen asignadas varias ciudades*”, entonces podría dejarse Ciudad como entidad relacionada con las entidades *Almacén* y *Proveedor*, o bien dejarse como atributos en ambas entidades.

1.1.3 Entidades frente a atributos multivaluados

En este aspecto existen discrepancias. Hay propuestas que prefieren incorporar en los esquemas E/R un atributo multivaluado como una entidad y otras prefieren representarlo como un atributo. En nuestro caso, con independencia de que el atributo sea simple o compuesto, si se sabe que tendrá un número limitado y no muy elevado de ocurrencias, entonces formará parte de la entidad que describe (siempre y cuando el concepto que representa no esté relacionado con otras entidades del esquema E/R). Un ejemplo puede ser: “*De un empleado interesa almacenar su DNI, nombre, dirección y teléfonos*”; en este caso el atributo *Teléfono* es un atributo multivaluado de la entidad *EMPLEADO*. En el ejemplo “*Un profesor se caracteriza por su nombre, DNI, dirección y los campus en los que imparte docencia*”; en este caso el atributo *Campus* también podría ser un atributo multivaluado de la entidad *PROFESOR* pero si existen supuestos adicionales en el universo del discurso que nos indican información adicional para describir un campus y además se relaciona con otras entidades (por ejemplo, *DEPARTAMENTO*, *AULARIO*, etc.) entonces debe reflejarse como una entidad.

1.1.4 Entidades frente a interrelaciones

Lo habitual es no tener problemas en la diferenciación entre entidades e interrelaciones. Las interrelaciones asocian una o varias entidades, mientras que las entidades no. Sin embargo, en cualquier interrelación puede realizarse un proceso de nominalización. Por ejemplo, mediante una nominalización puede convertirse “*los hombres se casan con mujeres*” en “*los hombres y mujeres forman matrimonios*”. Así, se ha sustentivado una interrelación y al introducir un nuevo concepto, se ha convertido en una entidad.

Aun cuando la nominalización puede resultar útil en un proceso de diseño complejo, en especial para tratar de reducir el grado de una interrelación muy compleja o para encontrar elementos de interés para el sistema que inicialmente no se habían tenido en cuenta, en nuestro caso, evitaremos las nominalizaciones que no se encuentran presentes en los enunciados o que no resultan evidentes en el Universo del Discurso.

1.2 NOTACIONES

A continuación se muestran las convenciones seguidas en la resolución de los problemas para la representación gráfica de los distintos constructores de un diagrama E/R.

- Entidad (fuerte).



- Entidad débil.



- Identificador Principal (IP)³.



- Identificador Alternativo (IA)⁴.



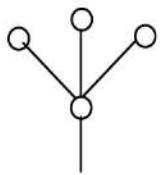
- Atributo.



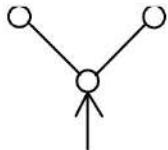
- Atributo Multivaluado.



- Atributo Compuesto.



- Atributo Multivaluado Compuesto.



- Atributo Opcional.



- Atributo Derivado⁵.



- Interrelación.



- Interrelación con Dependencia en Existencia.



- Interrelación con Dependencia en Identificación.



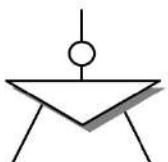
- Dominio.



- Jerarquía Solapada y Parcial (sin ninguna restricción).



- Jerarquía Solapada y Total.



- Jerarquía Exclusiva y Parcial.



- Jerarquía Exclusiva y Total.



1.3 CÓMO SE ESTRUCTURAN LOSPROBLEMAS

A fin de facilitar la comprensión de las soluciones propuestas a los problemas de modelado, para los enunciados más complejos, cada problema se desglosará arbitrariamente en varios pasos. En cada paso se estudiará un conjunto de supuestos semánticos que darán lugar a un subesquema E/R. En cada paso se irán añadiendo elementos al subesquema obtenido en el paso anterior y así sucesivamente hasta completar el estudio de todos los supuestos semánticos contemplados en el enunciado del problema. Supondremos que el enunciado constituye una descripción correcta (y casi siempre completa) del Universo del Discurso. Para los casos prácticos más sencillos se hará la resolución en un único paso.

Una aproximación utilizada habitualmente en la construcción de esquemas E/R es identificar primero las entidades, luego las interrelaciones y por último los atributos de las entidades e interrelaciones.

En nuestro caso se identificarán todos los elementos (entidades, atributos, interrelaciones, cardinalidades, etc.) por cada conjunto de supuestos semánticos analizados. Ello puede implicar que algunos conceptos se representen en los primeros pasos con determinados constructores y que posteriormente, en sucesivos pasos, los nuevos supuestos semánticos analizados nos proporcionen información adicional que modifique alguno de los constructores seleccionados. Por ejemplo, podría ocurrir que un determinado concepto se representara en una primera aproximación como un atributo y que según se avanza en el análisis de los supuestos del enunciado se descubra que debe representarse como una entidad.

A veces también se suele recurrir a otro tipo de herramientas que nos ayudan a detectar información que no aparece explícitamente representada en el enunciado y que resultan de gran utilidad a los diseñadores inexpertos. Así, una propuesta de metodología de realización de un esquema conceptual que tiene en cuenta estos aspectos constaría de los siguientes pasos⁶:

1. Estudiar el enunciado que describe el Universo del Discurso y elaborar dos listas: una con los candidatos a ser entidades y otra con las posibles interrelaciones junto con su tipo de correspondencia (1:1, 1:N, N:M). Además, se especificarán aquellos conceptos dudosos que no se sabe cómo representar (si como entidad o como interrelación).

2. Construir una matriz de entidades en la que las filas y las columnas son nombres de entidades y cada celda puede contener o no nombres de interrelaciones. Esta matriz tiene el siguiente aspecto:

	E1	E2	E3	...	EN
E1	I1	I2	--	...	I3
E2		I4	I5	...	--
E3			I6	...	--
...				...	IN
EN					

Las entidades son *E1, E2,..., EN* y las interrelaciones son *I1, I2, ..., IN*. Como la matriz es simétrica, las celdas que aparecen con una cruz se corresponden con interrelaciones que ya están especificadas en la otra mitad de la matriz. El símbolo -- en una celda indica que no existe interrelación entre las dos entidades referenciadas. Además, habría que indicar los tipos de correspondencia de cada interrelación. Por ejemplo, **I1** podría ser una interrelación 1:N. Es importante destacar que esta matriz no recoge las interrelaciones de grado superior ados.

En la elaboración de esta matriz es posible detectar interrelaciones que no aparecen explícitamente representadas en el enunciado y que, sin embargo, podría resultar interesante que se recogieran en el esquema E/R. Este tipo de interrelaciones se detectan, generalmente, por sentido común, aunque sería necesario siempre validarlas con el usuario.

3. Utilizando la matriz de entidades se construye un primer esquema E/R con las entidades, atributos, interrelaciones y sus tipos de correspondencia. A este esquema se le añaden las cardinalidades mínimas y máximas.
4. No toda la semántica del enunciado de los problemas se podrá representar en el diagrama E/R, por lo que se incluirá un apartado de supuestos semánticos no recogidos en el diagrama E/R. El esquema E/R final se compone por el diagrama E/R junto al apartado de supuestos semánticos no recogidos en el diagrama.

5. En este último paso se refina el esquema E/R del paso anterior estudiando las posibles redundancias siempre y cuando existan ciclos con interrelaciones semánticamente equivalentes. Existe redundancia en un esquema E/R cuando la misma semántica se recoge de manera duplicada, por lo que ese esquema podría representarse manteniendo la misma semántica con menos elementos.

En general, puede haber redundancia cuando existen ciclos en un esquema E/R (varias entidades unidas por varias interrelaciones relacionadas semánticamente formando un ciclo). En este caso, habría que comprobar si eliminando una interrelación, la semántica representada en ella puede obtenerse mediante las interrelaciones restantes. Para ello, hay que estudiar detalladamente las cardinalidades de las interrelaciones y hacer la comprobación tanto en un sentido como en el otro.

PROBLEMA 1.1: GESTIÓN DEALQUILERES

La Figura 1.4 corresponde a una parte del diseño de una BD que la Sociedad Pública de Alquiler dependiente del Ministerio de la Vivienda necesita para gestionar los alquileres de viviendas de particulares y favorecer así el alquiler entre la población.

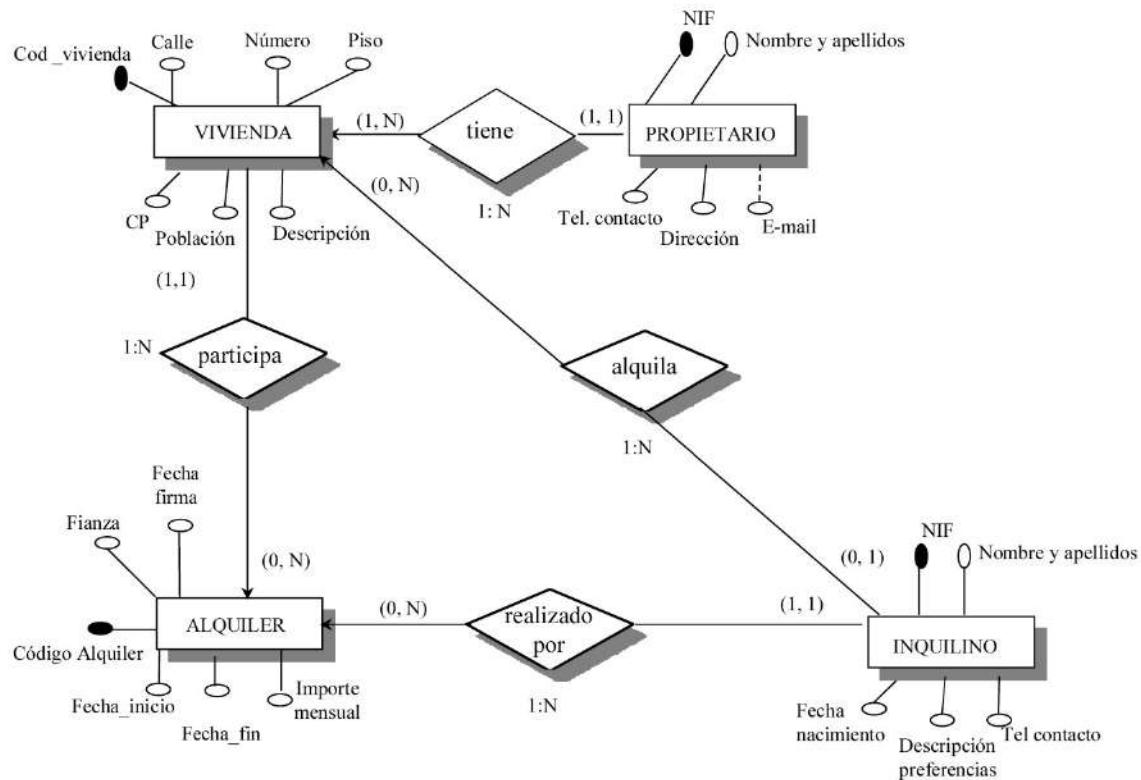


Figura 1.4. Diagrama E/R inicial sobre gestión alquileres

El diagrama E/R de la figura contiene información sobre los propietarios que ofrecen sus viviendas en alquiler y los inquilinos que están interesados en alquilar una. Se guarda información sobre los alquileres (código alquiler, fecha firma y fianza) que se llevan a cabo y las viviendas a las que corresponden.

Se pide:

1. Comprobar si en el diagrama E/R de la Figura 1.4 existe alguna redundancia. Si es así, modificar el esquema para eliminarla.
2. Se pide extender el esquema E/R de la Figura 1.4 para recoger los

siguientes supuestos semánticos (indicar aquellos supuestos que no hayan podido reflejarse en la solución propuesta):

- Se quiere incluir información sobre la duración de cada uno de los alquileres de una vivienda con el fin de manejar el histórico de alquileres de la BD. Así mismo, se quiere almacenar información sobre la renovación de un alquiler de una vivienda, es decir, saber si un determinado alquiler es renovación de otro o no (para un mismo inquilino) para poder seguir fácilmente la secuencia de alquileres de una vivienda con la misma persona.
- Se quiere incluir información sobre las agencias inmobiliarias (código de agencia, dirección y CIF) con las que la Sociedad Pública de Alquiler tiene convenios para gestionar los alquileres, de tal manera que cada alquiler es gestionado por una única agencia. Además, cada vivienda se oferta en una única agencia inmobiliaria y se quiere guardar información al respecto. Una agencia solo puede gestionar los alquileres de viviendas ofertadas por ella.

DISCUSIÓN DEL ENUNCIADO

PARTE 1

Comprobar si en el diagrama E/R de la Figura 1.4 existe alguna redundancia. Si es así, modificar el esquema para eliminarla.

En principio podrían existir atributos redundantes o interrelaciones redundantes. Para comprobar que en el diagrama de la Figura 1.4 no existen atributos redundantes se mostrará que la semántica asociada a cada atributo en el diagrama no se puede obtener a partir de otro atributo o conjunto de ellos. En este problema no existen atributos redundantes.

Para comprobar que en el diagrama de la Figura 1.4 no existen interrelaciones redundantes se tratará de justificar que la semántica asociada a cada interrelación en el ciclo *Alquila-Participa-Realizado_por* no puede ser reemplazada por ninguna otra interrelación o conjunto de ellas. En este caso sí existe una interrelación redundante en la Figura 1.4: la interrelación **Alquila** es redundante pues su semántica se puede obtener a partir de **Participa** y **Realizado_por**. La interrelación **Alquila** almacena información sobre qué individuo tiene actualmente la vivienda alquilada (en el caso de que se encuentre alquilada). Por otro lado, en la entidad *ALQUILER* junto con las interrelaciones **Participa** y **Realizado_por** se almacena información histórica sobre qué alquileres han existido sobre cada vivienda en la Sociedad Pública de Alquiler y quiénes han firmado ese alquiler como inquilinos. Gracias a esta información histórica también se puede obtener la información sobre qué inquilino tiene actualmente alquilada una determinada vivienda, semántica que representaba la interrelación **Participa**.

Si se mantuviera la interrelación **Alquila** en el esquema E/R sería necesario tener en cuenta que los datos de esta interrelación tendrían que ser consistentes con los datos de las interrelación **Participa** y **Realizado_por**. Es decir, hay que comprobar en todo momento que el inquilino actual de una determinada vivienda (en la interrelación **Alquila**) es el mismo inquilino que participa en la interrelación **Realizado_por** con la entidad *ALQUILER* de la misma vivienda, donde la fecha actual se encuentra entre los valores de *Fecha_inicio* y *Fecha_fin*. Por ejemplo, si se modificara algún dato en las fechas de la entidad *ALQUILER* o en las interrelaciones **Participa** o **Realizado_por**, se tendría

que comprobar la coherencia semántica con la interrelación **Alquila**.

¿Se podría considerar como redundante la interrelación **Participa** o la interrelación **Realizado_por**? La interrelación **Participa** nunca se podrá considerar como redundante pues, de no existir, no se podría recoger información histórica sobre qué viviendas han sido alquiladas o están en disposición de ser alquiladas al haber finalizado su contrato de alquiler. De la misma forma, si se eliminara la interrelación **Realizado_por**, se perdería la semántica asociada al responsable o firmante del alquiler.

La Figura 1.5 muestra el diagrama E/R del problema propuesto una vez eliminada la redundancia.

Supongamos por un momento que en la Figura 1.4, en lugar de representarse la interrelación **Alquila** con la información sobre el inquilino actual de cada vivienda, se representara una interrelación **Visita** entre las entidades *INQUILINO* y *VIVIENDA*. La interrelación **Visita** almacenaría información sobre todas las personas que han visitado una determinada vivienda, pudiendo decidir posteriormente si la alquilaban o no. La Figura 1.6 muestra el diagrama E/R de este supuesto. En este caso, ninguna de las tres interrelaciones (**Visita**, **Participa** y **Realizado por**) serían redundantes, ya que la semántica de la interrelación **Visita** no se manifiesta en el resto del diagrama, que trata de los alquileres que se han llevado a cabo (no de las visitas a las viviendas). Al igual que ocurría en el diagrama de la Figura 1.4, las interrelaciones **Participa** y **Realizado_por** tampoco son interrelaciones redundantes.

 Cuando existe un ciclo provocado por interrelaciones relacionadas semánticamente en el diagrama E/R es posible que existan interrelaciones redundantes (habrá que comprobar la redundancia en las interrelaciones), aunque no siempre su semántica se puede obtener a partir de otra información en el diagrama.

 En algunos casos se desea conservar en el diagrama E/R atributos redundantes (que deben ser marcados como atributos derivados) o interrelaciones redundantes. La información redundante en los diagramas E/R implicará complicaciones en el mantenimiento de los datos en la base de datos, pero en algunos casos los usuarios finales de la misma prefieren que existan datos redundantes típicamente justificados para hacer más eficiente el acceso a

ciertos datos consultados frecuentemente. Para mantener información redundante en un esquema E/R ha de existir una razón bien justificada.

PARTE 2

Se quiere incluir información sobre la duración de cada uno de los alquileres de una vivienda con el fin de manejar el histórico de alquileres de la BD. Así mismo, se quiere almacenar información sobre la renovación de un alquiler de una vivienda, es decir, saber si un determinado alquiler es renovación de otro o no (para un mismo inquilino) para poder seguir fácilmente la secuencia de alquileres de una vivienda con la misma persona.

En este caso, la información sobre la duración de cada alquiler ya está representada en el diagrama de la Figura 1.4, ya que la duración del alquiler vendrá dada por la diferencia entre la fecha de fin del alquiler (*Fecha_fin*) y la fecha de inicio del alquiler (*Fecha_inicio*). Por ello, no será necesario incluir un nuevo atributo en la entidad *ALQUILER* que represente la duración del mismo, a no ser que el usuario lo necesite por motivos de eficiencia en el acceso a los datos. En este caso, el atributo *duración* se representará como un atributo derivado en la entidad *ALQUILER*.

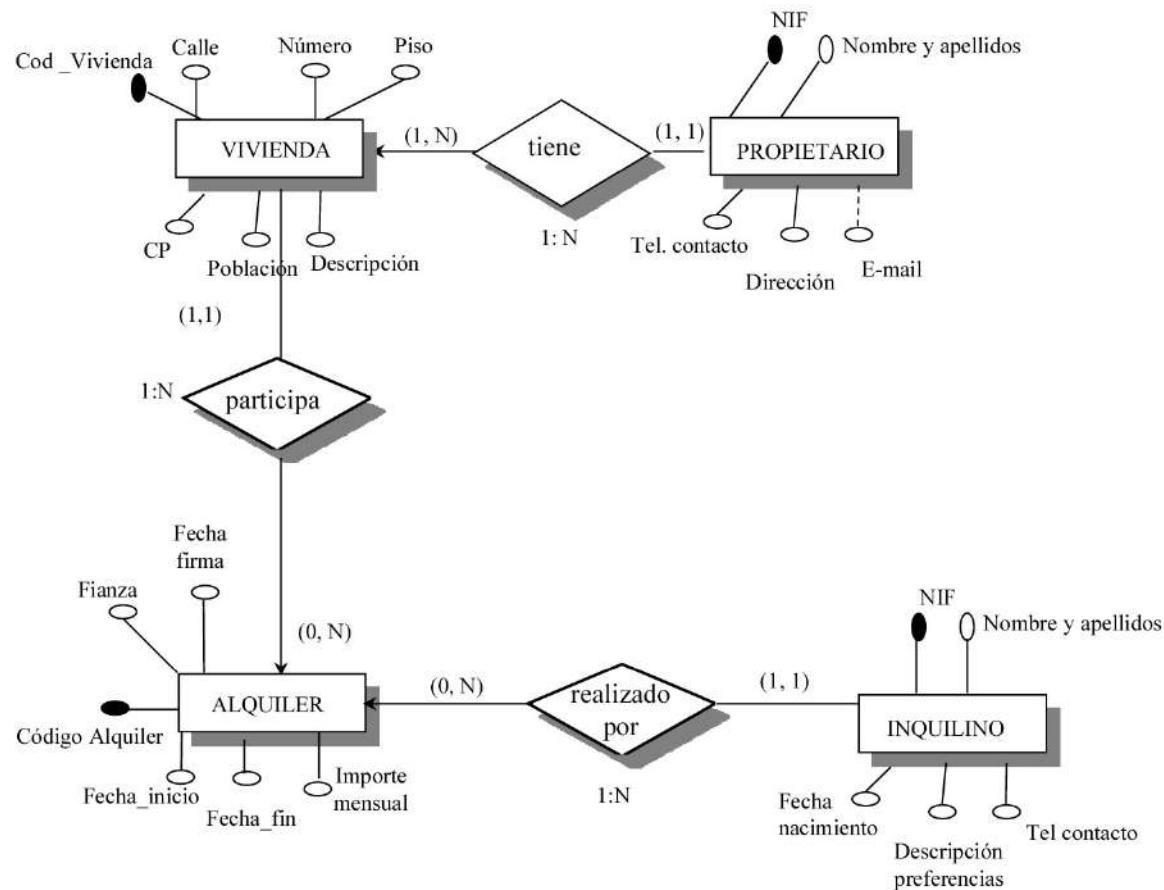


Figura 1.5. Diagrama E/R inicial sobre gestión de alquileres sin redundancias

Por otro lado, para almacenar información sobre qué contrato de alquiler es renovación de qué otro alquiler se ha de incluir una interrelación reflexiva sobre la entidad **ALQUILER**. Esta interrelación, a la que hemos denominado **Renovación**, representa la semántica de que un alquiler es renovación de ningún o un alquiler y al mismo tiempo un alquiler es renovado por o bien ningún o un alquiler como máximo (cardinalidad mínima igual a cero y cardinalidad máxima igual a uno en ambos casos). En la Figura 1.7 se muestra la representación de la interrelación reflexiva **Renovación**.

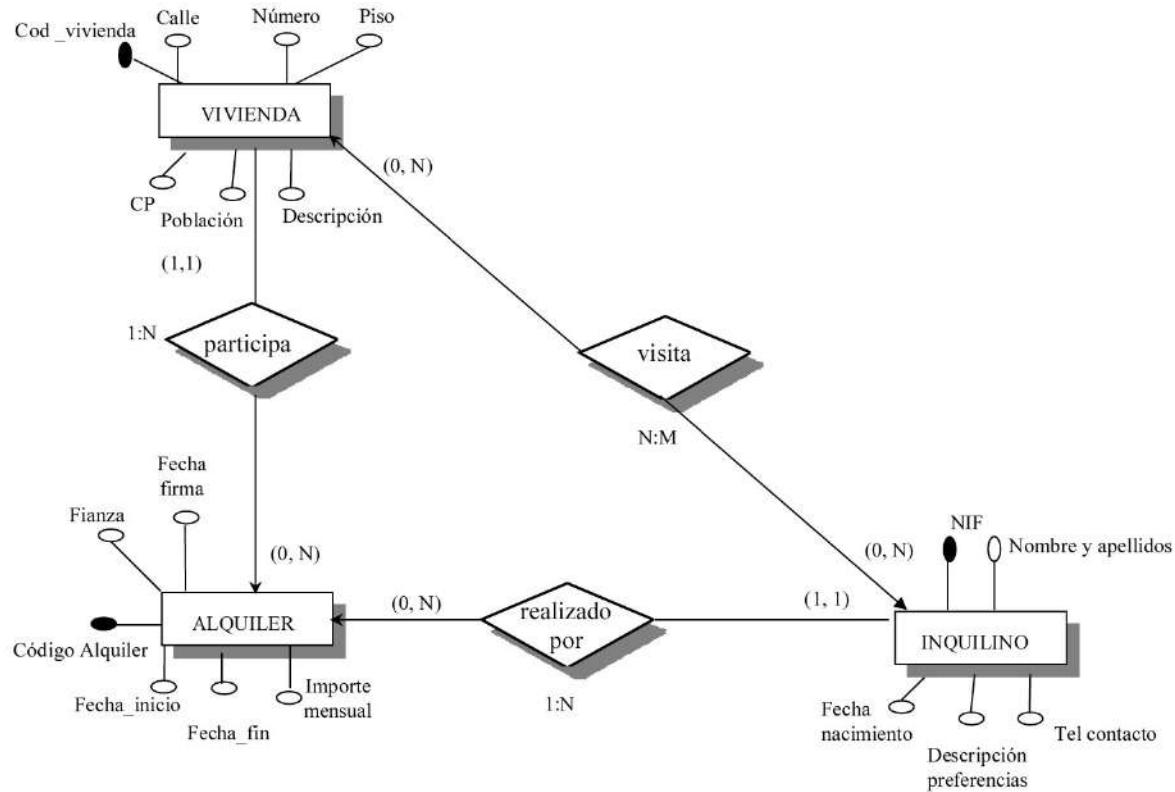


Figura 1.6. Diagrama E/R de ejemplo con un ciclo de interrelaciones sin redundancia

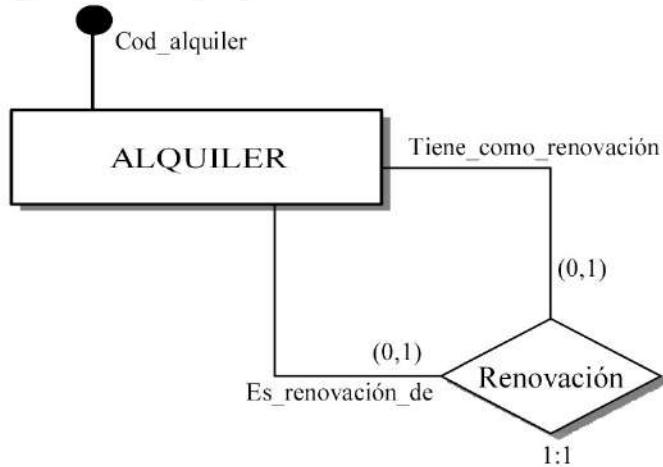


Figura 1.7. Diagrama E/R correspondiente a la interrelación Renovación

Para completar el esquema conceptual es necesario que añadamos información sobre los supuestos semánticos del universo del discurso (implícitos o explícitos en el enunciado del problema) que ha sido imposible incorporar en el diagrama E/R.

SUPUESTOS SEMÁNTICOS NO RECOGIDOS EN EL

DIAGRAMA E/R

En los diagramas anteriores no se ha podido representar la siguiente información semántica:

La fecha de firma (*Fecha_firma*) de un determinado alquiler ha de ser anterior o igual en el tiempo que la fecha de inicio (*Fecha_inicio*) del alquiler. Al mismotiempo, la fecha de fin (*Fecha_fin*) del alquiler ha de ser igual o posterior en el tiempo a la fecha de inicio del alquiler.

Habrá que tener en cuenta de igual manera que la fecha de nacimiento (*Fecha_nacimiento*) de un inquilino nunca podrá ser posterior en el tiempo a la fecha de firma (*Fecha_firma*) de la vivienda que está alquilando. Además, si se tuviera en cuenta que un inquilino no podrá alquilar una vivienda si es menor de edad, habrá que comprobar que: *INQUILINO.Fecha_nacimiento+18años <= ALQUILER.Fecha_firma* para todas las viviendas alquiladas por el mismo inquilino.

No pueden existir dos contratos de alquiler (ocurrencias en la entidad *alquiler*) sobre la misma vivienda cuyos períodos de renta se solapen.

De igual forma, se tendrá que controlar que un alquiler no puede nunca ser renovación de sí mismo ni de otros alquileres que han sido a su vez (en uno o varios grados) renovación delmismo.

Por último, ha sido imposible incorporar información en la Figura 1.7 sobre las fechas de renovación. Se ha de cumplir en todo momento que un contrato de alquiler sobre una determinada vivienda no puede ser renovado en una fecha anterior a la finalización del contrato anterior. Es decir, *ALQUILERvivienda_i.Fecha_fin <= ALQUILERrenovación_alquiler_vivienda_i.Fecha_ini*.

Por ser el primer ejercicio sobre esquemas E/R en este libro se comentarán también supuestos semánticos a tener en cuenta a la hora de implementar la base de datos y que habitualmente no se presentan explícitamente en la solución de estos ejercicios por suponerse obvios. Son también supuestos semánticos asociados a los diagramas E/R anteriores los siguientes:

- La *Fianza* y el *Importe Mensual* de la entidad *ALQUILER* han de ser números enteros mayores o iguales acero.
- Los valores de los atributos *NIF* y *Tel_contacto* de la entidad

INQUILINO han de cumplir un determinado formato. Por ejemplo, el NIF ha de ser una cadena de caracteres formada por un número de 8 dígitos, seguido por un guión y por una letra (ejemplo “02348756-K”). También han de comprobarse los formatos de los atributos *NIF*, *E-mail*, *Tel_contacto* de la entidad *PROPIETARIO* y el código postal (CP), el atributo *Piso* y el atributo *Número* de la entidad *VIVIENDA*.

PARTE 3

Se quiere incluir información sobre las agencias inmobiliarias (código de agencia, dirección y CIF) con las que la Sociedad Pública de Alquiler tiene convenios para gestionar los alquileres, de tal manera que cada alquiler es gestionado por una única agencia. Además, cada vivienda se oferta en una única agencia inmobiliaria y se quiere guardar información al respecto. Una agencia solo puede gestionar los alquileres de viviendas ofertadas por ella.

En este caso será necesario añadir una entidad *AGENCIA* identificada por su código de agencia (atributo identificador principal), pero sin olvidar que el CIF de la agencia también podría identificar a la ocurrencia (atributo identificador alternativo). La dirección de la agencia inmobiliaria se representará como un atributo univaluado, obligatorio y no identificativo de la entidad.

Además, también se ha de añadir una interrelación **Oferta** en entre las entidades *AGENCIA* y *VIVIENDA*. Esta interrelación guardará información sobre qué agencia inmobiliaria es la encargada de gestionar los alquileres de esa vivienda. La Figura 1.8 muestra la nueva entidad *AGENCIA* y la nueva interrelación junto con sus cardinalidades, donde se ha tenido en cuenta que una agencia inmobiliaria puede que no se encargue de ninguna vivienda en un momento dado, pero siempre toda vivienda ha de estar gestionada por una y solo una agencia inmobiliaria.

En este apartado podríamos pensar que debería existir una interrelación entre las agencias inmobiliarias y los contratos de alquiler, pero esta interrelación sería redundante, ya que en un contrato de alquiler se alquila una única vivienda y gracias a la interrelación **Oferta** ya se conocería qué agencia inmobiliaria gestiona el contrato de alquiler en cuestión.

La semántica de este apartado se ha podido representar completamente con los constructores del Modelo E/R, por lo que no será necesario añadir ningún

supuesto semántico más a la lista de supuestos semánticos descrita en el apartado anterior.

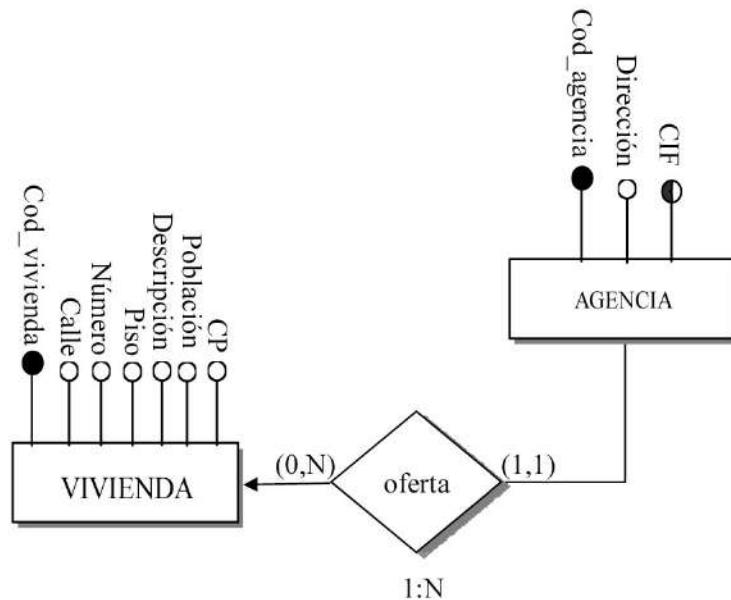


Figura 1.8. Esquema E/R correspondiente a Oferta

SOLUCIÓN PROPUESTA

Como resumen del ejercicio, se presenta el esquema E/R final en la Figura donde no se han podido incorporar los siguientes supuestos semánticos:

SUPUESTOS ASOCIADOS AL DIAGRAMA E/R

- SS1. La fecha de firma (*Fecha_firma*) de un determinado alquiler ha de ser anterior o igual en el tiempo que la fecha de inicio (*Fecha_inicio*) del alquiler. Al mismo tiempo, la fecha de fin (*Fecha_fin*) del alquiler ha de ser igual o posterior en el tiempo a la fecha de inicio del alquiler.
- SS2. La fecha de nacimiento (*Fecha_nacimiento*) de un inquilino nunca podrá ser posterior en el tiempo a la fecha de firma (*Fecha_firma*) de la vivienda que está alquilando. Si se tuviera en cuenta que un inquilino no podrá alquilar una vivienda si es menor de edad, habrá que comprobar que $INQUILINO.Fecha_nacimiento + 18\text{años} \leq ALQUILER.Fecha_firma$ para todas las viviendas alquiladas por el mismo inquilino.
- SS3. No pueden existir dos contratos de alquiler (ocurrencias en la entidad *ALQUILER*) sobre la misma vivienda cuyos períodos de renta se solapen.
- SS4. Un alquiler no puede nunca ser renovación de sí mismo ni de otros alquileres que han sido a su vez (en uno o varios grados) renovación del mismo.
- SS5. Un contrato de alquiler sobre una determinada vivienda no puede ser renovado en una fecha anterior a la finalización del contrato anterior. Es decir, $ALQUILER.vivienda_i.Fecha_fin \leq ALQUILER^{renovación_alquiler_vivienda_i}.Fecha_ini$.
- SS6. La *Fianza* y el *Importe Mensual* de la entidad *ALQUILER* han de ser números enteros mayores o iguales a cero.
- SS7. Los valores de los atributos *NIF* y *Tel_contacto* de la entidad *INQUILINO* han de cumplir un determinado formato. Por ejemplo, el *NIF* ha de ser una cadena de caracteres formada por un número de 8 dígitos, seguido por un guión y por una letra (ejemplo 02348756-K2). También han de comprobarse los formatos de los atributos *NIF*, *E-mail*, *Tel_contacto* de la

entidad *PROPIETARIO* y el código postal (CP), el atributo *Piso* y el atributo *Número* de la entidad *VIVIENDA*.

DIAGRAMA E/R

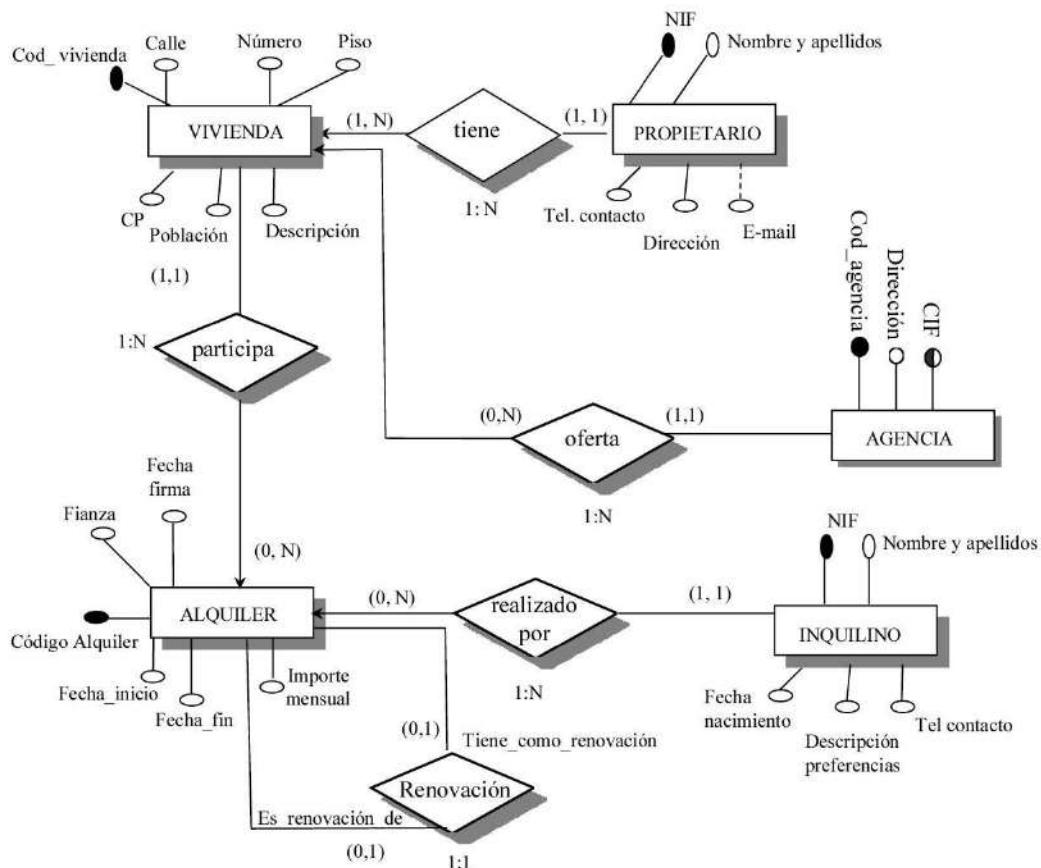


Figura 1.9. Diagrama E/R solución propuesto sobre gestión de alquileres

PROBLEMA 1.2: ADMINISTRACIÓN DE FINCAS

ENUNCIADO

Una firma de abogados dedicada a la administración de fincas desea tener una base de datos para facilitar la gestión de la información de sus clientes, es decir, de las distintas comunidades de vecinos que administra. La información que debe contener la BD concierne a los aspectos que se describen a continuación.

La firma tiene varios abogados y cada uno de ellos ejerce de administrador de una o más comunidades de vecinos, por lo que cobra a cada una de ellas unos honorarios anuales. Una comunidad de vecinos es gestionada por un único administrador (Nombre, DNI y N° de Colegiado). Las funciones de un administrador, sobre las que en este caso interesa guardar información, consisten en llevar la contabilidad de la comunidad, gestionando los recibos que pagan los vecinos mensualmente, así como los pagos a las distintas compañías que proporcionan algún servicio a la comunidad (limpieza, ascensores, seguridad, luz, etc.).

De las empresas que tienen contratadas las distintas comunidades de vecinos (por ejemplo, Iberdrola, Unión Fenosa, OTIS, etc.) se guarda su nombre, CIF, dirección, teléfono y una persona de contacto. Además, interesa tener estas compañías agrupadas en distintos sectores (luz, seguridad, ascensores, etc.).

De cada comunidad de vecinos gestionada por la firma de abogados interesa almacenar un código identificador, su nombre, calle, código postal y población. Cada comunidad consta de una serie de propiedades que pueden ser de tres tipos (vivienda particular, local comercial y oficina). Cada propiedad se caracteriza por un número de portal, planta y letra, un nombre y apellidos del propietario con su dirección completa (que puede ser ésta u otra) y un teléfono de contacto, un porcentaje de participación en los gastos de la comunidad así como los datos de la cuenta bancaria en la que el propietario desea se le domicilie el pago de los recibos.

Si el propietario no habita en su propiedad entonces se necesitan sus datos (nombre, apellidos, dirección y teléfono de contacto) así como los del inquilino que la habita (nombre, apellidos y teléfono de contacto), en caso de que esté

habitada la propiedad. Si el propietario habita en la propiedad solo son necesarios sus datos (nombre, apellidos, teléfono de contacto).

Si la vivienda es particular se guardará el número de habitaciones de que dispone; si es un local comercial se almacenará el tipo de comercio que se desarrolla en él y el horario (en caso de que esté en uso); si es una oficina se guardará la actividad a la que se destina.

Cada comunidad de vecinos tiene además un presidente y varios vocales (nombre, apellidos y propiedad de la que son dueños) elegidos entre todos los propietarios, que se encargan de tratar directamente con el administrador los distintos problemas que pudieran surgir.

En cuanto a la contabilidad, cada comunidad de vecinos tiene una cuenta en un banco. De los distintos bancos se almacena el código de banco, el nombre y una persona de contacto, mientras que para una cuenta bancaria se guarda un código de cuenta (que consta de un código de sucursal, dos dígitos de control y un número de cuenta) y un saldo. Para identificar una cuenta es necesario añadir al código de cuenta el código del banco en el que se encuentra.

Es necesario almacenar dos tipos de apuntes (ingresos y gastos) para la contabilidad⁷ de cada comunidad de vecinos.

- Por un lado, aunque es el banco el que emite los recibos de las cuotas de comunidad a los distintos propietarios, el administrador guarda información sobre dichos recibos que se ingresan en las cuentas bancarias de las comunidades, es decir, el número de recibo, fecha, importe y si se ha podido cobrar o no. Esta última información es importante para realizar a final de cada trimestre una relación de impagados.
- En cuanto a los apuntes relativos a los gastos se tienen los importes que cobran las empresas contratadas por cada comunidad de vecinos. Las compañías cobran sus recibos (Número de recibo, fecha e importe) cargándolos en la cuenta de cada comunidad.

Se pide:

Realizar el esquema E/R de los supuestos semánticos anteriores, indicando todos aquellos que no pudieron ser recogidos en el diagrama E/R.

DISCUSIÓN DEL ENUNCIADO

PARTE 1

La firma tiene varios abogados y cada uno de ellos ejerce de administrador de una o más comunidades de vecinos por lo que cobra a cada una de ellas unos honorarios anuales. Una comunidad de vecinos es gestionada por un único administrador (Nombre, DNI y N° de Colegiado).

Se detectan como posibles entidades *ADMINISTRADOR* y *COMUNIDAD DE VECINOS*. No es necesaria una entidad *FIRMA* puesto que se trata de gestionar solo la firma en cuestión dedicada a la administración de fincas, es decir, solo habría una ocurrencia de esta entidad. Con el fin de reflejar la semántica de que un abogado ejerce de administrador de una o varias comunidades de vecinos se requiere la interrelación 1:N binaria **Gestiona** cuyas cardinalidades indican que una comunidad de vecinos es gestionada por un único administrador y un administrador puede gestionar o bien ninguna o bien varias comunidades de vecinos.

También se distingue un atributo en la interrelación **Gestiona**, los “Honorarios” que cobra cada administrador a cada comunidad de vecinos que gestiona. Suponemos que solo se desea almacenar los honorarios actuales por cada comunidad (no se trata de un registro histórico). Por último, el identificador principal de la entidad *COMUNIDAD DE VECINOS* puede ser el *Núm_Colegiado*; el resto de atributos de esta entidad serían el *DNI* (que se definirá como identificador alternativo) y el *Nombre*.

Las funciones de un administrador, sobre las que en este caso interesa guardar información, consisten en llevar la contabilidad de la comunidad, gestionando los recibos que pagan los vecinos mensualmente, así como los pagos a las distintas compañías que proporcionan algún servicio a la comunidad (limpieza, ascensores, seguridad, luz, etc.).

En estos supuestos semánticos se mencionan los recibos de la comunidad que pagan los vecinos así como los pagos que se realizan a las compañías que ofrecen distintos servicios a las comunidades. Estos conceptos se detallarán con supuestos adicionales en la Parte 3.

De las empresas que tienen contratadas las distintas comunidades de vecinos (por ejemplo, Iberdrola, Unión Fenosa, OTIS, etc.) se guarda su nombre, CIF, dirección, teléfono y una persona de contacto. Además, interesa tener estas compañías agrupadas en distintos sectores (luz, seguridad, ascensores, etc.).

Este supuesto semántico concierne a las empresas que prestan sus servicios a las distintas comunidades de vecinos. Así, se distingue la entidad **COMPAÑÍA** cuyos atributos son *CIF* (que es el identificador principal de la entidad), el *Nombre*, *Dirección*, *Persona de Contacto*, *Teléfono* y *Sector* al que pertenece. *Sector* es un atributo univaluado de la entidad **COMPAÑÍA**, puesto que suponemos que una compañía pertenece a un único sector. Los atributos *Dirección* y *Teléfono* se pueden considerar como atributos identificadores alternativos si se supusiera que no existen dos compañías con la misma dirección y el mismo teléfono. En la solución propuesta para este problema no se ha realizado este supuesto.

También aparece la interrelación **Contrata** entre las entidades **COMUNIDAD DE VECINOS** y **COMPAÑÍA** con cardinalidades (1,N) a cada lado, de tal manera que una comunidad de vecinos puede tener contratadas varias compañías (puede que no tenga ninguna al principio) y una compañía puede ser contratada por una o varias comunidades de vecinos. Suponemos que en mi base de datos solo se almacenan las compañías que prestan servicio a alguna de las comunidades.

El diagrama E/R correspondiente a esta primera parte se muestra en la Figura 1.10

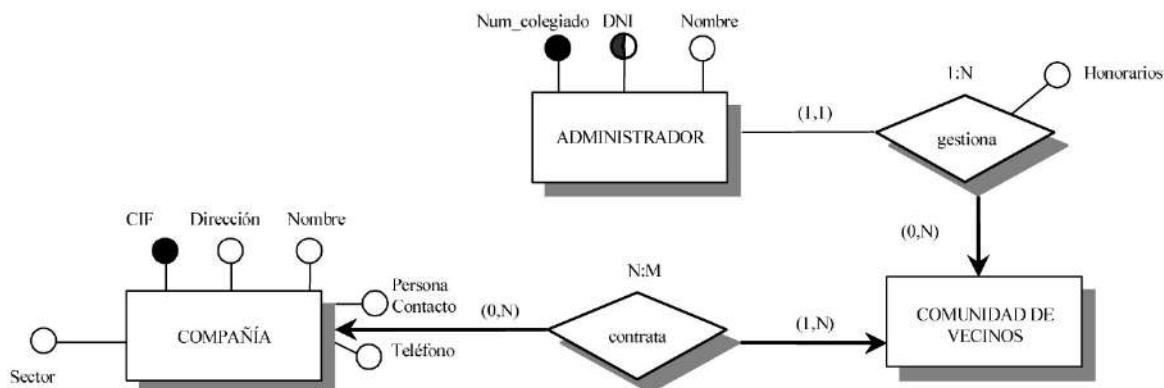


Figura 1.10. Diagrama E/R correspondiente a la Parte 1

PARTE 2

De cada comunidad de vecinos gestionada por la firma de abogados interesa almacenar un código identificador, su nombre, calle, código postal y población.

En este párrafo se indica que los atributos de la entidad *COMUNIDAD DE VECINOS* son *Código de Comunidad* (como identificador principal de la entidad), *Nombre, Calle, Código Postal* y *Población*.

Cada comunidad consta de una serie de propiedades que pueden ser de tres tipos (vivienda particular, local comercial y oficina). Cada propiedad se caracteriza por un número de portal, planta y letra, un nombre y apellidos del propietario con su dirección completa (que puede ser ésta u otra) y un teléfono de contacto, un porcentaje de participación en los gastos de la comunidad así como los datos de la cuenta bancaria en la que el propietario desea se le domicilie el pago de los recibos.

En estos supuestos semánticos se indica que existe la entidad denominada *PROPIEDAD* que engloba las viviendas particulares, los locales comerciales y las oficinas que forman cada comunidad de vecinos. Identificamos, además, la interrelación **Consta** entre *COMUNIDAD DE VECINOS* y *PROPIEDAD* de tal forma que una propiedad pertenece a una única comunidad y una comunidad consta de una propiedad comomínimo.

Los atributos de la entidad *PROPIEDAD* son *Portal, Planta, Letra, Porcentaje* (que representa la participación en los gastos de la comunidad que no es igual para todas las propiedades, pues depende de su tamaño) y *Nº Cuenta*. La entidad *PROPIEDAD* es una entidad débil pues cada propiedad requiere para su identificación el *Código de Comunidad* a la que pertenece, es decir, el portal, planta y letra de una determinada propiedad pueden repetirse en distintas comunidades de vecinos, por lo que es necesario añadir el *Código de Comunidad* a estos atributos para identificar una propiedad.

Para reflejar los datos del propietario se necesita estudiar el supuesto semántico mostrado a continuación.

Si el propietario no habita en su propiedad entonces se necesitan sus datos (nombre, apellidos, dirección y teléfono de contacto) así como los del inquilino que la habita (nombre, apellidos y teléfono de contacto), en caso de que esté habitada la propiedad. Si el propietario habita en la propiedad

solo son necesarios sus datos (nombre, apellidos, teléfono de contacto).

Modelaremos los datos de propietario e inquilino de una propiedad como atributos compuestos. Como una propiedad siempre tiene un propietario, se necesita un atributo obligatorio formado por *Nombre Propietario* (que contiene el nombre y apellidos del propietario), *Teléfono de Contacto* y *Dirección*. El atributo *Dirección* es a su vez un atributo opcional que no tendrá valor si el propietario habita en su propiedad; tendrá un valor si el propietario habita en otra vivienda. El supuesto semántico de cuándo este atributo ha de poseer valores nulos no ha sido posible recogerlo en el diagrama E/R.

En cuanto a los datos del inquilino, utilizaremos un atributo compuesto opcional (pues puede que la propiedad no tenga inquilino) formado por los atributos *Nombre Inquilino* (que contiene el nombre y apellidos del inquilino) y *Teléfono de Contacto*.

Si la vivienda es particular se guardará el número de habitaciones de que dispone; si es un local comercial se almacenará el tipo de comercio que se desarrolla en él y el horario (en caso de que esté en uso); si es una oficina se guardará la actividad a la que se destina.

Este supuesto concierne a los tipos de propiedades sobre los que se desea almacenar información. Se distinguen tres tipos de propiedades (vivienda particular, local comercial y oficina) cada una con atributos específicos. Por ello, se requieren tres tipos de entidades para cada tipo de propiedad que formarán una jerarquía con el supertipo *PROPIEDAD*. La entidad *VIVIENDA PARTICULAR* tiene el atributo *Número de Habitaciones*, la entidad *OFICINA* tiene el atributo *Actividad* y la entidad *LOCAL COMERCIAL* tiene los atributos opcionales *Tipo Comercio* y *Horario*, teniendo valores nulos si el local no está en uso.

Cada comunidad de vecinos tiene además un presidente y varios vocales (nombre, apellidos y propiedad de la que son dueños) elegidos entre todos los propietarios, que se encargan de tratar directamente con el administrador los distintos problemas que pudieran surgir.

Se modelarán los presidentes y vocales de las comunidades de vecinos como interrelaciones. Así, existe la interrelación **Presidente** entre *COMUNIDAD DE VECINOS* y *PROPIEDAD* de tal forma que una comunidad de vecinos tiene una única propiedad cuyo propietario es presidente y para una

determinada propiedad puede que su propietario sea o no presidente de una comunidad de vecinos. También existe la interrelación **Vocal** cuyas cardinalidades indican que una comunidad de vecinos tiene varios vocales (como mínimo uno) y que una determinada propiedad puede que sea no vocal de una comunidad de vecinos. Además, las interrelaciones **Presidente** y **Vocal** son exclusivas pues si una propiedad figura como presidente no puede figurar como vocal y viceversa.

El diagrama E/R correspondiente a esta segunda parte se muestra en la Figura 1.11.

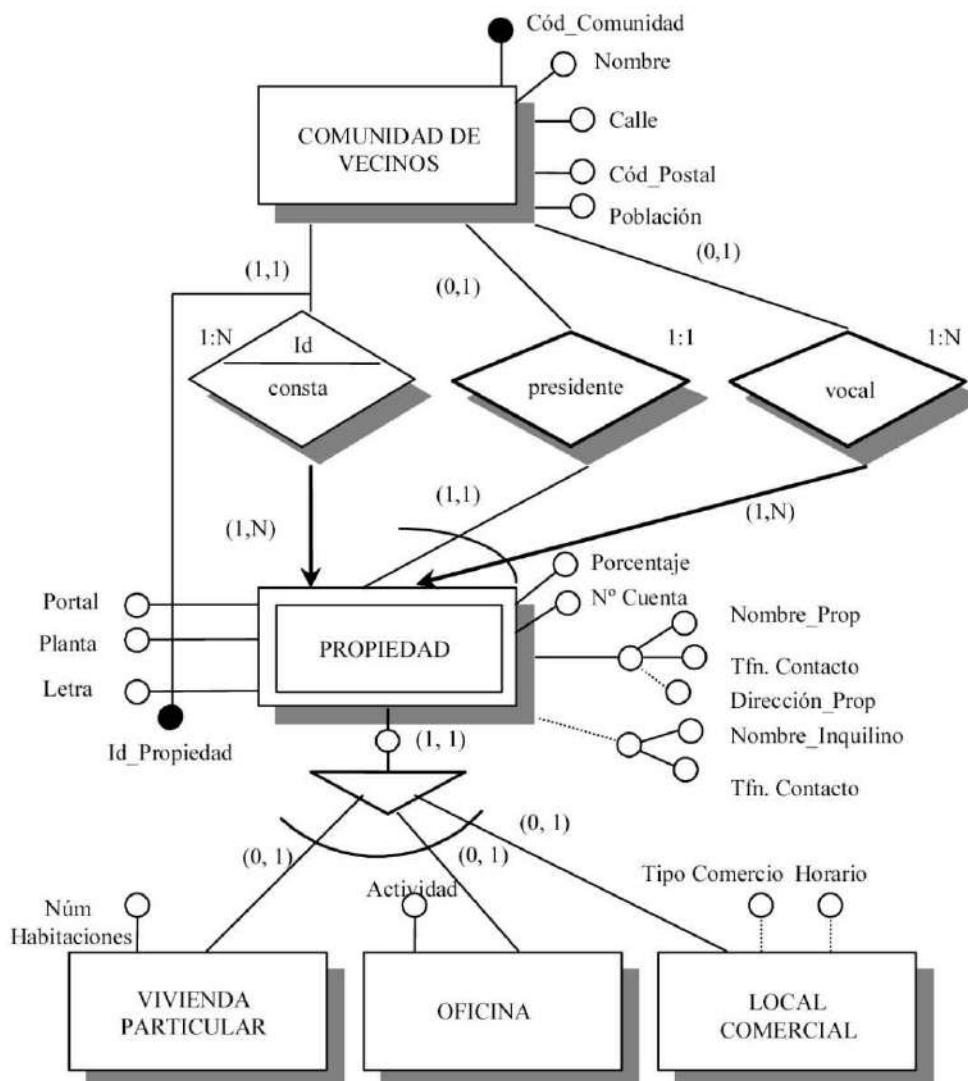


Figura 1.11. Diagrama E/R correspondiente a la Parte 2

PARTE 3

En cuanto a la contabilidad, cada comunidad de vecinos tiene una cuenta en un banco. De los distintos bancos se almacena el código de banco, el nombre y una persona de contacto, mientras que para una cuenta bancaria se guarda un código de cuenta (que consta de un código de sucursal, dos dígitos de control y un número de cuenta) y un saldo. Para identificar una cuenta es necesario añadir al código de cuenta el código del banco en el que se encuentra.

En este supuesto semántico se indica que se necesitan las entidades *BANCO* y *CUENTA*. La entidad *BANCO* tiene los atributos *Código Banco, Nombre* y *Persona de Contacto*. La entidad *CUENTA* es una entidad débil pues una cuenta necesita el código de banco en que se encuentra para su identificación; por ello, el identificador principal de *CUENTA* es la concatenación de los atributos *Sucursal, DC* (dígito de control), *Número* y *Código Banco*. Además, la entidad *CUENTA* también contiene el atributo *Saldo*.

Para conocer qué cuenta tiene cada comunidad de vecinos se necesita la interrelación **Controlada** entre *COMUNIDAD DE VECINOS* y *CUENTA* cuyas cardinalidades denotan que una comunidad tiene una única cuenta asociada y que una determinada cuenta corresponde a una única comunidad de vecinos.

Es necesario almacenar dos tipos de apuntes (ingresos y gastos) para la contabilidad de cada comunidad de vecinos. Por un lado, aunque es el banco el que emite los recibos de las cuotas de comunidad a los distintos propietarios, el administrador guarda información sobre dichos recibos que se ingresan en las cuentas bancarias de las comunidades, es decir, el número de recibo, fecha, importe y si se ha podido cobrar o no. Esta última información es importante para realizar a final de cada trimestre una relación de impagados.

En cuanto a la gestión de los recibos, en este párrafo se distingue la entidad *RECIBO CUOTA COMUNIDAD* correspondiente a las cantías que pagan los propietarios de las comunidades de vecinos; esta entidad tiene los atributos *Número Recibo* (como identificador principal), *Fecha, Importe* y *Estado* (para saber si se ha cobrado o no). Para saber a qué propiedad pertenece cada recibo se necesita la interrelación **Corresponde** entre *PROPIEDAD* y *RECIBO CUOTA COMUNIDAD* cuyas cardinalidades indican que un determinado recibo corresponde a una única propiedad y que a una propiedad le corresponden varios recibos.

Podría pensarse que es necesaria una interrelación **Se Ingresa** entre las entidades *CUENTA* y *RECIBO CUOTA COMUNIDAD*, pero esta interrelación es redundante puesto que esta información puede obtenerse a través de las interrelaciones **Corresponde** y **Controlada**.

En cuanto a los apuntes relativos a los gastos se tienen los importes que cobran las empresas contratadas por cada comunidad de vecinos. Las compañías cobran sus recibos(Número de recibo, fecha e importe) cargándolos en la cuenta de cada comunidad.

En relación con los gastos de las comunidades, se necesita la entidad *RECIBO COMPAÑÍA* cuyos atributos son *Número Recibo* (identificador principal), *Fecha e Importe*. Para saber a qué compañía pertenece cada recibo se requiere la interrelación **Emite** entre *COMPAÑÍA* y *RECIBO COMPAÑÍA* de tal manera que un recibo corresponde a una única compañía y que una compañía puede emitir varios recibos. Por último, la interrelación **Se Carga** entre *RECIBO COMPAÑÍA* y *CUENTA* denota en qué cuentas se cargan los distintos recibos que emiten las empresas (un recibo se carga en una única cuenta y en una cuenta se pueden cargar varios recibos).

En el diagrama E/R no se ha podido representar el supuesto semántico de que nunca se cargará a cuenta de la comunidad de vecinos un recibo que no pertenezca a una compañía contratada por la comunidad. En el ciclo formado por las interrelaciones **Contrata**, **Emite**, **Se carga** y **Controlada** ninguna interrelación se puede considerar redundante. Por ejemplo, no se puede considerar como redundante **Contrata** porque almacena información de las compañías contratadas por la comunidad de vecinos que todavía no han emitido ningún recibo. Se deja como ejercicio al estudiante justificar la no-redundancia del resto de las interrelaciones de este ciclo.

El diagrama E/R correspondiente a esta parte se muestra en la Figura 1.12.

El diagrama E/R correspondiente a esta parte se muestra en la Figura 1.12.

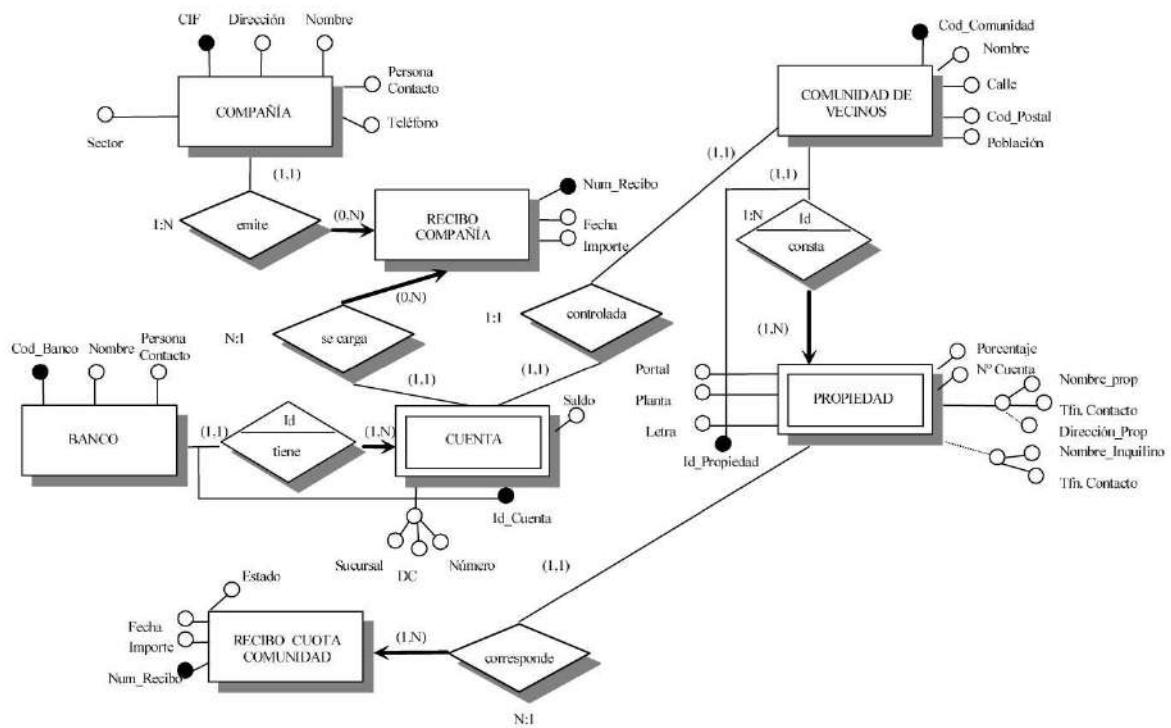


Figura 1.12. Esquema E/R correspondiente a la Parte 3

SOLUCIÓN PROPUESTA

DIAGRAMA E/R

El diagrama E/R final se muestra en la Figura 1.13.

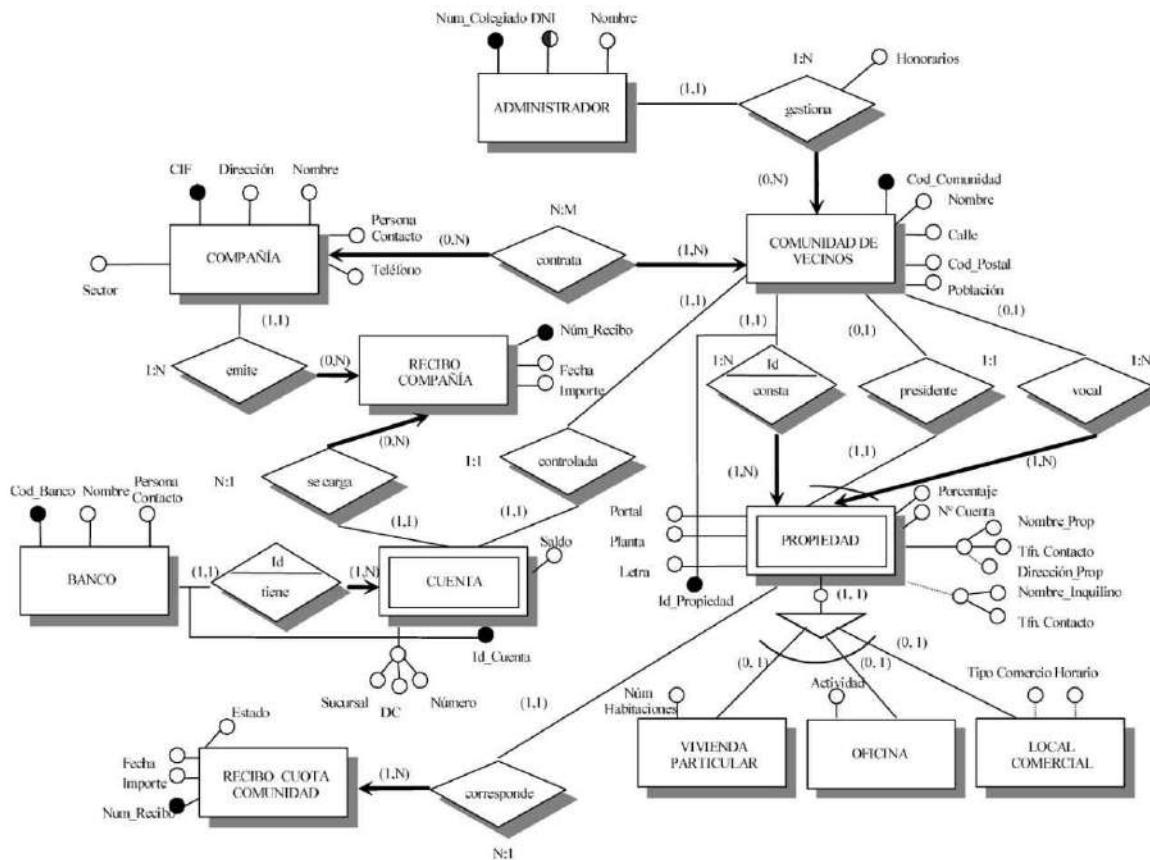


Figura 1.13. Diagrama E/R solución propuesto para comunidad de vecinos

Existen otras posibilidades de modelado de algunos supuestos semánticos de este caso. Por ejemplo, en las entidades *RECIBOCOMPAÑÍA* y *RECIBO CUOTA COMUNIDAD* se observa que existen tres atributos comunes (*Número Recibo*, *Fecha* e *Importe*); esto nos indica que sería posible definir una jerarquía total y exclusiva cuyo supertipo fuese la entidad *RECIBO* y cuyos subtipos fuesen *RECIBO COMPAÑÍA* y *RECIBO CUOTA COMUNIDAD*, como muestra la Figura La entidad *RECIBO COMPAÑÍA* no tendría atributos propios y la entidad *RECIBO CUOTA COMUNIDAD* contendría el atributo *Estado*. Estas entidades subtipo mantendrían las mismas interrelaciones que las definidas en el

diagrama E/R propuesto como solución, como se observa en la Figura1.14.

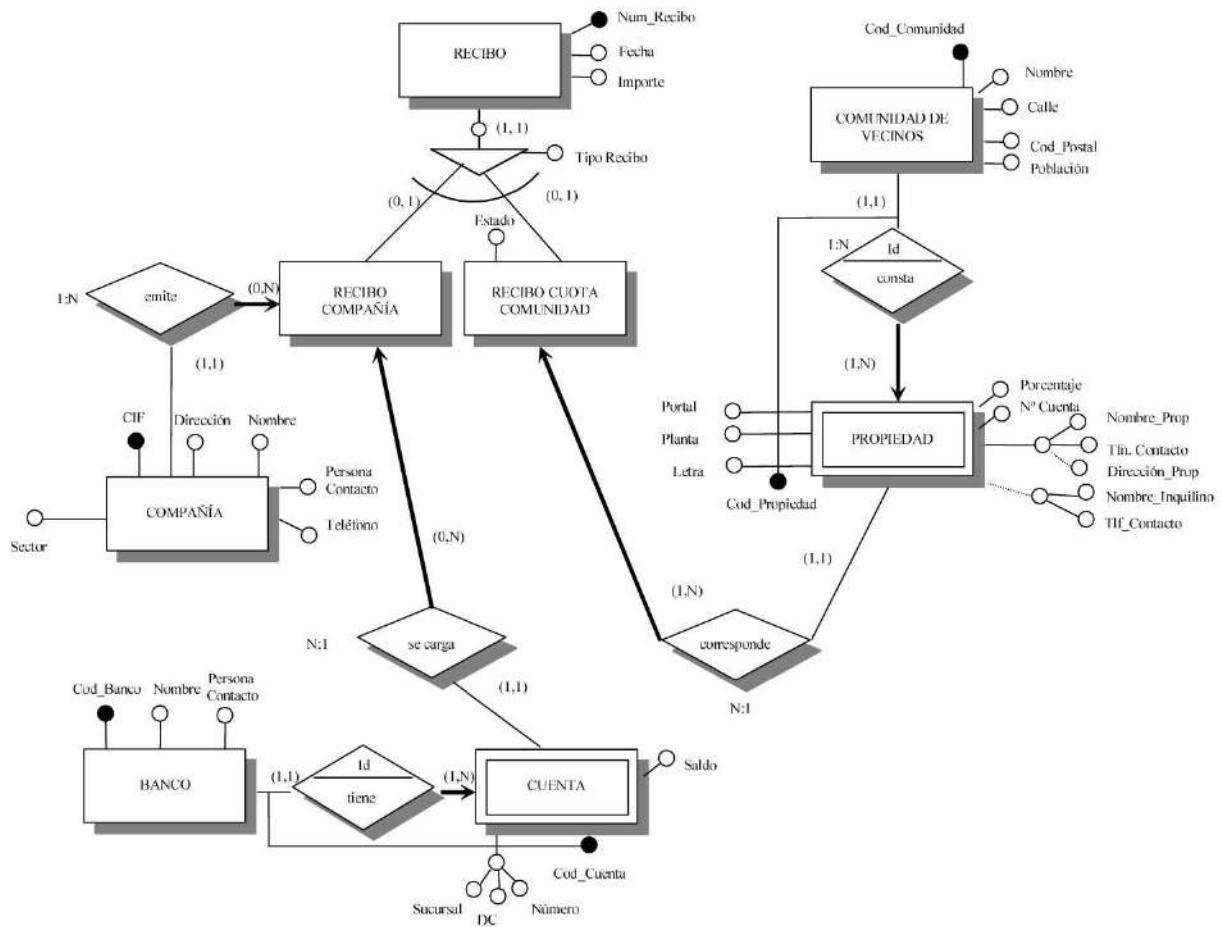


Figura 1.14. Diagrama E/R alternativo considerando una jerarquía en la entidad RECIBO

SUPUESTOS SEMÁNTICOS NO REFLEJADOS EN EL DIAGRAMA E/R PROPUESTO

No se ha podido recoger en el diagrama E/R propuesto en la Figura 1.14 los siguientes supuestos semánticos:

- SS1. Solo se almacenarán los datos del inquilino si la propiedad está alquilada y no vive el propietario en ella.
- SS2. Solo se almacenará la dirección del propietario si éste no vive en su propiedad.
- SS3. Nunca se cargará a cuenta de la comunidad de vecinos un recibo que no pertenezca a una compañía contratada por la comunidad.

SS4. Toda propiedad que figura como presidente o vocal en una comunidad de vecinos participa en la interrelación Consta para la misma comunidad de vecinos.

PROBLEMA 1.3. MEDICAMENTOS

Se desea diseñar una Base de Datos para controlar los costes económicos derivados del consumo de medicamentos por parte de los pacientes así como de los distintos servicios de especialidades que componen el hospital. Los supuestos semánticos que se van a contemplar son:

Cada paciente ingresado en el hospital consume una serie de fármacos durante el periodo de su hospitalización cuya gestión permitirá generar informes de gasto de fármacos por paciente, por servicio (oncología, pediatría digestiva, traumatología, etc.) o por diagnóstico y así llevar un control contable más exhaustivo de los gastos que el hospital sufraga por paciente o servicio.

Así, interesa almacenar la información relativa a los ingresos de pacientes con los datos de cada ingreso realizado en un servicio determinado de hospital, los consumos de fármacos producidos por un determinado ingreso, así como los consumos generales de fármacos generados por la actividad propia de los servicios del hospital que no están asignados a un paciente en particular (por ejemplo, suero fisiológico, alcohol, bicarbonato, etc.).

También se guardará información sobre los servicios así como el vademécum de fármacos donde se recogen todos los medicamentos existentes en el hospital que pueden ser consumidos bien por los pacientes ingresados bien por los servicios del hospital.

La BD deberá permitir la inserción, borrado, consulta y modificación de los pacientes que se encuentran en cada servicio del hospital. Cada uno de estos pacientes se identifica por su número de historia clínica y se desea conocer además el nombre, el número de la seguridad social (si lo tuviera), la dirección, un teléfono y la fecha de nacimiento. Un paciente puede haber estado ingresado en más de una ocasión en el hospital; cada ingreso se caracteriza por un número en secuencia dentro de cada número de historia clínica e interesa el servicio en el que ha sido ingresado, el diagnóstico y la fecha de ingreso y la fecha de alta si se hubiera producido. Un ingreso se realiza en un determinado servicio del hospital (traumatología, pediatría,etc.).

Cada uno de los consumos de cada paciente así como los consumos que cada servicio genera para su actividad propia se componen de un número determinado de unidosis⁸ de fármacos. De estos fármacos se desea conocer un nº

de registro, el nombre comercial, nombre clínico, el compuesto químico, su ubicación, el código de proveedor, el número de unidosis por envase, precio por unidosis y el precio total porenvase.

Será de gran importancia obtener los listados de gastos de unidosis por los pacientes de un determinado servicio y el gasto total de cada servicio.

Interesa también almacenar los facultativos que trabajan en el hospital identificados por su Nº de colegiado y caracterizados por su nombre, dirección, un teléfono de contacto y el servicio del hospital al que están adscritos, teniendo en cuenta que un médico solo puede trabajar en un determinado servicio.

Por otro lado, no solo interesa saber qué fármacos, en qué fecha y en qué cantidades se han consumido en un determinado ingreso de un paciente sino también el facultativo que los recetó⁹ teniendo en cuenta que durante un ingreso de un paciente un facultativo puede haberle recetado varios medicamentos pero que un medicamento solo es recetado a un determinado paciente ingresado por un único facultativo.

Un facultativo puede recetar el mismo medicamento a un paciente ingresado varias veces en distintas fechas y a un paciente ingresado le pueden recetar medicamentos distintos facultativos.

De los consumos generales de fármacos originados por la actividad propia de los servicios se almacenará el número de unidosis de cada fármaco así como la fecha del consumo.

Por último, se desea registrar la información relativa a las revisiones que los facultativos realizan a un determinado paciente en un determinado ingreso; se guardará la fecha, la hora y un pequeño informe.

Se pide:

Realizar el diseño conceptual de los supuestos semánticos anteriores basándose en el Modelo Entidad/Interrelación. Indicar claramente todos aquellos supuestos semánticos que ha sido imposible incorporar en el diagrama E/R.

DISCUSIÓN DEL ENUNCIADO

PARTE 1

Cada paciente ingresado en el hospital consume una serie de fármacos durante el periodo de su hospitalización cuya gestión permitirá generar informes de gasto de fármacos por paciente, por servicio (oncología, pediatría digestiva, traumatología, etc.) o por diagnóstico y así llevar un control contable más exhaustivo de los gastos que el hospital sufraga por paciente o servicio.

Así, interesa almacenar la información relativa a los ingresos de pacientes con los datos de cada ingreso realizado en un servicio determinado de hospital, los consumos de fármacos producidos por un determinado ingreso, y los consumos generales defármacos generados por la actividad propia de los servicios del hospital que no están asignados a un paciente en particular (por ejemplo, suero fisiológico, alcohol, bicarbonato,etc.).

También se guardará información sobre los servicios así como el vademécum de fármacos donde se recogen todos los medicamentos existentes en el hospital que pueden ser consumidos bien por los pacientes ingresados bien por los servicios del hospital.

En estos párrafos se identifican varios conceptos que se tienen que representar en nuestra base de datos, como son los pacientes ingresados en determinados servicios del hospital y los fármacos (tantos los consumidos por los pacientes en sus ingresos, como los consumidos por un determinado servicio hospitalario, como los de consumo general, como el vademécum del hospital). Estos primeros párrafos nos proporcionan información general sobre el universo del discurso. Será en párrafos posteriores donde nos indicarán más propiedades sobre los conceptos detectados en estos párrafos, por lo que los diagramas E/R los completaremos entonces.

PARTE 2

La BD deberá permitir la inserción, borrado, consulta y modificación de los pacientes que se encuentran en cada servicio del hospital. Cada uno de estos pacientes se identifica por su número de historia clínica y se desea

conocer además el nombre, el número de la seguridad social (si lo tuviera), la dirección, un teléfono y la fecha de nacimiento. Un paciente puede haber estado ingresado en más de una ocasión en el hospital; cada ingreso se caracteriza por un número en secuencia dentro de cada número de historia clínica e interesa el servicio en el que ha sido ingresado, el diagnóstico y la fecha de ingreso y la fecha de alta si se hubiera producido. Un ingreso se realiza en un determinado servicio del hospital (traumatología, pediatría, etc.).

Se detecta una entidad *PACIENTE* que se identifica por su *Número de historia clínica* y se caracteriza por su *Nombre, Número de la seguridad social (NSS)* (si lo tuviera), *Dirección, Teléfono* y *Fecha de nacimiento*. Todos los atributos asociados a la entidad *PACIENTE* serán atributos univaluados y obligatorios excepto el atributo *NSS*, que será opcional. Además, tanto el *Número de historia clínica* como el *NSS* podrían identificar al paciente, por lo que se elige uno de ellos como atributo identificador principal (AIP) y otro como atributo identificador alternativo (AIA). En este caso se ha elegido el *Número de historia clínica* como AIP y el *NSS* como AIA.

Por otro lado, se indica que un paciente puede haber estado ingresado en más de una ocasión en el hospital. La información sobre el ingreso de cada paciente se almacenará en una entidad denominada *INGRESO* que depende en identificación de la entidad *PACIENTE*. Por lo tanto, existirá una interrelación 1:N en identificación entre la entidad *PACIENTE* y la entidad *INGRESO*, especificando que un paciente puede haber estado ingresado en nuestro hospital al menos una vez (si no, no estaría registrado en nuestra base de datos como paciente). La entidad *INGRESO* se identificará por el atributo identificador de la entidad *PACIENTE* unido al *Número de secuencia* del ingreso. Además, *INGRESO* posee como atributos univaluados la *Fecha de ingreso*, el *Diagnóstico* y la *Fecha de alta* del paciente. Todos estos atributos serán obligatorios excepto la *Fecha de alta*, que puede que no se conozca.

Por último, es necesario indicar que un ingreso siempre se realiza en un determinado servicio del hospital. Aunque en el enunciado no viene explícitamente indicado, se supone que un paciente no puede estar ingresado en dos servicios hospitalarios simultáneamente y que en un servicio del hospital pueden estar inscritos varios ingresos al mismo tiempo o puede que no haya ninguno. También se supone que un servicio del hospital se puede identificar por su *Nombre*, debido a que no existirán dos servicios en el hospital con el mismo nombre.

El diagrama E/R correspondiente a estos supuestos se muestra en la Figura 1.15.

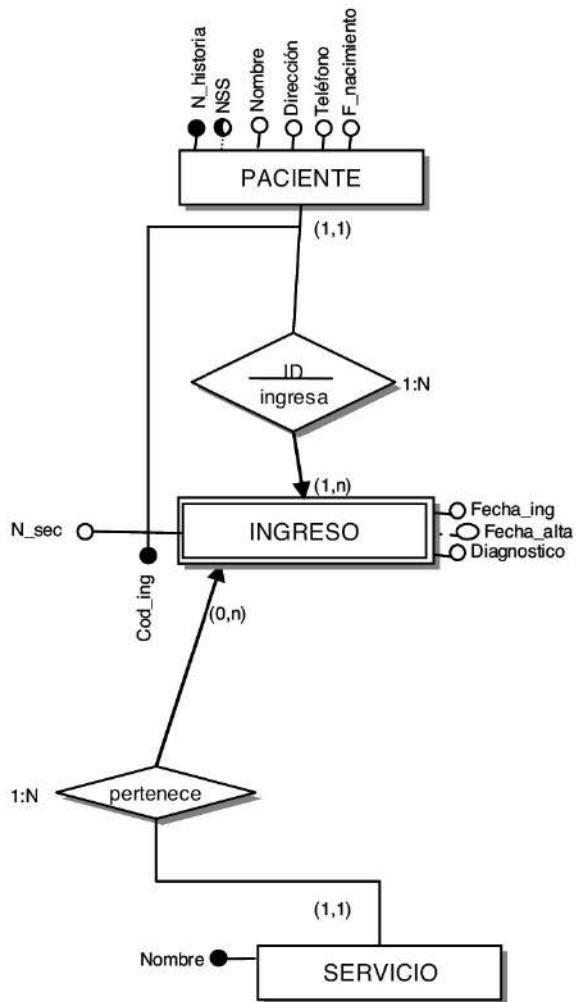


Figura 1.15. Diagrama E/R correspondiente a la Parte 2

En este diagrama no se ha podido representar el supuesto semántico de que toda fecha de alta de un ingreso ha de ser posterior o igual a la fecha de ingreso. Además, también será necesario comprobar que la fecha de alta de un paciente es posterior o igual a la fecha de nacimiento del paciente. Otro supuesto semántico que no se puede representar en el diagrama E/R es que los períodos de ingreso de un determinado paciente no se puedan solapar.

PARTE 3

Cada uno de los consumos de cada paciente así como los consumos que cada servicio genera para su actividad propia se componen de un número

determinado de unidosis de fármacos. De estos fármacos se desea conocer un nº de registro, el nombre comercial, nombre clínico, el compuesto químico, su ubicación, el código de proveedor, el número de unidosis por envase, precio por unidosis y el precio total por envase.

Será de gran importancia obtener los listados de gastos de unidosis por los pacientes de un determinado servicio y el gasto total de cada servicio.

En este párrafo se habla de un nuevo concepto: los fármacos. La entidad FÁRMACO se puede identificar por su *Número de registro*, por su *Nombre comercial* o por su *Nombre clínico*. Elegimos el primero como atributo identificador principal y los nombres comercial y clínico como atributos identificadores alternativos. Además, los fármacos se caracterizan por su *Compuesto químico*, *Ubicación* en el hospital, el *Código del proveedor*, el *Número de unidosis por envase*, el *Precio por unidosis* y el *Precio total por envase*. Todos estos atributos serán atributos univaluados y obligatorios. Además, el precio total por envase se considerará como un atributo derivado, ya que su valor se podrá calcular mediante la fórmula:

$$\text{Precio_envase} = \text{Precio_unidosis} * \text{Nº_unidosis_envase}$$

En cuanto al segundo párrafo de esta parte, nos recuerda la importancia de obtener los listados de los gastos por fármacos de cada ingreso y de cada servicio del hospital. Todavía no añadiremos esta información al diagrama E/R ya que no se ha almacenado aún información de los fármacos consumidos por cada ingreso. Esta información se plantea posteriormente en elunciado.

El diagrama E/R correspondiente a estos supuestos semánticos se muestra en la Figura 1.16.

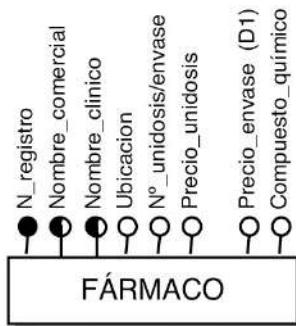


Figura 1.16. Diagrama E/R de la entidad FÁRMACO

PARTE 4

Interesa también almacenar los facultativos que trabajan en el hospital identificados por su Nº de colegiado y caracterizados por su nombre, dirección, un teléfono de contacto y el servicio del hospital al que están adscritos, teniendo en cuenta que un médico solo puede trabajar en un determinado servicio.

En estos párrafos se descubre un nuevo concepto: los facultativos o médicos que trabajan en el hospital. La entidad *FACULTATIVO* se identificará por el *Número de colegiado* y tendrá como atributos univalueados y obligatorios el *Nombre*, *Dirección* y *Teléfono de contacto*. Para indicar a qué servicio del hospital se encuentra adscrito, será necesario incluir una interrelación binaria (**Trabaja**) con la entidad *SERVICIO* que previamente habíamos identificado en la Parte 2 de este ejercicio. Se tendrá en cuenta que un médico solo puede trabajar en un servicio de forma simultánea y que en un servicio al menos ha de trabajar un facultativo (aunque esta información no se haya descrito de forma explícita en el enunciado). El tipo de correspondencia de la interrelación Trabaja, por lo tanto, será 1:N.

El diagrama E/R que describe estos supuestos semánticos se muestra en la Figura 1.17.

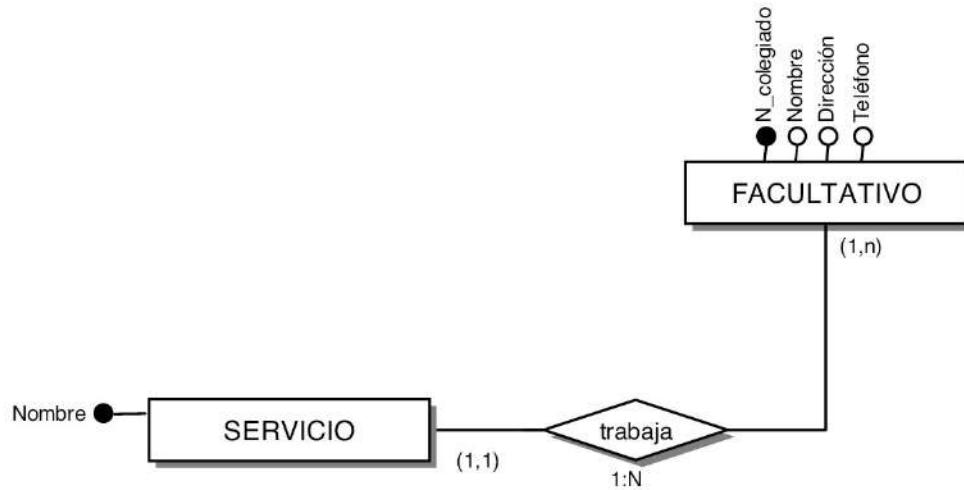


Figura 1.17. Diagrama E/R de la interrelación Trabaja

Según el enunciado, se podría haber supuesto que la entidad *FACULTATIVO* es una entidad débil respecto a la entidad *SERVICIO*, ya que si un servicio del hospital no existe, no tiene sentido hablar de facultativos del

hospital. En este caso, la interrelación **Trabaja** debería representarse como una interrelación en existencia según muestra la Figura 1.18. Sin embargo, en nuestra base de datos la información sobre los médicos y las recetas que estos prescriben es información muy valiosa, por lo que no se ha tratado como si fuera una entidad débil.

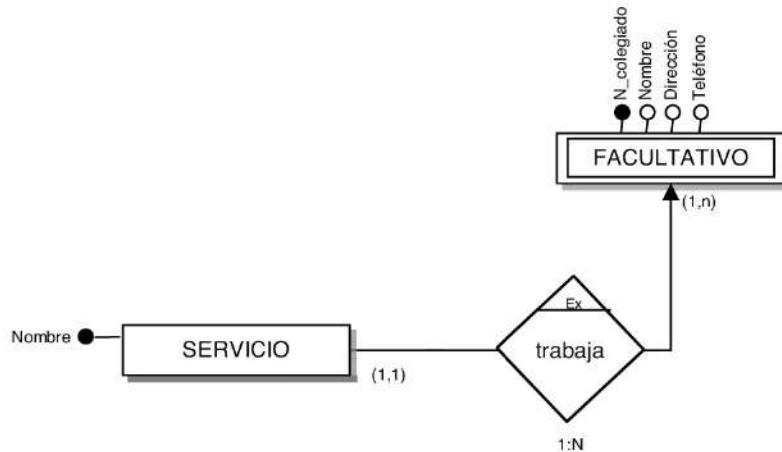


Figura 1.18. Propuesta incorrecta de modelado de la interrelación *Trabaja*

En el diagrama E/R de la Figura 1.19 se ha supuesto que si un facultativo se pone enfermo y ha de ser ingresado en el hospital, se creará una ocurrencia con los datos del facultativo en la entidad **PACIENTE**. Otra opción válida sería crear una generalización total solapada con las entidades **FACULTATIVO** y **PACIENTE**, donde la nueva entidad **PERSONA** tendrá los atributos comunes de ambos. En este caso sería necesario encontrar un atributo común a ambos que pudiera considerarse como el atributo identificador principal de la entidad (por ejemplo, añadiendo un código identificador). La Figura 1.19 muestra el diagrama E/R con la jerarquía.

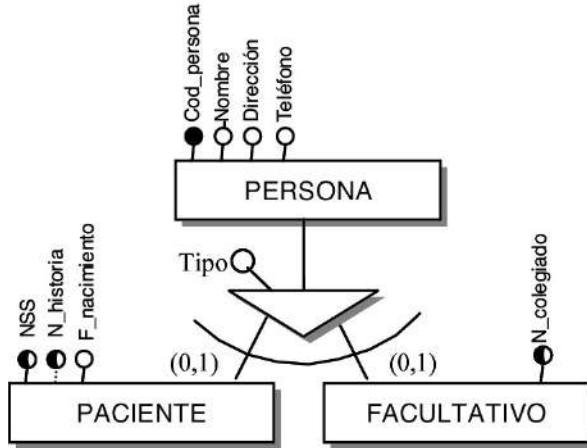


Figura 1.19. Propuesta de jerarquía de la entidad PERSONA

En la propuesta de solución de este ejercicio se ha optado por no incluir la jerarquía, ya que se ha incluido el atributo *Código_persona* de forma artificial y los beneficios de incluir la generalización no son tantos.

PARTE 5

Por otro lado, no solo interesa saber qué fármacos, en qué fecha y en qué cantidades se han consumido en un determinado ingreso de un paciente sino también el facultativo que los recetó teniendo en cuenta que durante un ingreso de un paciente un facultativo puede haberle recetado varios medicamentos pero que un medicamento solo es recetado a un determinado paciente ingresado por un único facultativo.

Un facultativo puede recetar el mismo medicamento a un paciente ingresado varias veces en distintas fechas y a un paciente ingresado le pueden recetar medicamentos distintos facultativos.

Para almacenar esta información en la base de datos será necesario incluir en el diagrama E/R la interrelación ternaria Receta que indique qué fármaco se ha consumido en cada ingreso por prescripción de qué facultativo. Un fármaco recetado en un ingreso solo puede ser recetado por uno y solo un facultativo; un facultativo a un paciente puede recetarle varios fármacos en el mismo ingreso; y a su vez un facultativo puede recetar el mismo fármaco a varios pacientes. Por lo tanto, el tipo de correspondencia de la interrelación es 1:N:M.

En un principio podríamos pensar que la interrelación ternaria relacionaba la entidad *PACIENTE* en lugar de la entidad *INGRESO* con *FACULTATIVO* y

FÁRMACO, pero esto sería un error, ya que necesitaríamos conocer en qué ingreso se recetaron los fármacos para llevar la cuenta del gasto por ingreso y servicio.

Para almacenar información sobre la fecha y la cantidad de fármacos recetada será necesario añadir un atributo multivaluado compuesto por estos datos (*Fecha_receta* y *Nº_unidosis*). Será multivaluado debido a que el mismo facultativo ha podido recetar el mismo fármaco en el mismo ingreso de un paciente varias veces y en distintas cantidades pero en distintas fechas. En la interrelación ternaria *Receta* se ha de tener en cuenta que no se permite que dos facultativos diferentes receten el mismo fármaco al mismo paciente en exactamente la misma fecha. Por ello, sería más conveniente incluir información sobre horas y minutos en la fecha de la receta.



La interrelación **Receta** no implica que todo paciente ingresado haya recibido una receta, aunque las cardinalidades mínimas de la interrelación sean igual a uno. Las cardinalidades mínimas de la interrelación indican que siempre que se haya prescrito una receta en una fecha dada, al menos haya sido a un paciente, ha tenido que ser al menos por un facultativo y al menos se ha recetado un fármaco.

Por otro lado, al almacenar la información de qué fármacos ha consumido cada paciente con esta interrelación, tenemos que echar la vista atrás en el enunciado y recordar el siguiente párrafo:

Será de gran importancia obtener los listados de gastos de unidosis por los pacientes de un determinado servicio y el gasto total de cada servicio.

Este párrafo nos está indicando que se consultará habitualmente la información de qué gastos farmacológicos se hicieron por ingreso y qué gastos farmacológicos se hicieron en un determinado servicio. Por ello, sería conveniente incluir dos atributos derivados en el diagrama E/R: un atributo derivado *Gasto_ingreso* en la entidad *INGRESO* cuyo valor se calculará gracias a la suma de los gastos por fármacos recetados en ese ingreso; y otro atributo derivado *Gasto_servicio* en la entidad *SERVICIO* cuyo valor se calculará sumando todos los gastos de los ingresos adscritos a dicho servicio unido a los gastos provocados por consumo general del servicio (comentado posteriormente).

La Figura 1.20 muestra el diagrama E/R que representa la semántica

asociada a estos párrafos. La entidad *SERVICIO* aparece sin interrelacionarse en este diagrama para mostrar el atributo derivado, aunque ya vimos en diagramas anteriores cómo se interconectaba con el resto del diagrama entidad/interrelación.

En el diagrama E/R de la Figura 1.20 ha sido imposible recoger la semántica de que un facultativo que receta un fármaco a un paciente ingresado en una fecha (incluidas horas y minutos) le receta un único número de unidosis; un número de unidosis no implicaría la fecha en la que un facultativo receta los fármacos a un paciente.

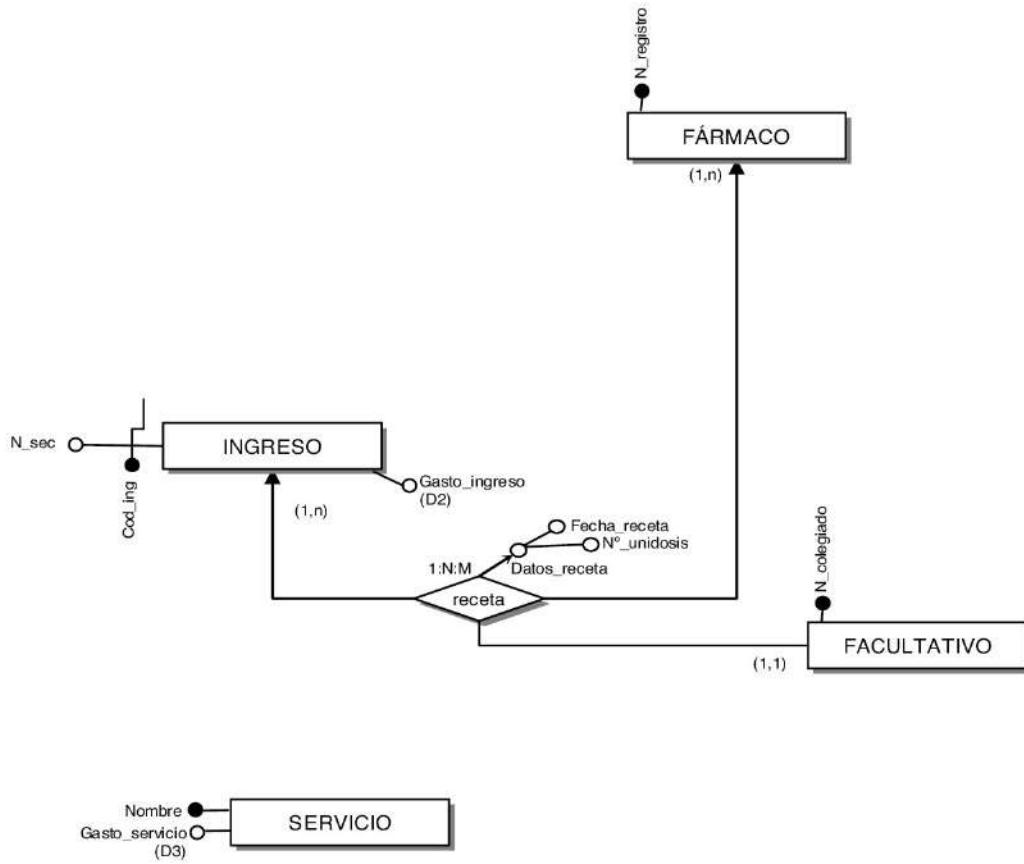


Figura 1.20. Diagrama E/R propuesto para modelar la interrelación Receta

Tras haber realizado el diagrama E/R de este apartado nos podría haber surgido una duda respecto a interrelaciones redundantes. ¿Es redundante la interrelación **Trabaja** detectada en el apartado anterior? Esta duda es razonable, ya que se tiene información sobre las recetas que prescribe un determinado facultativo a un paciente ingresado y se conoce a qué servicio exactamente pertenece ese ingreso, se podría suponer en qué servicio ha trabajado un

determinado facultativo. Aun así, la interrelación Trabaja no se puede considerar como redundante, puesto que el facultativo puede trabajar en un determinado servicio y nunca haber prescrito ninguna receta a pacientes del servicio (se perdería información sobre los facultativos que nunca han prescritorecetas).

PARTE 6

De los consumos generales de fármacos originados por la actividad propia de los servicios se almacenará el número de unidosis de cada fármaco así como la fecha del consumo.

Se añadirá la interrelación **Consumo_general** entre las entidades **SERVICIO** y **FÁRMACO** para representar la información de qué fármacos han sido consumidos por cada servicio. Se trata de una interrelación con tipo de correspondencia N:M que, además, posee dos atributos multivaluados: la *Fecha_consumo* del fármaco y el *Número_unidosis* de cada fármaco que se consumió. En esta interrelación se desea almacenar información sobre los consumos originados por la actividad propia de cada servicio (no están asociados a los consumos poringreso), por lo que no se tratará de atributos derivados.

La Figura 1.21 muestra el diagrama E/R de los supuestos semánticos comentados.

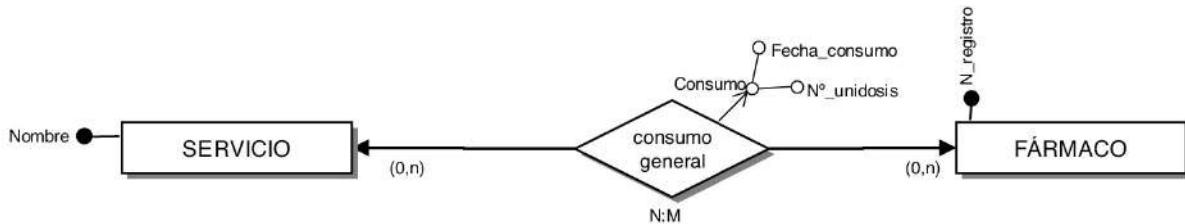


Figura 1.21. Diagrama E/R de la interrelación Consumo General

En el diagrama E/R de la Figura 1.21 ha sido imposible recoger la semántica de que se ha consumido un único número de unidosis de cada fármaco en una determinada fecha (incluidas horas y minutos), mientras que el número de unidosis no implicaría la fecha en la que se ha consumido el fármaco en un determinado servicio.

PARTE 7

Por último, se desea registrar la información relativa a las revisiones que los facultativos realizan a un determinado paciente en un determinado ingreso; se guardará la fecha, la hora y un pequeño informe.

Para almacenar esta semántica será necesario añadir una interrelación *Revisa* entre las entidades *INGRESO* y *FACULTATIVO*. La interrelación almacenará la información de qué facultativo ha tratado a cada paciente en cada ingreso, la fecha, la hora y un informe al respecto. El enunciado no es explícito en cuanto al número de facultativos que pueden firmar los informes para un determinado ingreso. Si supusiéramos que para cada ingreso solo existe un facultativo responsable de firmar los informes y en ningún caso podría ser firmado por otro facultativo, se trataría de una interrelación 1:N. Por otro lado, si se supone que un paciente ingresado puede ser visitado por varios facultativos y todos ellos pueden firmar los informes, se trataría de una interrelación N:M. En la Figura 1.22 se ha supuesto que únicamente un facultativo puede firmar los informes para cada ingreso. También se ha supuesto que sobre un ingreso puede que en un momento dado no exista aún ningún informe por parte de un facultativo (cardinalidad mínima de cero).

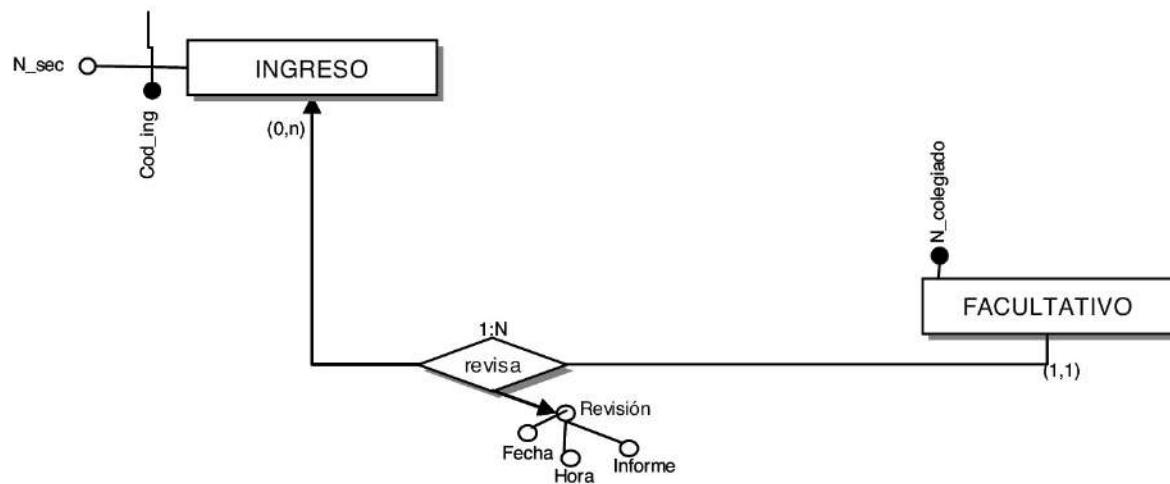


Figura 1.22. Diagrama E/R correspondiente a la interrelación *Revisa*

En principio podríamos haber pensado erróneamente que la interrelación *Revisa* relacionaba las entidades *PACIENTE* y *FACULTATIVO*, pero es más correcto indicarlo sobre el *INGRESO* de cada paciente.

Un supuesto semántico que no se ha podido recoger en este diagrama es que la fecha de la revisión se ha de encontrar entre las fechas de baja y alta de un ingreso. En el caso de que la fecha de alta no se conozca se supondrá que la

fecha de la revisión ha de ser anterior a la actual y posterior a la fecha de baja del ingreso. Tampoco se ha podido recoger la semántica de que existe un único informe en la fecha y hora en la que un facultativo revisa a un determinado paciente ingresado. No se podría asegurar que un facultativo escribiera para el mismo paciente exactamente el mismo informe en distintas fechas y horas. Será necesario comprobar también que un facultativo no puede realizar dos revisiones en la misma fecha y hora.

Tras haber representado la interrelación **Revisa** nos podemos dar cuenta de que existen ciclos en el diagrama E/R con semántica relacionada, lo que podría producir redundancias. Está claro que la interrelación **Revisa** es necesaria, porque en ningún otro sitio se almacena información sobre los informes a los pacientes, pero, ¿la interrelación **Trabaja** detectada en la Parte 4 de este problema podría ser redundante? La respuesta es que no, no se puede considerar como redundante ya que puede ocurrir que un facultativo que trabaja en un determinado servicio aún no haya realizado ninguna revisión de pacientes ingresados en el mismo servicio. Si elimináramos la interrelación **Trabaja**, perderíamos esta información.

SOLUCIÓN PROPUESTA

La Figura 1.23 muestra el diagrama E/R completo que, junto con los supuestos semánticos que no se han podido representar en el diagrama, componen el esquema E/R del ejercicio.

SUPUESTOS ASOCIADOS AL DIAGRAMA E/R

No se puede recoger en el diagrama de la Figura 1.23 los siguientes supuestos semánticos:

- SS1. Los periodos en los que un paciente se encuentra ingresado no se pueden solapar.
- SS2. En la entidad *INGRESO* siempre se ha de cumplir que el valor del atributo *Fecha_ing* es anterior al valor del atributo *Fecha_alta* en el caso de que sea conocido.
- SS3. La fecha de nacimiento de un paciente siempre ha de ser anterior o igual a la fecha de ingreso de ese paciente en el hospital.
- SS4. En la interrelación *Revisa* se ha de cumplir que la fecha de la revisión ha de encontrarse entre las fechas de inicio y alta de un ingreso. En el caso de que la fecha de alta del ingreso no se conozca, se supondrá que la fecha de la revisión ha de ser anterior a la actual y posterior a la fecha de inicio del ingreso.
- SS5. En la interrelación **Receta** se ha de cumplir que la fecha de la receta, al igual que ocurría con la fecha de la revisión, ha de encontrarse entre las fechas de ingreso del paciente.
- SS6. Cada vez que un facultativo revisa a un determinado paciente (en una fecha y hora) se crea un único informe.
- SS7. Un facultativo no puede realizar dos revisiones de forma simultánea.
- SS8. Existe un único valor para el atributo *Nº_unidosis* para cada vez (fecha/hora/minuto) que un facultativo le receta un fármaco a un paciente ingresado.
- SS9. Lo mismo ocurre para la interrelación *Consumo_general*. La fecha de

consumo de un fármaco en un servicio implica el número de unidosis consumidas.

SS10. En la entidad *FÁRMACO* el atributo *Precio_envase* se calculará según la fórmula: “*Precio_envase* = *Precio_unidosis***Nº_unidosis*/*envase*”.

SS11. En la entidad *INGRESO* el atributo *Gasto_ingreso* se calculará a partir de las ocurrencias de la interrelación Receta para ese ingreso. Se sumará el gasto de cada receta, calculando este gasto multiplicando el número de unidosis consumida por el valor del atributo *Precio_unidosis* de la entidad *FÁRMACO*.

SS12. En la entidad *SERVICIO* el atributo *Gasto_servicio* se calculará sumando los gastos derivados de los ingresos (sumando el valor del atributo *Gasto_ingreso* de todos los ingresos pertenecientes al servicio) a los gastos derivados del consumo general del servicio en el hospital (derivados de la interrelación **Consumo_general**).



Debido a que en el diagrama E/R se almacena información actual del servicio del hospital en el que trabaja cada facultativo y no se puede asegurar que anteriormente el facultativo no haya trabajado en otros servicios, no se tendrá que controlar que un facultativo únicamente puede recetar a pacientes ingresados en el servicio en el que él trabaja y únicamente puede revisar informes de dichos pacientes. En el caso de que se hubiera almacenado información histórica en el diagrama de los servicios en los que ha trabajado cada facultativo, sería necesario realizar dichas comprobaciones.

DIAGRAMA E/R

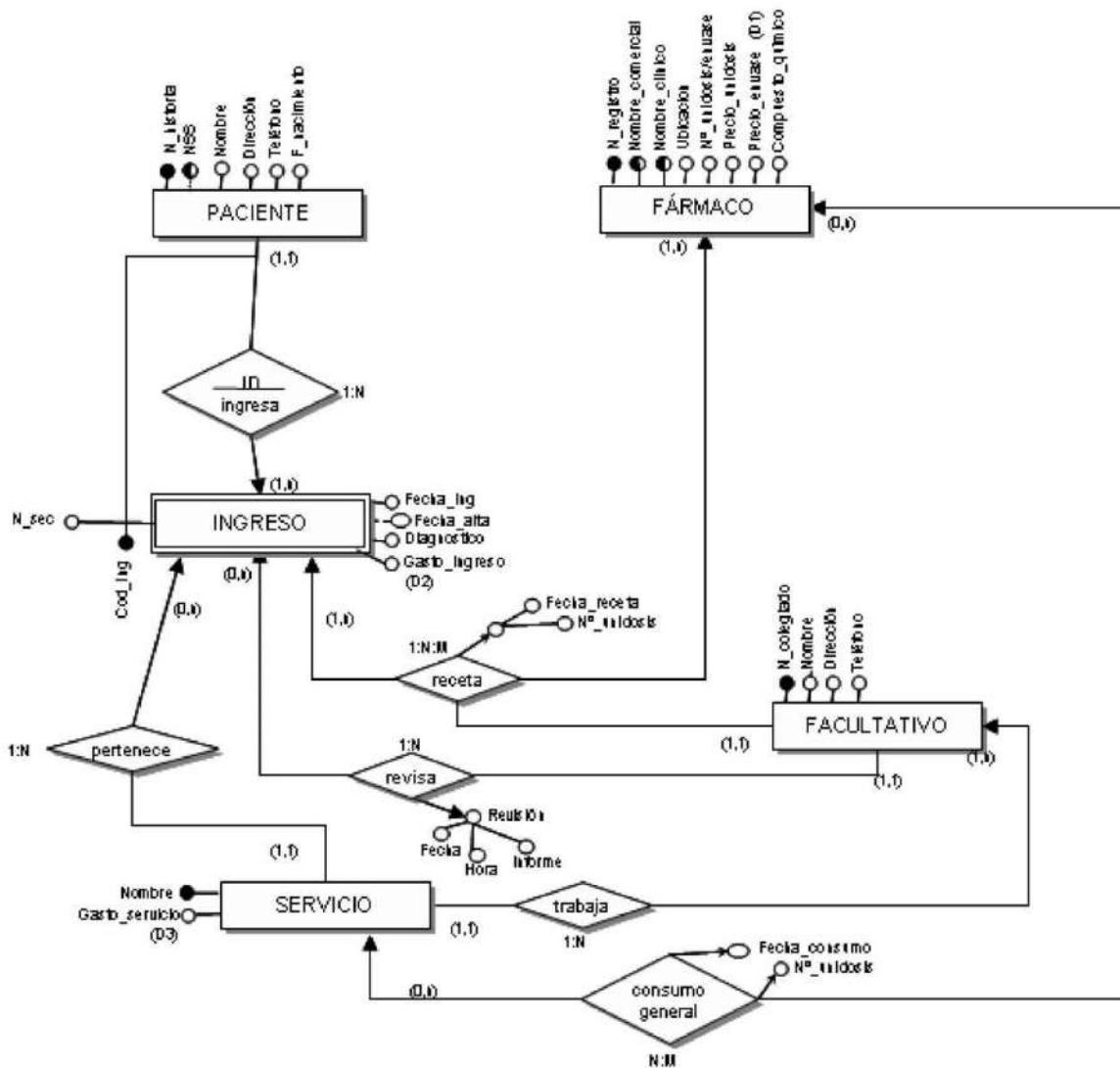


Figura 1.23. Diagrama E/R propuesto sobre la gestión de medicamentos en un hospital

PROBLEMA 1.4: PROYECTOS DE INVESTIGACIÓN

El Departamento de Informática de la Universidad Carlos III de Madrid necesita una base de datos para almacenar la información concerniente a los proyectos de investigación tanto actuales como pasados en los que trabajan los profesores y así poder llevar a cabo una gestión más eficiente. La información que se desea almacenar corresponde a los siguientes supuestos semánticos.

En el departamento los profesores participan en proyectos de investigación caracterizados por un código de referencia único, por un nombre, un acrónimo, un presupuesto total, el programa de I+D que lo financia, una fecha de inicio y una fecha de finalización y una breve descripción de los objetivos del proyecto.

En los proyectos trabajan profesores del departamento durante un periodo de tiempo, es decir, una fecha de inicio y una fecha de fin, pudiendo ocurrir que un profesor trabaje en el mismo proyecto en varias épocas (f_ini , f_fin) diferentes. Un profesor se identifica por su nombre y apellidos y se caracteriza por su despacho y teléfono y puede trabajar en varios proyectos simultáneamente y en un proyecto de investigación trabajan varios profesores. De todos los profesores que trabajan en el proyecto hay uno que es el investigador principal de proyecto que interesa conocer. Es importante tener en cuenta que el profesor investigador del proyecto nunca varía a lo largo de la vida del proyecto de investigación.

Los profesores pueden ser doctores o no doctores, de tal manera que un profesor no doctor siempre tiene a un único profesor doctor como supervisor en un momento determinado, interesando almacenar los supervisores y periodos de tiempo de la supervisión que ha tenido un determinado profesor no doctor. En relación con la participación de los profesores en proyectos de investigación, el investigador principal de un proyecto siempre tiene que ser un doctor.

Por otro lado, los proyectos de investigación producen una serie de publicaciones sobre las que también interesa guardar información. Una publicación se caracteriza por un número en secuencia dentro de cada proyecto de investigación y se guardará el título y los profesores que la han escrito; las publicaciones son de dos tipos, publicaciones en congresos y publicaciones en revista; de las primeras se almacenará el nombre del congreso, su tipo (nacional o internacional), la fecha de inicio y de fin, el lugar de celebración, país y la editorial que ha publicado las actas del congreso (si es que se han publicado); de

las publicaciones en revista interesa saber el nombre de la revista, la editorial, el volumen, el número y las páginas de inicio y fin.

No solamente interesa conocer los profesores que han participado en las publicaciones de los proyectos de investigación sino también las líneas de publicación que cubren estas publicaciones. Una línea de investigación se identifica por un código, un nombre (por ejemplo, “recuperación de información multilingüe”, “bases de datos espacio-temporales”, etc.) y un conjunto de descriptores (por ejemplo, la línea de investigación “bases de datos temporales” puede tener como descriptores “Bases de Datos”, “SGBD Relacional”, “Dimensión temporal”).

Los profesores tendrán asociados en la BD las líneas de investigación en las que trabajan incluso podría ocurrir que hubiera profesores que no tuvieran ninguna línea asignada.

Así, tanto los profesores doctores como los no doctores pueden escribir publicaciones sobre una o más líneas de investigación y nos interesa saber sobre qué línea de investigación ha escrito un determinado profesor en una publicación, teniendo en cuenta que un profesor que participa en una publicación solo escribe en el ámbito de una línea de investigación y que una determinada publicación puede cubrir varias líneas de investigación.

Por último, aparte de la información especificada para los proyectos de investigación también se almacenarán las líneas de investigación que abarca cada proyecto.

Se pide:

Realizar el esquema E/R teniendo en cuenta los supuestos semánticos anteriores. Indicar si no se ha podido recoger algún supuesto semántico en el diagrama E/R.

DISCUSIÓN DEL ENUNCIADO

PARTE 1

En el departamento los profesores participan en proyectos de investigación caracterizados por un código de referencia único, por un nombre, un acrónimo, un presupuesto total, el programa de I+D que lo financia, una fecha de inicio y una fecha de finalización y una breve descripción de los objetivos del proyecto.

En los proyectos trabajan profesores del departamento durante un periodo de tiempo, es decir, una fecha de inicio y una fecha de fin, pudiendo ocurrir que un profesor trabaje en el mismo proyecto en varias épocas (f_{ini} , f_{fin}) diferentes. Un profesor se identifica por su nombre y apellidos y se caracteriza por su despacho y teléfono y puede trabajar en varios proyectos simultáneamente y en un proyecto de investigación trabajan varios profesores. De todos los profesores que trabajan en el proyecto hay uno que es el investigador principal de proyecto que interesa conocer. Es importante tener en cuenta que el profesor investigador del proyecto nunca varía a lo largo de la vida del proyecto de investigación.

Los profesores pueden ser doctores o no doctores, de tal manera que un profesor no doctor siempre tiene a un único profesor doctor como supervisor en un momento determinado, interesando almacenar los supervisores y periodos de tiempo de la supervisión que ha tenido un determinado profesor no doctor. En relación con la participación de los profesores en proyectos de investigación, el investigador principal de un proyecto siempre tiene que ser un doctor.

En estos supuestos identificamos la entidad *PROFESOR* y la entidad *PROYECTO DE INVESTIGACIÓN*. El proyecto de investigación posee un código de referencia que es único, por lo que se tratará como el atributo identificador principal (AIP) de la entidad. Además también se desea almacenar su nombre, acrónimo, el nombre del proyecto de I+D que lo financia, la fecha de inicio del proyecto, la fecha de finalización del mismo y una breve descripción de objetivos del proyecto. Todos estos atributos serán atributos univaluados y obligatorios, donde el *Nombre* del proyecto se supone único en el departamento (no habrá dos proyectos con el mismo nombre), por lo que podrá actuar como

atributo identificador alternativo (AIA) de la entidad. En el diagrama E/R no se podrá representar que la fecha de inicio del proyecto de investigación siempre ha de ser anterior o igual a la fecha de finalización del mismo.

El segundo párrafo trata de la participación de los profesores en los proyectos de investigación. Se indica que se desea almacenar información histórica acerca de la participación de los profesores en los proyectos. Para ello se creará una interrelación binaria N:M (**Participa**) entre la entidad *PROFESOR* y la interrelación *PROYECTO*, donde un profesor puede no haber participado en ningún proyecto o puede haber participado en muchos y, a su vez, en un proyecto al menos ha de participar un profesor. La interrelación binaria tendrá asociado un atributo multivaluado compuesto donde se indican los períodos de tiempo (*fecha_inicio* y *fecha_fin*) en los que un profesor ha participado en un proyecto. Al considerar un atributo multivaluado se permite que el mismo profesor haya participado en el mismo proyecto varias veces a lo largo del tiempo. Se ha supuesto que la fecha de finalización puede ser desconocida, por lo que se trata como un atributo opcional. Lo que no se puede representar con el diagrama (y que es muy necesario controlar en la base de datos) es que no se deberían superponer los períodos de tiempo en los que un profesor participa en un proyecto dado. Tampoco se ha podido representar en el diagrama que la fecha de inicio de participación de un profesor en un proyecto de investigación implica una única fecha de finalización de la participación. De igual manera, si la fecha de finalización fuera conocida, ésta implicaría una única fecha de inicio de participación en el proyecto para cada profesor. Además, la fecha de inicio de participación en un proyecto siempre ha de ser anterior a la fecha de finalización.

El AIP de la entidad *PROFESOR* es el atributo compuesto por el nombre y los apellidos del mismo. De cada profesor, además, se desea almacenar información sobre su despacho y número de teléfono. Como no se tiene información de que los despachos y teléfonos estén o no compartidos por varios profesores, no se podrá tomar ninguna decisión respecto a si se podría identificar a un profesor a través de ellos (no se podrán tratar como AIA).

Siempre existirá un profesor con el rol de investigador principal en cada proyecto. La última frase del párrafo tres indica que este profesor ha de ser doctor (no puede tratarse de un profesor que no sea doctor). Esta diferencia es significativa, por lo que nos interesaría crear una generalización de la entidad profesor, especificando si estos son doctores o no. Los subtipos de la jerarquía son las entidades *DOCTOR* y *NO_DOCTOR*, mientras que el atributo discriminante de la jerarquía será el tipo de profesor, cuyo dominio es {Doctor,

`NoDoctor}`. La jerarquía será exclusiva, ya que un profesor doctor nunca es un profesor no doctor. También será total, ya que únicamente existen profesores no doctores y profesores doctores.

La interrelación binaria 1:N **Investigador_principal** indica cuándo un profesor doctor actúa como investigador principal de un proyecto. Un profesor doctor podría no ser nunca un investigador principal, podría serlo de varios proyectos simultáneamente y además un proyecto solamente puede tener un investigador principal. Esta interrelación no almacena información histórica acerca de qué profesores han sido investigadores principales de cada proyecto a lo largo del tiempo. Solo almacena información actual sobre el investigador principal del proyecto. Un supuesto semántico que ha sido imposible representar en este diagrama es que un profesor únicamente puede ser investigador principal de aquellos proyectos en los que participa.

Por último, estos supuestos semánticos tratan de la relación de supervisión entre los profesores doctores y los profesores no doctores. La interrelación binaria N:M **Supervisa** entre la entidad *DOCTOR* y *NO_DOCTOR* indica que un doctor es o ha sido supervisor de cero o varios profesores no doctores, mientras que un profesor no doctor ha podido tener a lo largo del tiempo varios profesores doctores como supervisores. Se ha incluido un atributo multivaluado compuesto para representar los períodos en los que un profesor doctor supervisó a un profesor *no_doctor*. La fecha de finalización de la supervisión se ha supuesto opcional, ya que podría ser desconocida. En este caso no se podrá recoger la semántica de que un profesor no doctor nunca podrá tener a varios profesores doctores como supervisores de forma simultánea. Tampoco se ha podido representar que la fecha de inicio de supervisión de un profesor a otro implica una única fecha de fin y, en el caso de que ésta fuera conocida, una fecha de finalización implicaría una única fecha de inicio de supervisión de un profesor a otro. Además, la fecha de inicio de supervisión siempre ha de ser anterior a la fecha de fin de supervisión.

La Figura 1.24 muestra el diagrama E/R de los supuestos semánticos comentados en esta parte.

La semántica asociada al investigador principal de un proyecto también se podría haber modelado según aparece en la Figura 1.25. En esta figura se ha incluido la información de qué profesor es investigador principal en la interrelación *Participa* mediante un atributo univaluado *Investigador_Principal* que toma valores booleanos (con esta solución no es necesaria la interrelación **Investigador_principal**). Esta manera de modelar la semántica tiene sus

ventajas y desventajas frente a la anterior forma de modelarlo. En este caso no sería necesario comprobar que los profesores solo son investigadores principales de los proyectos de investigación en los que participan, pero sería necesario comprobar que existe un único investigador principal por proyecto de investigación y que éste es doctor. Ambas formas de modelado son correctas, aunque nosotros proponemos la de incluir la interrelación **Investigador_principal**, ya que se recogen más supuestos semánticos con el diagrama E/R y además existiría una única ocurrencia por proyecto de investigación en la interrelación, mientras que tratándose de un atributo en la interrelación **Participa**, habría que indicar para cada profesor que participe en el proyecto si es o no investigador principal. Si suponemos que existen muchos profesores trabajando en el mismo proyecto de investigación, la opción del atributo *Investigador_principal* supondría utilizar más recursos de espacio en la máquina y el acceso/mantenimiento de esta información de forma menos eficiente (habría que consultar todos los profesores que participan en el proyecto y ver cuál de ellos (solo lo puede tener uno) tiene el valor *Verdadero* en el atributo *Investigador_principal*.

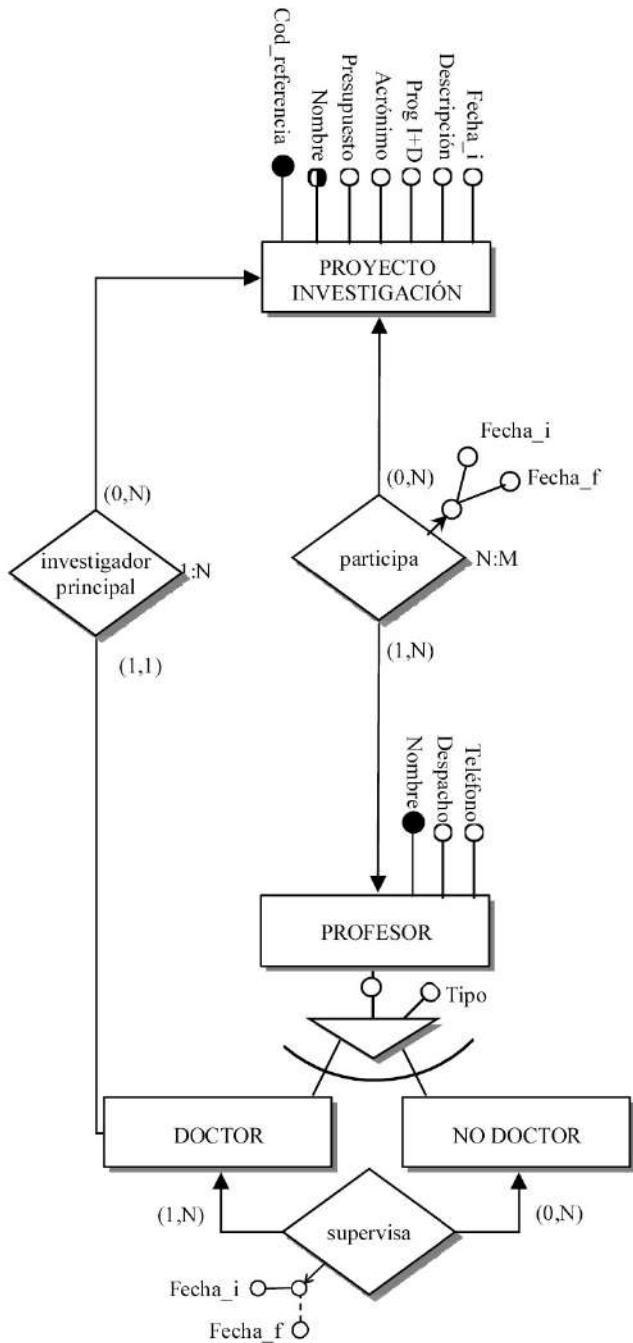


Figura 1.24. Diagrama E/R correspondiente a la Parte 1

PARTE 2

Por otro lado, los proyectos de investigación producen una serie de publicaciones sobre las que también interesa guardar información. Una publicación se caracteriza por un número en secuencia dentro de cada

proyecto de investigación y se guardará el título y los profesores que la han escrito; las publicaciones son de dos tipos, publicaciones en congresos y publicaciones en revista; de las primeras se almacenará el nombre del congreso, su tipo (nacional o internacional), la fecha de inicio y de fin, el lugar de celebración, país y la editorial que ha publicado las actas del congreso (si es que se han publicado); de las publicaciones en revista interesa saber el nombre de la revista, la editorial, el volumen, el número y las páginas de inicio y fin.

También se desea almacenar información sobre las publicaciones que surgen en los proyectos de investigación, por lo que se añade la entidad *PUBLICACIÓN* al esquema E/R. Las publicaciones dependen en identificación de la entidad *PROYECTO_INVESTIGACIÓN*, por lo que su atributo identificador principal (AIP) estará formado por el AIP de la entidad *PROYECTO_INVESTIGACIÓN* unido a un número de secuencia, creándose la interrelación en identificación **Produce**. Además, la entidad *PUBLICACIÓN* tendrá asociado como atributo univaluado y obligatorio el Título de la publicación. También será necesario almacenar el tipo de publicación que se trata (publicación en congreso o publicación en revista). En este párrafo se indica claramente que las publicaciones en congreso y en revista poseen propiedades comunes (como el título) y propiedades diferentes. Por ello, se ve necesario crear una generalización con estos conceptos, surgiendo los subtipos *CONGRESO* y *REVISTA* para la jerarquía de *PUBLICACIÓN*. La jerarquía será total (ya que no se tomarán en consideración más tipos de publicaciones) y exclusiva (ya que un artículo en congreso nunca será un artículo en revista). El atributo discriminante de la jerarquía será el atributo *Tipo* de publicación, que tomará valores en el dominio *TIPO_PUBLICACIÓN*={congreso, revista}. El subtipo *CONGRESO* tendrá como atributos el *Tipo* de congreso (tomando valores en el dominio *TIPO_CONGRESO*={nacional, internacional}), *Nombre* del congreso, la *Fecha_inicio* y *Fecha_fin*, el *Lugar* de celebración, el *País* y la *Editorial* que publicó las actas, todos ellos univaluados y obligatorios, excepto la *Editorial*, que será opcional. Por otro lado, se indica que de las publicaciones se desea almacenar el *Nombre* de la revista, el *Volúmen*, el *Número*, la *Página_inicio* y *Página_fin* del artículo. En este momento nos damos cuenta de que el *Nombre* es un atributo común de los dos subtipos de la jerarquía, por lo que se convertirá en un atributo del supertipo de la jerarquía.

El tipo de congreso se podría haber modelado erróneamente como una jerarquía donde el supertipo sería la entidad *CONGRESO* y los subtipos serían

las entidades *NACIONAL* e *INTERNACIONAL*. En este caso no se modelaría como jerarquía ya que los conceptos *CONGRESO_NACIONAL* y *CONGRESO_INTERNACIONAL* no poseen atributos propios ni interrelaciones con otras entidades, por lo que sería más correcto modelar el tipo de congreso como un atributo de la entidad *CONGRESO*.

Para poder indicar qué profesores han escrito dicha publicación será necesario añadir una interrelación al esquema E/R. En principio podríamos suponer que se trata de una interrelación binaria entre las entidades *PROFESOR* y *PUBLICACIÓN*, pero en los siguientes párrafos veremos que será necesario también tener en cuenta la línea de investigación en la que trabajó cada profesor en la publicación, por lo que se convertirá en una interrelación ternaria. Se verá a continuación.

En la Figura 1.26 se muestra el diagrama E/R de los supuestos semánticos comentados.

En este diagrama no se ha podido recoger la semántica de que en una revista la página de inicio siempre ha de ser anterior o igual a la página de fin. De la misma forma, la fecha de inicio de un congreso siempre ha de ser anterior o igual a la fecha de finalización del mismo. Además, la fecha de inicio de la publicación en congreso debería ser posterior o igual a la fecha de inicio del proyecto de investigación.

PARTE 3

No solamente interesa conocer los profesores que han participado en las publicaciones de los proyectos de investigación si no también las líneas de publicación que cubren estas publicaciones. Una línea de investigación se identifica por un código, un nombre (por ejemplo, “recuperación de información multilingüe”, “bases de datos espacio-temporales”, etc.) y un conjunto de descriptores (por ejemplo, la línea de investigación “bases de datos temporales” puede tener como descriptores “Bases de Datos”, “SGBD Relacional, “Dimensión temporal”).

Los profesores tendrán asociados en la BD las líneas de investigación en las que trabajan incluso podría ocurrir que hubiera profesores que no tuvieran ninguna línea asignada.

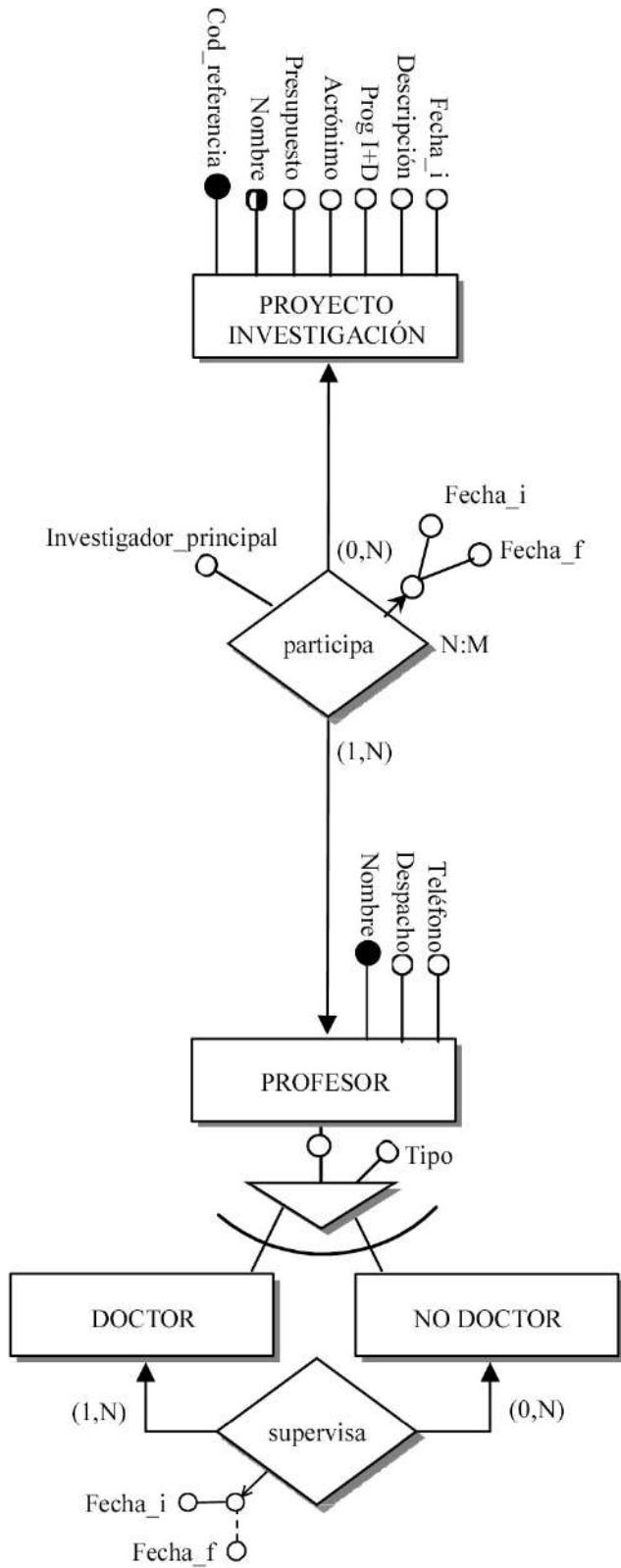


Figura 1.25. Diagrama E/R alternativo correspondiente a la Parte 1

Así, tanto los profesores doctores como los no doctores pueden escribir publicaciones sobre una o más líneas de investigación y nos interesa saber sobre qué línea de investigación ha escrito un determinado profesor en una publicación, teniendo en cuenta que un profesor que participa en una publicación solo escribe en el ámbito de una línea de investigación y que una determinada publicación puede cubrir varias líneas de investigación

Por último, aparte de la información especificada para los proyectos de investigación también se almacenarán las líneas de investigación que abarca cada proyecto.

De estos supuestos semánticos surge una nueva entidad, **LÍNEA_INVESTIGACIÓN**. Ésta se identifica mediante un código propio, y posee como propiedades un nombre y un conjunto de descriptores, representándose los descriptores como un atributo multivaluado.

Un profesor puede desarrollar cero o varias líneas de investigación, mientras que una línea de investigación puede ser desarrollada por cero o varios profesores. La interrelación binaria **Desarrolla**, que surge de este supuesto semántico, tendrá como tipo de correspondencia N:M.

Como se adelantó en párrafos anteriores, también se desea almacenar información sobre qué profesor ha escrito qué publicación y sobre qué línea de investigación ha escrito en dicha publicación. Para ello será necesario incluir en el esquema una interrelación ternaria, **Escribe**. Dicha interrelación indicará que un profesor ha escrito sobre una determinada línea de investigación en varias publicaciones; además, también indicará que en una publicación varios profesores pudieron escribir sobre la misma línea de investigación; por último, la interrelación también representa que un profesor en una publicación únicamente puede escribir sobre una línea de investigación. Por lo tanto, el tipo de correspondencia de la interrelación ternaria **Escribe** es 1:N:M.

Con la interrelación ternaria **Escribe** se almacena información sobre las líneas de investigación de cada publicación, pero no se almacena información sobre qué líneas de investigación cubre un determinado proyecto de investigación (podrían existir proyectos de investigación de los que todavía no se ha escrito ningún artículo). De igual forma, la interrelación ternaria tampoco almacenaría toda la información acerca de las líneas de investigación que desarrollan los profesores, ya que podría ocurrir que un profesor aún no haya escrito ningún artículo de alguna de sus líneas de investigación. Por ello será

necesario incluir las interrelaciones binarias **Incluye** y **Desarrolla** con las entidades *PUBLICACIÓN* y *PROFESOR* ya que, al contrario de como se podría suponer, no son interrelaciones redundantes.

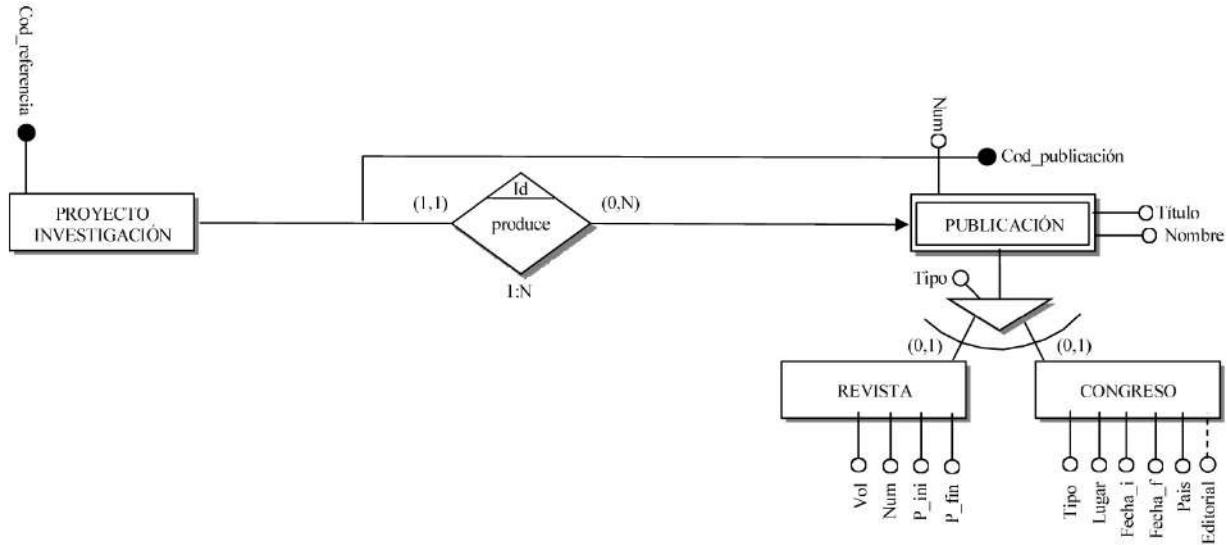


Figura 1.26. Diagrama E/R correspondiente a la Parte 2

Los supuestos semánticos comentados se representan en el diagrama E/R de la Figura 1.27. En diagrama E/R no se ha podido recoger la información de que un profesor solo puede escribir publicaciones sobre líneas de investigación que desarrolla. Además, un profesor no puede escribir en una publicación de un proyecto de investigación si éste no trabaja en el mismo. Por otro lado, las líneas de investigación de una determinada publicación han de encontrarse como ocurrencias en la interrelación **Incluye**. Así mismo, un proyecto de investigación no podrá incluir líneas de investigación que no desarrolle los profesores que participan en él.

SOLUCIÓN PROPUESTA

La Figura 1.28 muestra el diagrama E/R completo que, junto con los supuestos semánticos que no se han podido representar en el diagrama, componen el esquema E/R del ejercicio.

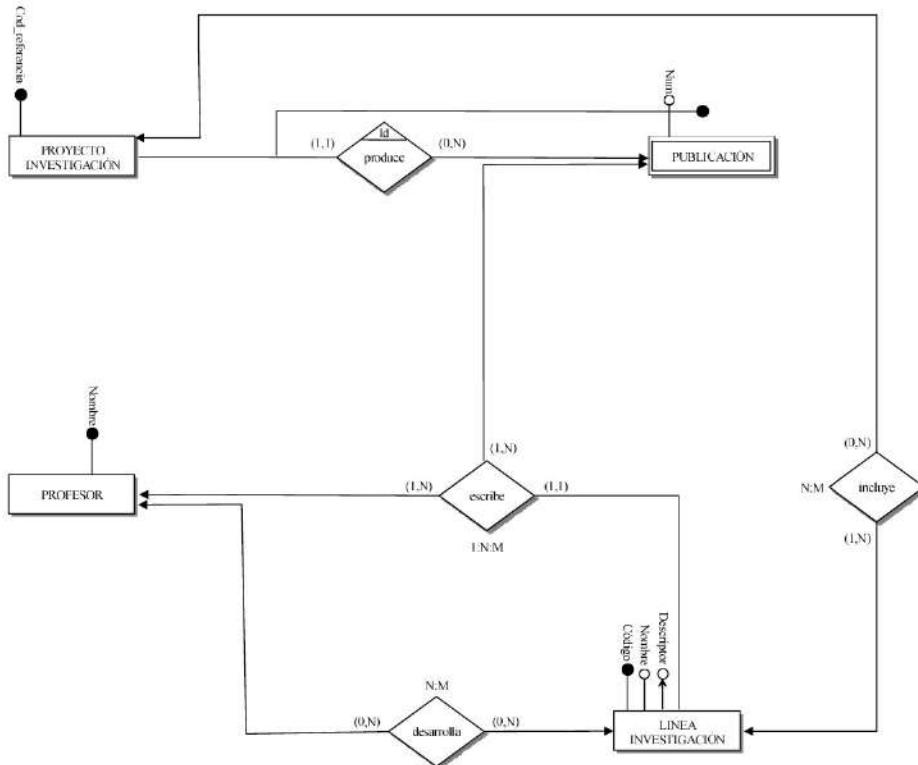


Figura 1.27. Diagrama E/R correspondiente a la Parte 3

DIAGRAMA E/R

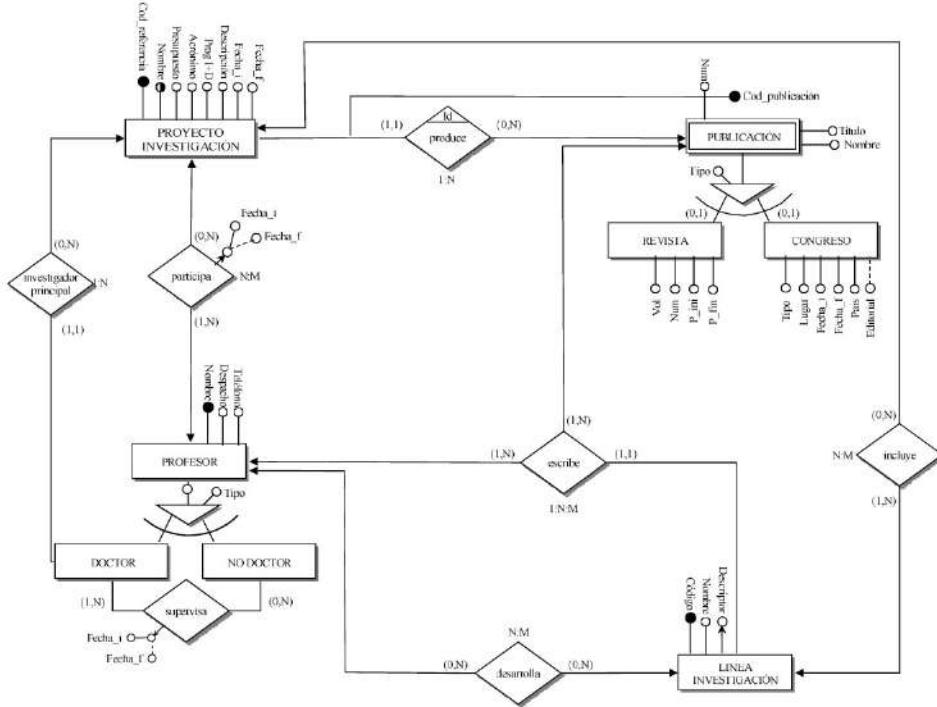


Figura 1.28. Diagrama E/R Propuesto para la gestión de proyectos de investigación

SUPUESTOS ASOCIADOS AL DIAGRAMA E/R

A continuación se resume la semántica que no ha sido posible incorporar al diagrama E/R de la Figura 1.28.

- SS1. No se pueden superponer los períodos de tiempo en los que un profesor participa en un proyecto dado.
- SS2. La fecha de inicio de participación en un proyecto siempre ha de ser anterior a la fecha de finalización.
- SS3. La fecha de inicio de participación de un profesor en un proyecto de investigación implica una única fecha de finalización de la participación. De igual manera, si la fecha de finalización fuera conocida, ésta implicaría una única fecha de inicio de participación en el proyecto para cada profesor.
- SS4. Un profesor no doctor nunca podrá tener a varios profesores doctores como supervisores de forma simultánea.
- SS5. La fecha de inicio de supervisión implica una única fecha de fin y, en el caso de que ésta fuera conocida, una fecha de finalización implicaría una

única fecha de inicio de supervisión de un profesor a otro.

SS6. La fecha de inicio de supervisión de un profesor doctor a un profesor no doctor siempre ha de ser anterior a la fecha de fin de supervisión.

SS7. Los profesores solo pueden ser investigadores principales de los proyectos de investigación en los que participan.

SS8. Un profesor solo puede escribir publicaciones sobre líneas de investigación que desarrolla.

SS9. Un profesor no puede escribir en una publicación de un proyecto de investigación si éste no trabaja en el mismo.

SS10. Un proyecto de investigación solo puede incluir líneas de investigación de los profesores que trabajan en el proyecto.

SS11. Las líneas de investigación de una determinada publicación han de encontrarse como ocurrencias en la interrelación Incluye entre las entidades *LÍNEA_INVESTIGACIÓN* y *PROYECTO_INVESTIGACIÓN*.

SS12. La página de inicio de una publicación en una revista siempre ha de ser anterior a la página de fin de la misma.

SS13. La fecha de inicio de un congreso siempre ha de ser anterior o igual a la fecha de fin del mismo.

SS14. La fecha de inicio de un proyecto de investigación siempre ha de ser anterior o igual a la fecha de fin del mismo.

SS15. La fecha de inicio del proyecto de investigación siempre ha de ser anterior o igual a la fecha de inicio de la publicación en un congreso.

SS16. El atributo discriminante *Tipo* de la jerarquía de *PUBLICACIÓN* toma valores en el dominio *TIPO_PUBLICACIÓN*={congreso, revista}.

SS17. El atributo *Tipo* de la entidad *CONGRESO* toma valores en el dominio *TIPO_CONGRESO*={nacional, internacional}.

SS18. El atributo discriminante *Tipo* de la jerarquía *PROFESOR* toma valores en el dominio *TIPO_PROFESOR*={doctor, no_doctor}.

PROBLEMA 1.5: VIAJES DE INVESTIGACIÓN

El diagrama E/R de la Figura 1.29 corresponde a una parte del diseño de una BD para almacenar la información relativa a los gastos de los viajes de los profesores de la universidad concernientes a la investigación que realizan.

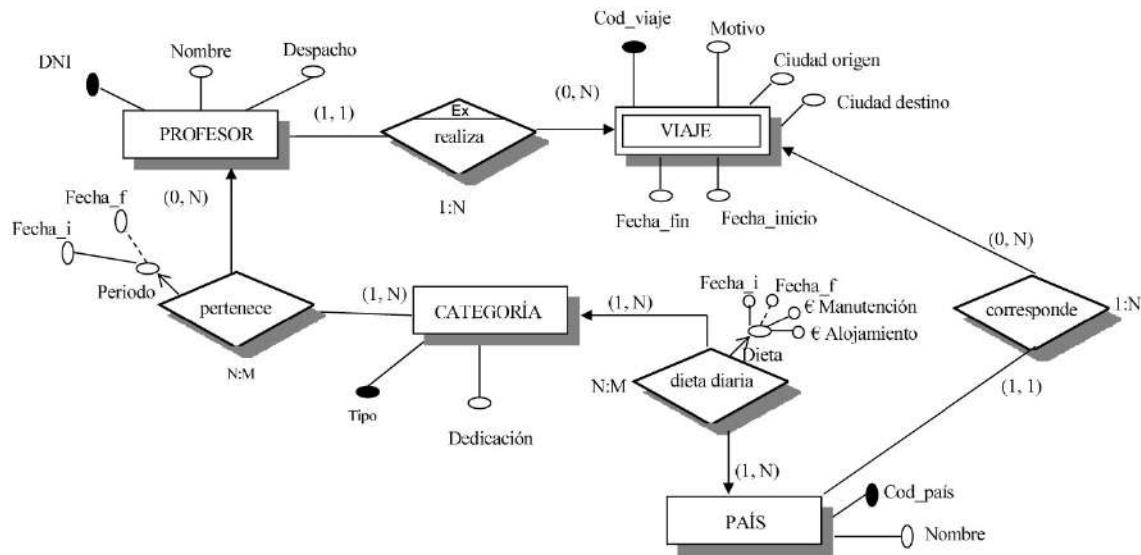


Figura 1.29. Diagrama E/R inicial sobre gestión de viajes en la universidad

Se pide:

Extender y/o modificar el esquema E/R de la Figura 1.29 para recoger los siguientes supuestos semánticos (indicar aquellos supuestos que no hayan podido reflejarse en la solución propuesta):

DISCUSIÓN DEL ENUNCIADO

PARTE 1

En el diagrama de la Figura 1.29 no se han podido recoger algunos supuestos semánticos que se comentan a continuación, completando de este modo el esquema E/R inicial del problema:

Es necesario tener en cuenta que la fecha de inicio en la que un profesor pertenece a una determinada categoría siempre ha de ser anterior o igual a la fecha de finalización (en el caso de que sea un dato conocido).

No se pueden solapar los períodos de tiempo en los que un profesor pertenece a una determinada categoría. Esto implica que la fecha de inicio en la que un profesor pertenece a una determinada categoría implica una única fecha de fin. También ocurre que si la fecha de fin es conocida, ésta implicará una única fecha de inicio de pertenencia a la categoría.

De igual forma, en los datos sobre las dietas diarias según la categoría y el país destino del viaje, es necesario comprobar que la fecha de inicio de las dietas siempre ha de ser anterior o igual a la fecha de finalización de las mismas.

No se pueden solapar los períodos de tiempo en los que se fijan las dietas diarias para una misma categoría y país.

Se ha de comprobar también que en el periodo de tiempo en el que se fijan las dietas diarias, la fecha de inicio del periodo implica un único valor de fecha de finalización, de euros para manutención y euros para alojamiento. La fecha de finalización de un periodo de dietas diarias también implican una única fecha de inicio, euros para manutención y euros para alojamiento.

Otro supuesto semántico que no se ha podido representar en el esquema E/R es que un profesor ha de pertenecer a alguna categoría en el momento en el que está realizando el viaje.

También será necesario siempre tener en la base de datos información sobre las dietas diarias en las fechas en las que se han llevado a cabo viajes.

La fecha de fin de un viaje siempre ha de ser posterior o igual a la fecha de inicio del mismo.

PARTE 2

Cada viaje realizado genera una serie de gastos que pueden ser de varios tipos (manutención, transporte y alojamiento) y que el profesor que realiza el viaje debe justificar a su finalización. A cada gasto le corresponde un número en secuencia dentro de cada viaje y tiene un importe. Para los gastos de transporte se almacena el medio de transporte utilizado (tren, avión, taxi, etc.) y el número de medios de transporte de manera que puedan agruparse los de un mismo tipo (por ejemplo, tres recibos de taxi con un importe total de 50 €); para los gastos de alojamiento se almacenará el número de noches y el tipo de habitación (simple, doble, triple, etc.) ocupada en el hotel y para la manutención solo se almacena el número de días para el que se solicita manutención (que no tiene porqué coincidir con la duración del viaje) pues su importe ya está controlado en la interrelación Dieta Diaria según el país de destino y la categoría del profesor que realiza el viaje. Nunca es posible superar la dieta diaria establecida para cada categoría y país de destino del viaje.

Se tendrá en cuenta, por tanto, la entidad *GASTO* que depende en identificación de la entidad *VIAJE*. El atributo identificador principal (AIP) de *GASTO* estará formado por el AIP de la entidad *VIAJE* junto a un número de secuencia.

Por otro lado, existen tres tipos de gastos, siendo necesario almacenar para cada uno de ellos propiedades diferentes. Por ello, se creará una jerarquía donde la entidad *GASTO* es el supertipo de la jerarquía y las entidades *MANUTENCIÓN*, *TRANSPORTE* y *ALOJAMIENTO* son los subtipos de la misma. La jerarquía es total (ya que no existen otro tipo de gastos de viaje) y exclusiva (ya que un gasto de un tipo nunca lo podrá ser de otro tipo al mismo tiempo). El atributo discriminante de la jerarquía será el *Tipo* de gasto, que tomará valores en el dominio {manutención, transporte, alojamiento}.

En los gastos de transporte se almacena información sobre el *Medio* de transporte, el *Número de recibos* que se tienen de ese medio y el *Importe_T* del gasto de transporte, que incluirá la suma total de todos los recibos en ese medio de transporte. En el enunciado indican explícitamente que se han de agrupar todos los recibos de un determinado medio de transporte en una sola ocurrencia de *GASTO*, sin embargo en el diagrama E/R de la Figura 1.30 ha sido imposible representar la semántica de que existe un único gasto de transporte por medio de locomoción en un viaje.

En los gastos de alojamiento, por otro lado, se desea almacenar información sobre el *Número de noches* hospedado, el *Tipo de habitación* que se ocupó y el *Importe_A* de gasto de alojamiento. No se ha podido representar en el diagrama el supuesto semántico de que el importe de alojamiento nunca podrá superar la dieta diaria establecida para cada categoría de profesor y país destino del viaje en las fechas en las que se ha realizado el mismo.

Por último, en los gastos de manutención se desea almacenar información sobre el *Número de días* en el que se tendrá en cuenta la manutención (que no tienen por qué coincidir con el número de días del viaje). Además, y por motivos de eficiencia a la hora de consultar la base de datos, se podría incluir en la entidad un atributo derivado denominado *Importe_M* cuyo valor es el cálculo de multiplicar *Número_días* del gasto de manutención por la dieta diaria de manutención en las fechas que se realizó el viaje. La función con la que se calcula el atributo derivado *Importe_M* no se ha podido representar en el diagrama. De igual forma, tampoco se ha podido representar el supuesto semántico de que el valor del atributo *Número_días* tenido en cuenta en los gastos de manutención ha de ser igual o menor al número de días del viaje (restando el atributo *Fecha_fin* al atributo *Fecha_inicio* de la entidad VIAJE).

La Figura 1.30 muestra el diagrama E/R de los supuestos semánticos comentados.

Es importante tener en cuenta en este ejercicio un detalle de modelado en la jerarquía. Aunque todos los subtipos de la jerarquía poseen un atributo *Importe* del gasto, éste no puede considerarse como un atributo común y, por tanto, definirlo como una propiedad del supertipo, ya que en cada caso el valor del atributo ha de cumplir diferentes características e incluso en el caso de los gastos de manutención, éste es derivado.

PARTE 3

También es necesario almacenar cómo se financian los gastos de un viaje. Un viaje puede cubrirlo o bien el Departamento de la universidad al que pertenece el profesor o bien un Proyecto de Investigación en el que trabaje como investigador el profesor que realiza el viaje. No puede haber financiaciones parciales de los viajes, es decir, que unos gastos estén cubiertos por un Proyecto de investigación y otros por un departamento. De un departamento se almacena el nombre del departamento, el director y el dinero que tiene para sufragar los viajes de sus profesores; de cada

proyecto de investigación se guarda un código, un nombre, el investigador principal, el organismo que lo financia y el dinero de que dispone para viajes, lo que permitirá saber si es posible pagar el viaje o no. Tanto los investigadores principales de los proyectos de investigación como los directores de los departamentos de la universidad serán profesores de la misma. Se desea almacenar información sobre quién era el director del departamento cuando se firmó la concesión de gastos sobre un determinado viaje. De igual forma, también se desea almacenar información sobre quién era el investigador principal del proyecto cuando se firmó la financiación de cada viaje.

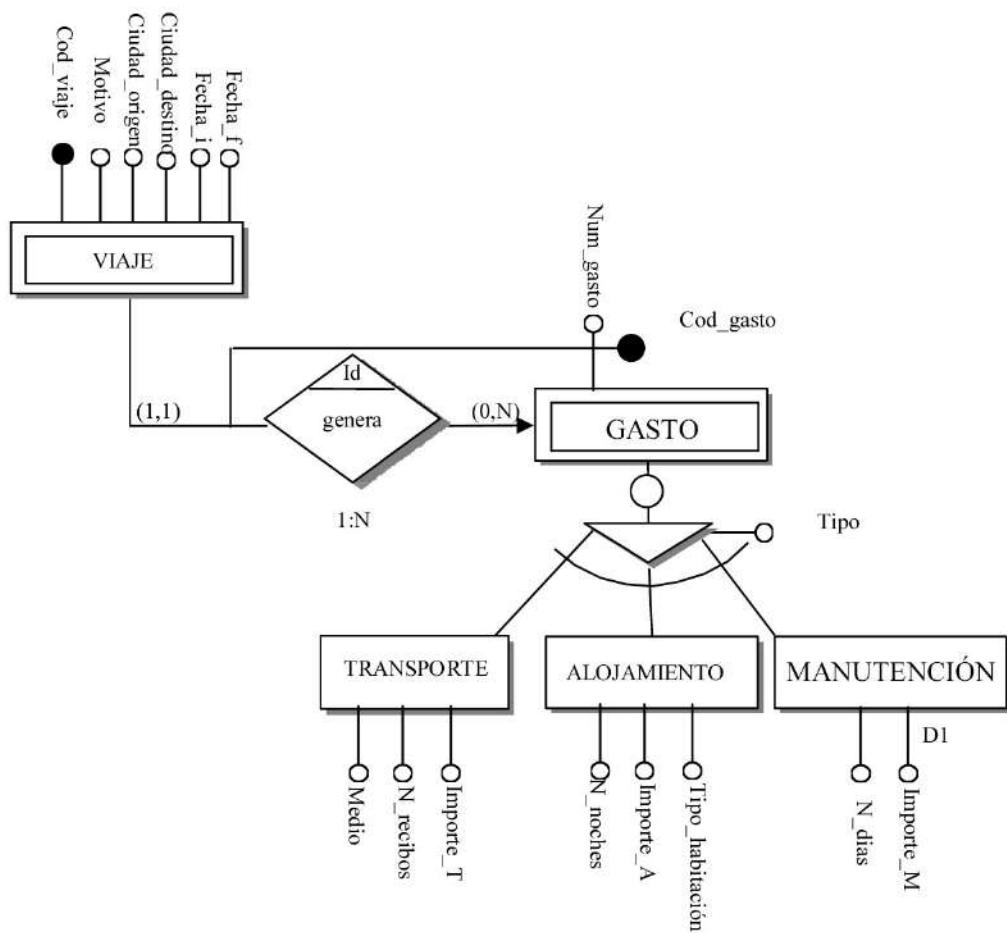


Figura 1.30. Diagrama E/R sobre gastos de viaje

El gasto del viaje de un determinado profesor lo puede cubrir o bien el departamento donde se encuentra adscrito el profesor, o bien un proyecto de investigación donde esté trabajando. El viaje se paga de forma completa por uno o por otro, pero nunca se compartirán gastos. Por ello, se incluyen en el

diagrama dos interrelaciones exclusivas: la interrelación **Financiado** entre las entidades *VIAJE* y *PROYECTO_INVESTIGACIÓN* y la interrelación **Sufragado** entre las entidades *VIAJE* y *DEPARTAMENTO*. Estas interrelaciones indican que un viaje puede ser financiado por ningún o un proyecto de investigación y puede ser sufragado por ningún o un departamento, pero si es financiado por un proyecto de investigación no puede ser sufragado por un departamento y viceversa.

Será necesario comprobar que cuando un departamento de la universidad sufraga un viaje es porque el profesor que realiza el viaje estaba trabajando en el mismo en ese momento. Para ello será necesario añadir información acerca de los periodos de tiempo que ha estado cada profesor adscrito a un determinado departamento de la universidad. La interrelación **Adscrito** almacena la información de cuándo un profesor ha estado trabajando en un determinado departamento. Un profesor ha podido estar trabajando en varios departamentos de la universidad, pero siempre en períodos distintos de tiempo. También puede ocurrir que un profesor haya trabajado en el mismo departamento en varios períodos de tiempo, por lo que se modelarán los atributos de *Fecha_inicio* y *Fecha_fin* como atributos multivaluados en la interrelación **Adscrito**.

Como supuestos semánticos que no se han podido reflejar en el diagrama E/R, se debe comprobar que:

- El valor de *Fecha_fin* (en el caso de que sea conocido) en la interrelación Adscrito es siempre igual o posterior al valor de la *Fecha_inicio*.
- Un profesor no puede estar adscrito a distintos departamentos en el mismo momento.
- La fecha de inicio en la que un profesor trabaja en un determinado departamento implica una única fecha de finalización. De igual forma ocurre con la fecha de fin (en el caso de que sea conocida), implicaría una única fecha de inicio de adscripción a un departamento.
- Las fechas del viaje (o al menos la fecha de inicio del viaje) ha de coincidir con las fechas en la que el profesor estaba adscrito al departamento.
- Durante el periodo en el que el profesor se encuentra adscrito a un departamento, éste pertenece a alguna categoría de profesorado.
- Todo viaje ha de ser financiado o sufragado por un proyecto de

investigación o un departamento (no se puede quedar sin ser financiado).

También será necesario comprobar que si un proyecto de investigación financia un viaje es porque el profesor que realiza el viaje estaba trabajando en el mismo en ese momento. Para ello será necesario añadir información histórica sobre en qué proyectos de investigación ha trabajado cada profesor a lo largo de su vida académica, pudiendo ocurrir que un profesor haya trabajado en varios períodos de tiempo en el mismo proyecto. Además, un profesor puede trabajar en distintos proyectos al mismo tiempo. La interrelación **Trabaja** almacena esta información en el diagrama.

Los supuestos semánticos que no se han podido reflejar en el diagrama E/R se comentan a continuación:

- El valor de *Fecha_fin* (en el caso de que sea conocido) en la interrelación Trabaja es siempre igual o posterior al valor de la *Fecha_inicio*.
- Las fechas del viaje (o al menos la fecha de inicio del viaje) ha de coincidir con las fechas en la que el profesor ha estado trabajando en el proyecto de investigación.
- Durante el periodo de trabajo en un determinado proyecto de investigación, el profesor pertenece a alguna categoría de profesorado.
- Durante el periodo de trabajo en el proyecto el profesor está adscrito a algún departamento de la universidad.

En cuanto a los investigadores principales de los proyectos de investigación, será necesario introducir en el diagrama una nueva interrelación denominada **Investigador_principal** que indique qué profesor ha sido investigador principal de cada proyecto en cada momento. Incluyendo el atributo multivaluado compuesto indicando la fecha de inicio en la que el profesor fue investigador principal del proyecto, se indica que un mismo profesor ha podido ser investigador principal del mismo proyecto en varias ocasiones. Con esta interrelación no se ha podido incorporar al diagrama el supuesto semántico de que en un momento dado solo un profesor puede ser investigador principal. Tampoco se ha podido reflejar que la fecha de inicio en la que un profesor es investigador principal de un proyecto siempre ha de ser anterior o igual a la fecha de fin. Además, es necesario tener en cuenta que la fecha de inicio implica

una única fecha de fin y, si ésta fuese conocida, implicaría una única fecha de inicio. Será también necesario comprobar que en las fechas en el que el profesor ha sido investigador principal de un proyecto, éste se encontraba trabajando en el mismo.

Para almacenar información sobre el director de los departamentos de la universidad, también será necesario reflejarlo con una nueva interrelación **Director** y con un atributo multivaluado compuesto donde se indica el periodo en el que ha sido director del mismo. Los supuestos semánticos que no se pueden recoger en el diagrama y las diferentes formas de modelarlo son similares al caso anterior, por lo que no se comentan de nuevo.

Por último, tanto la entidad *PROYECTO_INVESTIGACIÓN* como la entidad *DEPARTAMENTO* poseen un atributo que indican el dinero de que disponen para gastos de viajes. En ambos casos será necesario comprobar que el total de gastos asociados a los viajes que financian nunca será superior al dinero de que disponen. Este es un supuesto semántico que no se puede reflejar en el diagrama E/R.

La Figura 1.31 muestra el diagrama E/R de los supuestos comentados anteriormente:

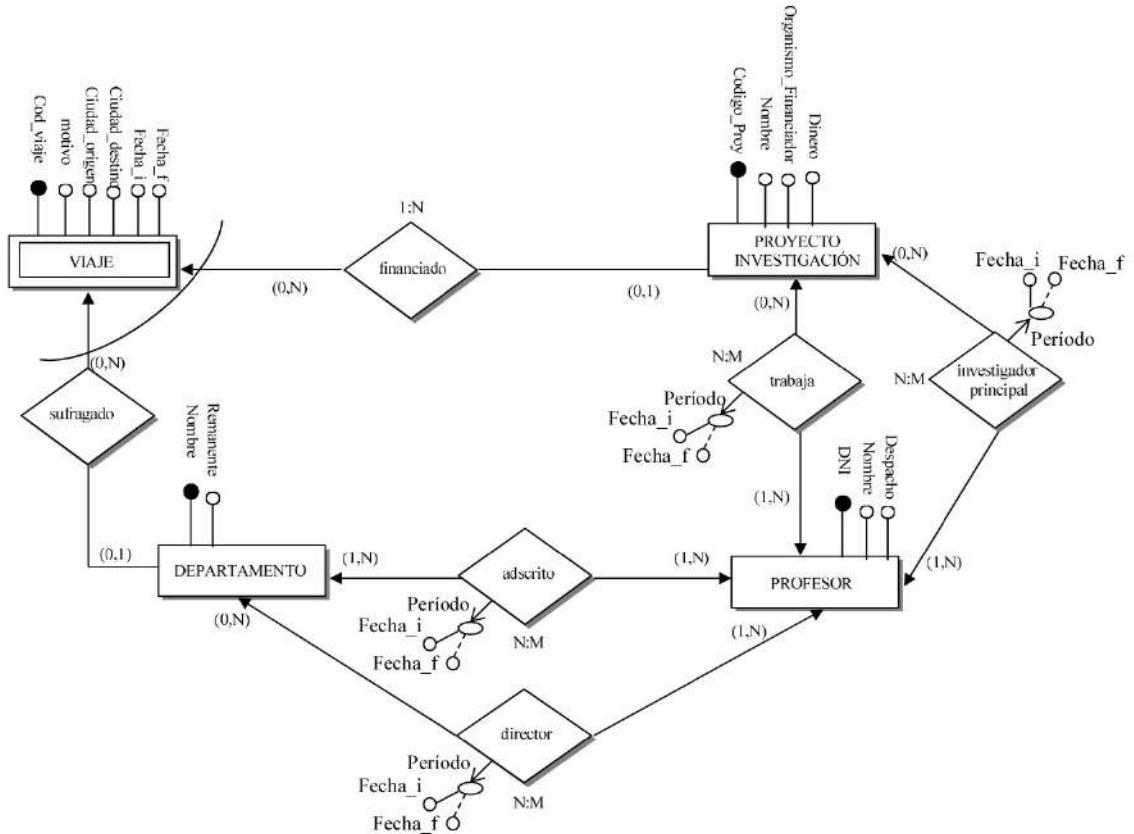


Figura 1.31. Diagrama E/R sobre financiación de viajes

SOLUCIÓN PROPUESTA

DIAGRAMA E/R:

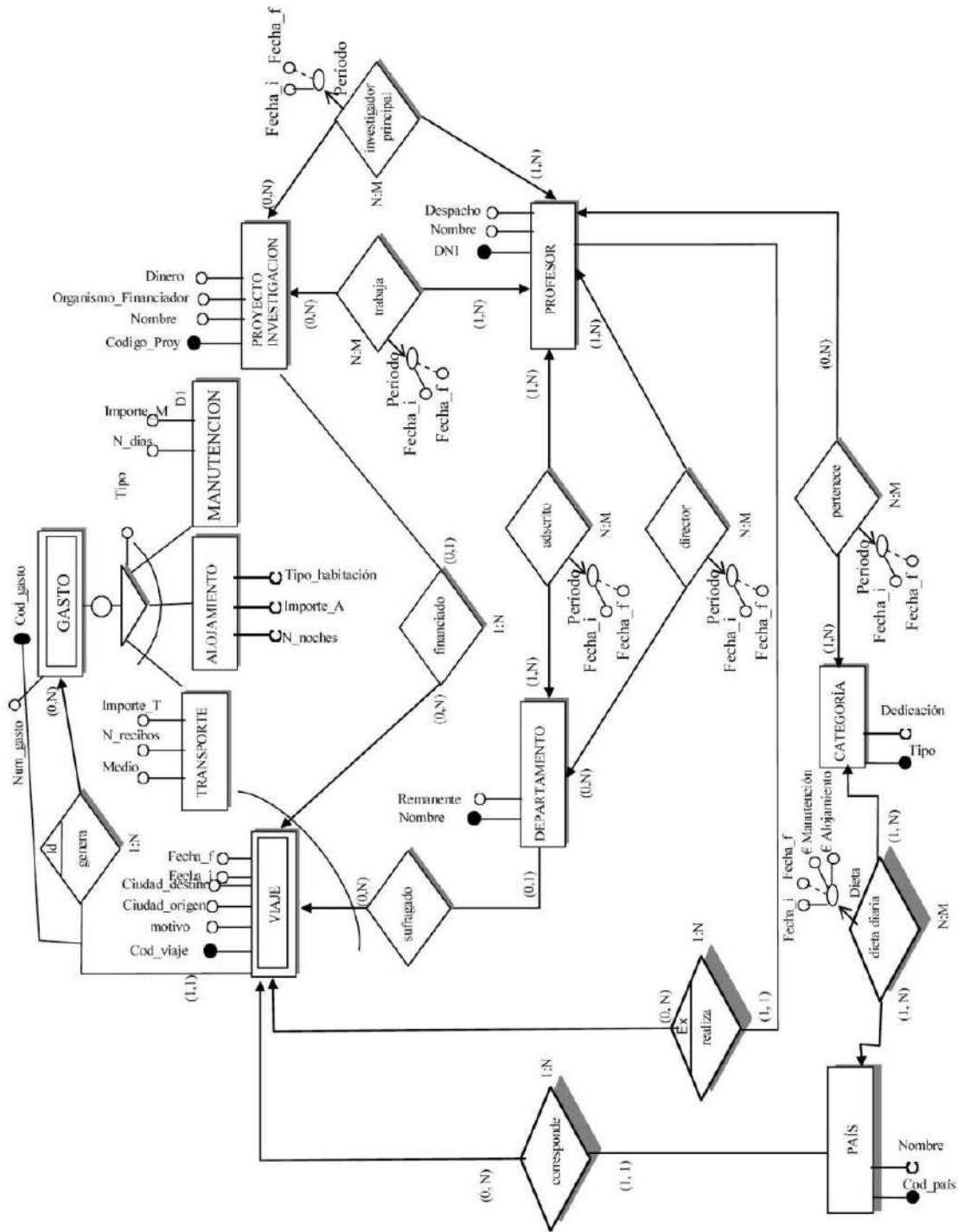


Figura 1.32. Diagrama E/R propuesto para la gestión de viajes de investigación

SUPUESTOS ASOCIADOS AL DIAGRAMA E/R:

SS1. Es necesario tener en cuenta que la fecha de inicio en la que un profesor

pertenece a una determinada categoría siempre ha de ser anterior o igual a la fecha de finalización (en el caso de que sea un dato conocido).

- SS2. No se pueden solapar los periodos de tiempo en los que un profesor pertenece a una determinada categoría. Esto implica que la fecha de inicio en la que un profesor pertenece a una determinada categoría implica una única fecha de fin. También ocurre que si la fecha de fin es conocida, ésta implicará una única fecha de inicio de pertenencia a la categoría.
- SS3. En los datos sobre las dietas diarias según la categoría y el país destino del viaje, es necesario comprobar que la fecha de inicio de las dietas siempre ha de ser anterior o igual a la fecha de finalización de las mismas.
- SS4. No se pueden solapar los periodos de tiempo en los que se fijan las dietas diarias para una misma categoría y país.
- SS5. En el periodo de tiempo en el que se fijan las dietas diarias, la fecha de inicio del periodo implica un único valor de fecha de finalización, de euros para manutención y euros para alojamiento. La fecha de finalización de un periodo de dietas diarias también implican una única fecha de inicio, euros para manutención y euros para alojamiento.
- SS6. Un profesor ha de pertenecer a alguna categoría en el momento en el que está realizando el viaje.
- SS7. La fecha de fin de un viaje siempre ha de ser posterior o igual a la fecha de inicio del mismo.
- SS8. El atributo *Tipo* de la entidad *GASTO* toma valores en el dominio {manutención, transporte, alojamiento}.
- SS9. Existe una única ocurrencia en gasto de transporte por cada medio de transporte.
- SS10. El importe de alojamiento nunca podrá superar la dieta diaria establecida para cada categoría de profesor y país destino del viaje en las fechas en las que se ha realizado el mismo.
- SS11. El atributo derivado *Importe_M* de la entidad *MANUTENCIÓN* se calcula multiplicando el número de días del gasto de manutención por la dieta diaria de manutención en las fechas que se realizó el viaje.
- SS12. El valor del atributo *Número_días* tenido en cuenta en los gastos de manutención ha de ser igual o menor al número de días del viaje (restando

el atributo *Fecha_fin* al atributo *Fecha_inicio* de la entidad *VIAJE*.

SS13. El valor de *Fecha_fin* (en el caso de que sea conocido) en la interrelación *Adscrito* es siempre igual o posterior al valor de la *Fecha_inicio*.

SS14. Un profesor no puede estar adscrito a dos departamentos al mismo tiempo. Esto implica que la fecha de inicio en la que un profesor trabaja en un determinado departamento determina una única fecha de finalización. De igual forma ocurre con la fecha de fin (en el caso de que sea conocida), implicaría una única fecha de inicio de adscripción a un departamento.

SS15. Las fechas del viaje (o al menos la fecha de inicio del viaje) ha de coincidir con las fechas en la que el profesor estaba adscrito al departamento que sufraga el viaje.

SS16. Durante el periodo en el que el profesor se encuentra adscrito a un departamento, éste pertenece a alguna categoría de profesorado.

SS17. El valor de *Fecha_fin* (en el caso de que sea conocido) en la interrelación *Trabaja* es siempre igual o posterior al valor de la *Fecha_inicio*.

SS18. Las fechas del viaje (o al menos la fecha de inicio del viaje) ha de coincidir con las fechas en la que el profesor ha estado trabajando en el proyecto de investigación que financia el viaje.

SS19. Durante el periodo de trabajo en un determinado proyecto de investigación, el profesor pertenece a alguna categoría de profesorado (comprobado mediante SS16 y SS19).

SS20. Durante el periodo de trabajo en el proyecto el profesor está adscrito a algún departamento de la universidad.

SS21. En un momento dado solo un profesor puede ser investigador principal de un proyecto de investigación.

SS22. Al menos un profesor ha de trabajar en todo momento en un proyecto dado (al menos el director del proyecto).

SS23. La fecha de inicio en la que un profesor es investigador principal de un proyecto siempre ha de ser anterior o igual a la fecha de fin. Además, ésta implica una única fecha de fin y, si ésta fuese conocida, implicaría una única fecha de inicio.

SS24. En las fechas en el que el profesor ha sido investigador principal de un proyecto, éste se encontraba trabajando en el mismo.

SS25. En un momento dado solo un profesor puede ser director de un departamento.

SS26. La fecha de inicio en la que un profesor es director de un departamento ha de ser anterior o igual a la fecha de fin. Además, ésta implica una única fecha de fin y, si ésta fuese conocida, implicaría una única fecha de inicio.

SS27. En las fechas en el que el profesor ha sido director de un departamento, éste estaba adscrito al mismo.

SS28. Al menos un profesor ha de estar adscrito a cada departamento (al menos el director del mismo).

SS29. El total de gastos asociados a los viajes que financian los proyectos de investigación o que sufragan los departamentos nunca será superior al dinero de que disponen respectivamente.

SS30. Todo viaje ha de ser financiado o sufragado por un proyecto de investigación o un departamento (no se puede quedar sin ser financiado).

PROBLEMA 1.6: GESTIÓN DE PROYECTOS INFORMÁTICOS

Una empresa de consultoría desea crear una base de datos para facilitar la gestión de los proyectos informáticos que desarrolla para sus empresas clientes. Los requisitos que hay que recoger se muestran a continuación:

La empresa desarrolla proyectos de los que se almacena su código, nombre, cliente para el que se desarrolla el proyecto, una breve descripción, presupuesto, número de horas totales estimadas, fecha de inicio y fecha de fin. Cada proyecto se compone de una serie de fases identificadas por un número en secuencia en cada proyecto. Cada fase se caracteriza, además, por su nombre, fecha de comienzo, fecha de fin y estado en que se encuentra (en curso o finalizada).

Los empleados de la empresa (código, DNI, nombre, dirección, titulación y años de experiencia) están asignados a los proyectos que desarrolla la empresa. Interesa almacenar los empleados que son jefes de proyecto junto con su dedicación total en horas prevista a cada proyecto así como el coste de su participación en euros, teniendo en cuenta que cada proyecto lo lidera un único jefe de proyecto. En cuanto a los informáticos que participan en los proyectos, se quiere conocer los que son analistas y los que son programadores, así como el número de horas totales previstas dedicadas en cada proyecto y el coste en euros que supone cada dedicación. De los programadores se almacenarán también los lenguajes en los que son expertos.

En cada fase de un proyecto se generan una serie de productos (software, informes técnicos y prototipos) sobre los que se quiere guardar información. Cada producto previsto para una fase tiene un código, un nombre, una descripción, si está finalizado o no y tiene como responsable un analista. Además, la obtención de un producto puede abarcar varias fases de un proyecto y se quiere guardar los empleados informáticos involucrados en cada producto cuantificando el número de horas de trabajo dedicadas a cada producto teniendo en cuenta cada fase de un proyecto en la que se desarrolla. Para el software se almacena, además, su tipo (diagrama, programa, etc.) y para los prototipos su versión y su ubicación.

Por otro lado, a cada fase de un proyecto le corresponde una serie de recursos de los que se quiere conocer su código identificador, nombre, descripción, tipo (Hw o Sw) así como el periodo de tiempo que se utilizan en

cada fase.

Cada empleado, en cada proyecto en el que trabaja, puede originar una serie de gastos (dietas, viajes, alojamiento, etc.) que se cargan a los proyectos. Cada gasto se caracteriza por un código único, una descripción, una fecha, un importe y el tipo de gasto.

Por último, interesa conocer qué proyectos están relacionados con uno dado, es decir, cuáles son los proyectos que incluyen aspectos similares según distintas palabras clave (gestión de personal, gestión de stocks, etc.), con el fin de facilitar su desarrollo reutilizando algún componente.

Se pide:

Realizar el diagrama E/R extendido correspondiente a los supuestos anteriores, explicando si se ha considerado algún supuesto semántico adicional. Si alguna especificación del enunciado no ha podido reflejarse en el esquema, hacerlo constar.

DISCUSIÓN DEL ENUNCIADO

PARTE 1

La empresa desarrolla proyectos de los que se almacena su código, nombre, cliente para el que se desarrolla el proyecto, una breve descripción, presupuesto, número de horas totales estimadas, fecha de inicio y fecha de fin. Cada proyecto se compone de una serie de fases identificadas por un número en secuencia en cada proyecto. Cada fase se caracteriza, además, por su nombre, fecha de comienzo, fecha de fin y estado en que se encuentra (en curso o finalizada).

Los empleados de la empresa (código, DNI, nombre, dirección, titulación y años de experiencia) están asignados a los proyectos que desarrolla la empresa. Interesa almacenar los empleados que son jefes de proyecto junto con su dedicación total en horas prevista a cada proyecto así como el coste de su participación en euros, teniendo en cuenta que cada proyecto lo lidera un único jefe de proyecto. En cuanto a los informáticos que participan en los proyectos, se quiere conocer los que son analistas y los que son programadores, así como el número de horas totales previstas dedicadas en cada proyecto y el coste en euros que supone cada dedicación. De los programadores se almacenarán también los lenguajes en los que son expertos.

En el primer párrafo se pueden identificar dos entidades *PROYECTO* y *FASE*. Cada fase se corresponde con un proyecto, por lo que su existencia depende de éste. Además, como cada fase se identifica por un número de secuencia dentro del proyecto al que pertenece, se trata de una dependencia en identificación.

Con relación a los atributos, en *PROYECTO*, se debería comprobar que la fecha de fin de un proyecto es superior a la fecha de inicio del mismo. De igual manera, las horas estimadas por proyecto no deben exceder el número total de horas que resulta de restar las fechas de fin e inicio. En cuanto a *FASE*, además de comprobar que la fecha de fin sea posterior a la fecha de inicio, hay que tener en cuenta que los intervalos de fechas estén incluidos en el intervalo compuesto por la fecha de inicio y la fecha de fin del proyecto y que la suma de todas las fechas correspondientes al conjunto de fases de un determinado proyecto, no

exceda la fecha de finalización del mismo. Con referencia al estado de cada fase, éste será un atributo que tomará valores en el dominio {en curso, finalizada}. Por último, se podría suponer que el nombre de los proyectos es un atributo AIA, pues habitualmente estos son únicos.

En el segundo párrafo se describen los tipos de empleados que participan en los proyectos. Estos pueden ser jefes de proyecto o informáticos y además los informáticos pueden ser analistas o programadores con lo que se tiene una jerarquía a dos niveles. Se puede considerar que en la empresa hay más tipos de empleados aunque no sean relevantes en el dominio que se está modelando, por lo que la jerarquía *EMPLEADO* será parcial pero sin solapamiento, pues un jefe de proyecto no es un informático ni viceversa. De esta manera, el atributo discriminante *Tipo* tomará valores en el conjunto {jefe proyecto, informático, null}. En cambio, como los informáticos solo son de dos tipos, analistas o programadores, la jerarquía *INFORMÁTICO* sí es total y también sin solapamiento, con lo que su atributo discriminante *Tipo* tomará valores en el conjunto {analista, programador}. Como los empleados tienen un código de empleado y también se desea recoger su DNI, ser puede considerar el primero como atributo AIP y el segundo como AIA.

Interesa almacenar el liderazgo de los proyectos, así como la dedicación en horas y el coste total de los jefes de proyecto, por tanto se debe crear una interrelación **Dirigido** entre *PROYECTO* y *JEFE PROYECTO*. Como cada proyecto es liderado por un único jefe de proyecto, pero un jefe puede liderar varios proyectos, la interrelación será 1:N. Además deberá llevar dos atributos *Num_horas* y *Coste total*, debiendo comprobarse adicionalmente que el número de horas no excede en ningún caso del cómputo de las horas resultante de restar las fechas de fin e inicio de cada proyecto. Al igual que con los jefes de proyecto, se desea almacenar la relación existente entre los proyectos y los empleados informáticos, almacenando también las horas totales de dedicación y el coste. Todo ello se refleja en la interrelación N:M **Trabaja** entre *INFORMÁTICO* y *PROYECTO*, debiendo hacerse las comprobaciones pertinentes en los atributos. Por último resaltar que de los programadores se desea recoger los lenguajes en los que son expertos, esto se debe recoger con un atributo multivalorado “Lenguajes”, para dar la posibilidad de que haya más de uno. Será necesario comprobar de forma adicional que el presupuesto de un determinado proyecto nunca puede ser inferior a la suma de todos los costes derivados de sus empleados.

Se podría haber pensado en la posibilidad de, en lugar de definir las

interrelaciones **Trabaja** y **Dirigido**, crear una única interrelación directamente entre la entidad *EMPLEADO* (el supertipo de la jerarquía) y la entidad *PROYECTO*. Esta decisión sería válida, pero representa menos semántica en el diagrama E/R. En el caso de tener una única interrelación, sería necesario comprobar adicionalmente a lo comentado en el apartado anterior que para cada proyecto existe al menos uno y solo un jefe de proyecto trabajando en el mismo.

También sería necesario comprobar que al menos un informático también está trabajando en cada proyecto.

La Figura 1.33 muestra todas las decisiones de diseño tomadas previamente.

PARTE 2

En cada fase de un proyecto se generan una serie de productos (software, informes técnicos y prototipos) sobre los que se quiere guardar información. Cada producto previsto para una fase tiene un código, un nombre, una descripción, si está finalizado o no y tiene como responsable un analista. Además, la obtención de un producto puede abarcar varias fases de un proyecto y se quiere guardar los empleados informáticos involucrados en cada producto cuantificando el número de horas de trabajo dedicadas a cada producto teniendo en cuenta cada fase de un proyecto en la que se desarrolla. Para el software se almacena, además, su tipo (diagrama, programa, etc.) y para los prototipos su versión y su ubicación.

Como en cada fase se generan una serie de productos, se tiene una nueva entidad *PRODUCTO* y una interrelación **Genera** con la entidad *FASE*. Además, como un mismo producto puede generarse en varias fases, será N:M. Además se desea recoger qué analista es responsable de qué producto y esto se puede recoger mediante una interrelación **Es responsable**.

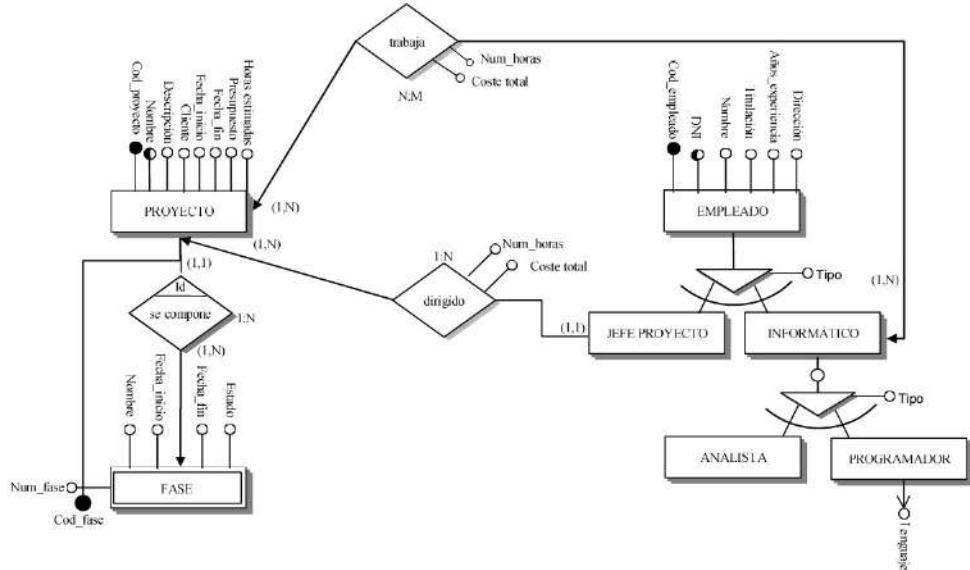


Figura 1.33. Diagrama E/R correspondiente a la Parte 1

Los productos son de tres tipos, pero solo los prototipos y el software tienen características propias, por lo que se puede incluir una jerarquía parcial y con solapamiento (un prototipo, por la naturaleza de los proyectos, incorpora software) con solo dos subtipos *SOFTWARE* y *PROTOTIPO*. Los valores del atributo *Estado* de la entidad deben pertenecer al dominio {si, no} (indicando si el producto se encuentra o no finalizado) y el atributo discriminante *Tipo* tomará valores en el conjunto {software, prototipo, informe técnico}.

Se desea recoger información acerca de los empleados informáticos involucrados en productos teniendo en cuenta la fase del proyecto en la que se desarrolla. Un informático en una fase estará involucrado como mínimo en un producto y como máximo en N. Un producto generado en una fase contará como mínimo con un informático y como máximo con varios, para su desarrollo. Por último, un informático involucrado en la creación de un producto, puede desarrollarlo como mínimo en una fase y como máximo en varias. Por todo ello se está ante una interrelación ternaria **Involucrado** N:M:P, con un atributo propio *Num_horas* que indica el número de horas de cada empleado aplicadas en la elaboración del producto. Será necesario comprobar que no exceda del número total de horas asignadas a la fase en concreto que se esté tratando.

La Figura 1.34 muestra los supuestos semánticos tratados en este apartado.

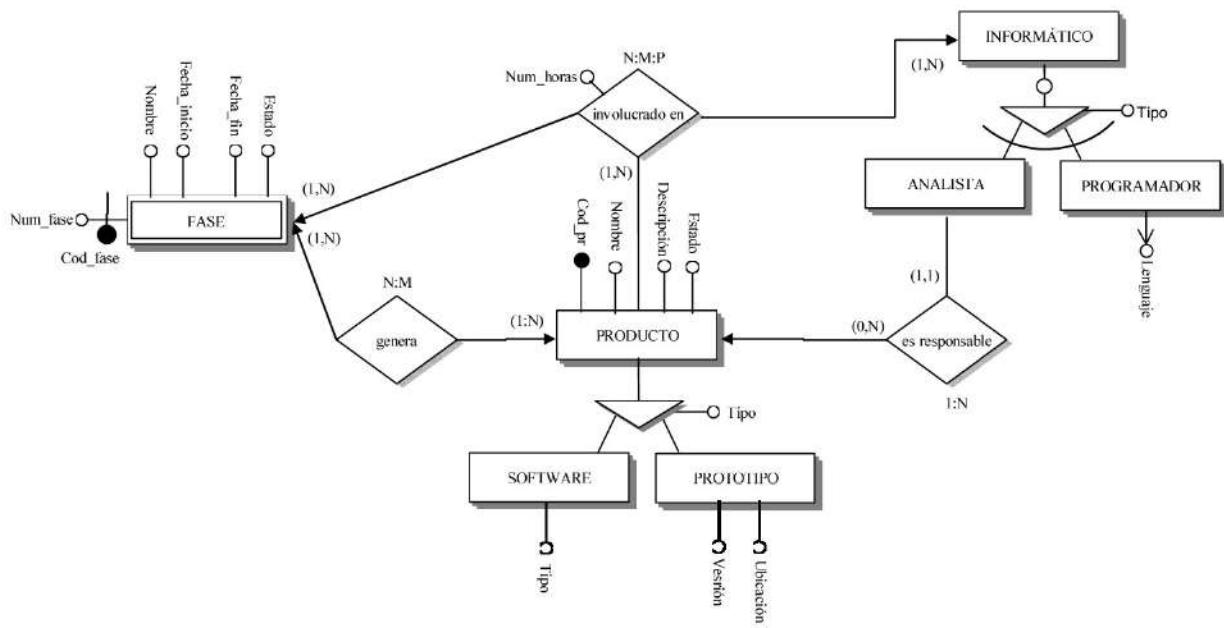


Figura 1.34. Diagrama E/R correspondiente a la Parte 2

Hay que observar que la semántica almacenada en **Involucrado** es distinta a la que se recoge en **Trabaja**, puesto que un empleado informático puede no estar en todos los productos de un determinado proyecto en el que trabaja. Además, el diseño visto previamente impide recoger que todo empleado que está involucrado en el desarrollo de algún producto en alguna fase de un proyecto, éste participa también en la interrelación **Trabaja** del mismo proyecto. De igual forma, será necesario comprobar que un analista ha de aparecer en la interrelación **Involucrado** para todos los productos en los que aparece como responsable del producto.

PARTE 3

Por otro lado, a cada fase de un proyecto le corresponde una serie de recursos de los que se quiere conocer su código identificador, nombre, descripción, tipo (Hw o Sw) así como el periodo de tiempo que se utilizan en cada fase.

Cada empleado, en cada proyecto en el que trabaja, puede originar una serie de gastos (dietas, viajes, alojamiento, etc.) que se cargan a los proyectos. Cada gasto se caracteriza por un código único, una descripción, una fecha, un importe y el tipo de gasto.

Por último, interesa conocer qué proyectos están relacionados con uno

dado, es decir, cuáles son los proyectos que incluyen aspectos similares según distintas palabras clave (gestión de personal, gestión de stocks, etc.), con el fin de facilitar su desarrollo reutilizando algún componente.

De la lectura del primer párrafo se deduce una nueva entidad *RECURSO* cuyo atributo *Tipo* pertenecerá al dominio {hw, sw} y que se relacionará con *FASE* mediante la interrelación **Se asigna**. Esta interrelación tendrá como atributos propios la fecha de inicio y la fecha de fin de utilización del recurso para poder recoger el periodo de tiempo. La interrelación será de tipo N:M pues un mismo recurso puede ser utilizado en distintas fechas en varias fases. Suponemos que un recurso no puede ser utilizado en la misma fase del proyecto en distintos periodos de tiempo, ya que en otro caso los atributos fecha de inicio y fecha de fin se tendrían que definir como atributos multivaluados.

Una vez más habrá que comprobar que la fecha de fin sea posterior a la fecha de inicio en esta interrelación, que el intervalo está contenido en el compuesto por las fechas de inicio y de fin de la fase correspondiente y que los distintos periodos de utilización de un recurso no pueden solaparse.

En el segundo párrafo se introduce una nueva entidad *GASTO* relacionada con las entidades *PROYECTO* y *EMPLEADO* mediante las interrelaciones 1:N **Origina** y **Realizado**, respectivamente. De nuevo será necesario comprobar que la fecha en la que se origina el gasto, es posterior a la del correspondiente proyecto pero anterior a su fecha de finalización. Además, con el diseño propuesto no se puede asegurar que el empleado que realiza el gasto esté trabajando en el proyecto al que se carga dicho gasto. También se debería comprobar que el importe de un gasto asociado a un determinado proyecto sea inferior al presupuesto de éste y que la suma de todos los importes de todos los gastos asociados a un proyecto también sea inferior y, en otro caso, rechazar el gasto.

Por último interesa recoger qué proyectos están relacionados con uno dado. Esto significa la introducción de una interrelación reflexiva **Relacionado con** entre proyectos, de manera que cada proyecto puede estar relacionado con cero o con varios proyectos. Para almacenar palabras clave que faciliten la búsqueda por similitud entre proyectos relacionados, se debe incluir un atributo multivaluado *Keywords* en la nueva interrelación. Será necesario comprobar de forma adicional que un proyecto no está **Relacionado** consigo mismo.

La Figura 1.35 ilustra los supuestos semánticos tomados en consideración en esta parte.

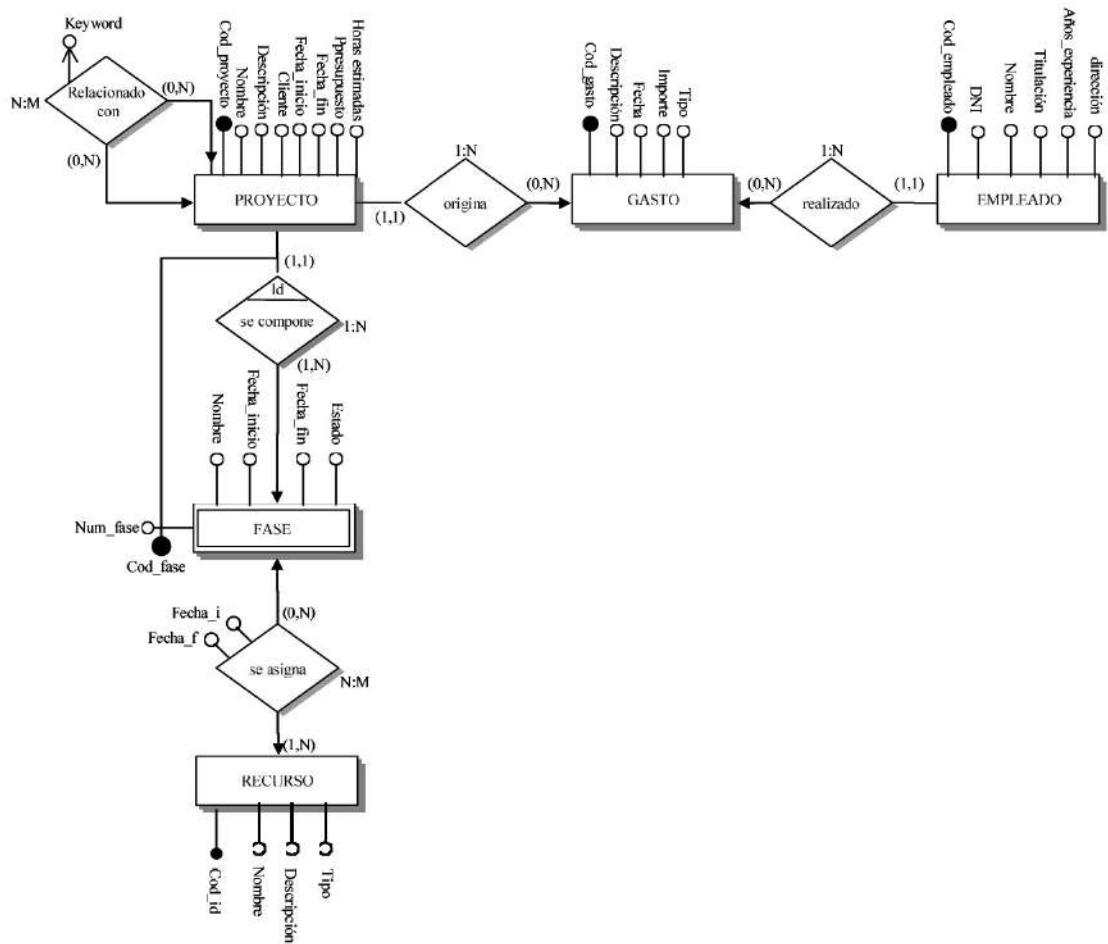


Figura 1.35. Diagrama E/R correspondiente a la Parte 3

Solución propuesta

DIAGRAMA E/R

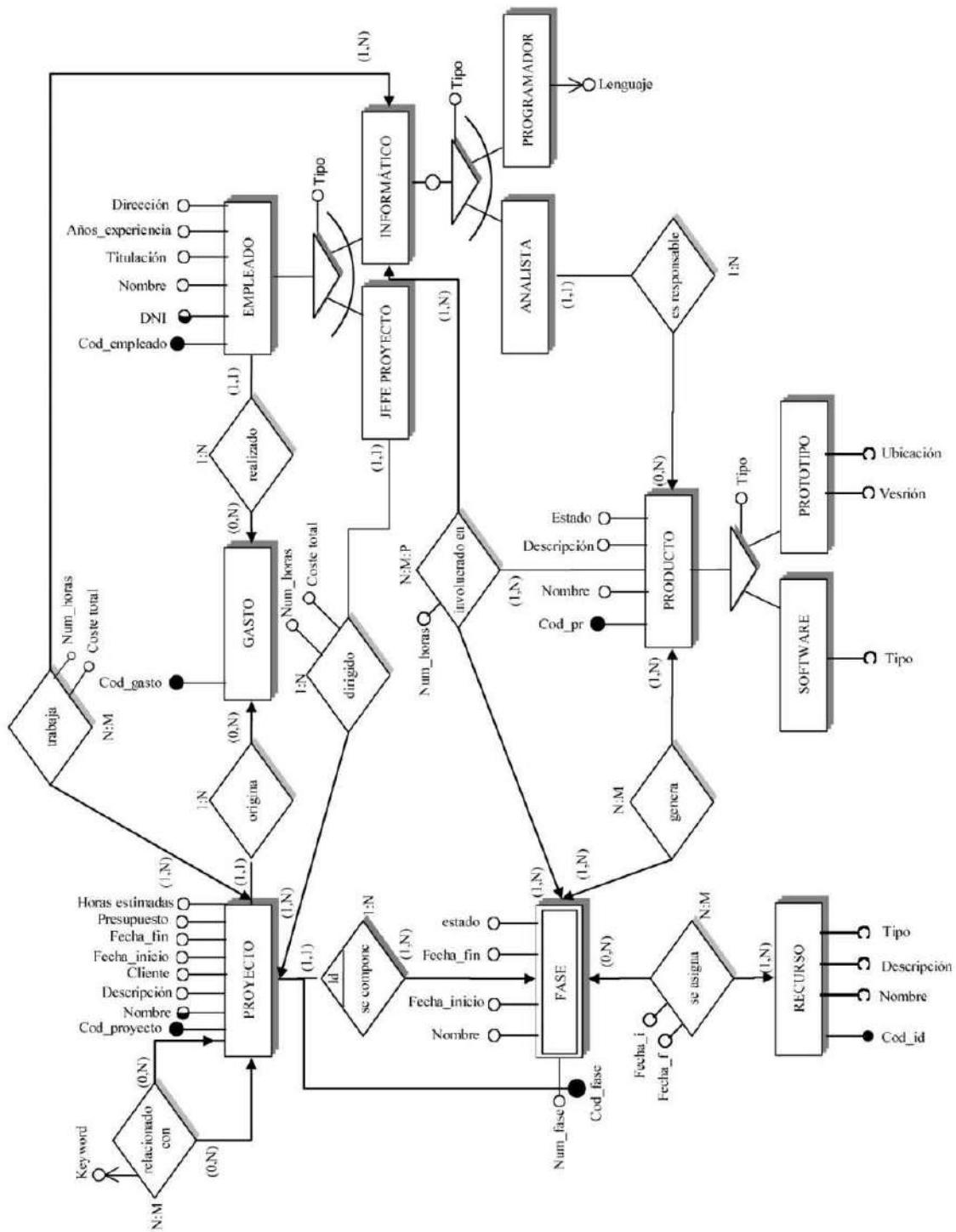


Figura 1.36. Diagrama E/R propuesto para la gestión de proyectos informáticos

SUPUESTOS ASOCIADOS AL DIAGRAMA E/R

Dominios

- SS1. FASE.Estado = {en curso, finalizada}.
- SS2. PRODUCTO.Estado = {si, no}.
- SS3. PRODUCTO.Tipo = {software, prototipo, informe técnico}.
- SS4. EMPLEADO.Tipo = {jefe proyecto, informático, null}.
- SS5. INFORMÁTICO.Tipo = {analista, programador}.
- SS6. RECURSO.Tipo = {hw, sw}.

Otros

- SS7. PROYECTO.Fecha_fin >= PROYECTO.Fecha_inicio.
- SS8. PROYECTO.Horas estimadas <= PROYECTO.Fecha_fin - PROYECTO.Fecha_inicio.
- SS9. FASE.Fecha_fin >= FASE.Fecha_inicio.
- SS10. Comprobar que los intervalos de fechas en la entidad FASE estén incluidos en el intervalo compuesto por la fecha de inicio y la fecha de fin del proyecto.
- SS11. Comprobar que la suma de todas las fechas correspondientes al conjunto de fases de un determinado proyecto, no exceda la fecha de finalización del mismo.
- SS12. **Dirigido.Num_horas** = PROYECTO.Fecha_fin - PROYECTO.Fecha_inicio. SS13. **Trabaja.Num_horas** = PROYECTO.Fecha_fin - PROYECTO.Fecha_inicio. SS14. **Involucrado.Num_horas** = < FASE.Fecha_fin - FASE.Fecha_inicio.
- SS15. **Se_asigna.Fecha_f** >= **Se_asigna.Fecha_i**.
- SS16. Comprobar que el intervalo de tiempo de la interrelación **Se_asigna** está contenido en el compuesto por las fechas de inicio y de fin de la fase correspondiente.
- SS17. Los periodos de tiempo de la interrelación **Se_asigna** no pueden solaparse para el mismo recurso.

SS18. Un informático ha de aparecer en la interrelación **Trabaja** para todos los proyectos en los que aparece como Involucrado en alguna de sus fases.

SS19. Un analista ha de aparecer en la interrelación **Involucrado** para todos los productos en los que aparece como responsable del producto.

SS20. PROYECTO.Fecha_inicio =< GASTO.Fecha =<
PROYECTO.Fecha_fin.

SS21. Controlar que el empleado que realiza el gasto esté trabajando en el proyecto al que se carga dicho gasto.

SS22. El presupuesto de un proyecto siempre ha de ser mayor o igual a la suma de todos los costes de sus empleados más los gastos que generan.

SS23. Un proyecto no está **Relacionado** consigo mismo.

PROBLEMA 1.7: MEDIO AMBIENTE

Dado el esquema E/R de la Figura 1.37 correspondiente al diseño de una BD de medio ambiente para recoger las características y ubicación de los ejemplares de árboles de las calles de una ciudad.

Se pide:

Extender el diagrama E/R de la Figura 1.37 para recoger los siguientes supuestos semánticos, indicando aquellos supuestos que no hayan podido reflejarse en la solución propuesta.

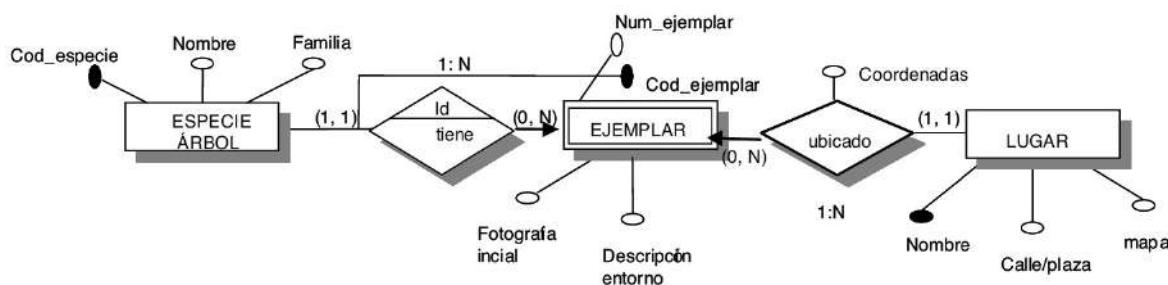


Figura 1.37. Diagrama E/R inicial sobre medio ambiente

A. Interesa guardar la información no solo de los ejemplares existentes en la actualidad si no también de los desaparecidos; de estos últimos se almacenará la fecha de desaparición de ejemplar. Esto permitirá almacenar en otra BD (que no es objeto de este ejercicio) un histórico de los ejemplares desaparecidos.

B. Además, el objetivo de la BD es que toda la actividad dedicada al cuidado y mejora que se realiza sobre todos los ejemplares de los árboles quede registrada. Para ello, se almacenará en la BD una serie de actividades a las que puede someterse cualquier ejemplar de un árbol. Por un lado se guardan todas las podas de cada ejemplar, interesando el tipo de poda, comentarios, fotografía y si se trata de una poda o de una tala; esta información permite comparar de una poda a otra si el trabajo ha sido correcto y permitió mejorar el crecimiento. También se registran las mediciones tomadas a cada ejemplar (altura, tamaño de la copa y la edad relativa), las limpiezas, almacenando el tipo de limpieza (si es del tronco o de la base) y una descripción y, por último, la eliminación de las plagas de la que se guarda una descripción de los daños, la especie atacante (lombrices, pulgones, etc.) y el tratamiento que se aplica para combatirla.

Discusión del Enunciado

PARTE 1

A. Interesa guardar la información no solo de los ejemplares existentes en la actualidad si no también de los desaparecidos; de estos últimos se almacenará la fecha de desaparición de ejemplar. Esto permitirá almacenar en otra BD (que no es objeto de este ejercicio) un histórico de los ejemplares desaparecidos.

Con el fin de representar los distintos tipos de ejemplares de la BD se ha representado en el diagrama E/R de la Figura 1.38 una jerarquía total exclusiva con dos subtipos, *DESAPARECIDO* y *ACTUAL*. Solo el subtipo *DESAPARECIDO* tiene atributos propios (*Fecha_desaparición*).

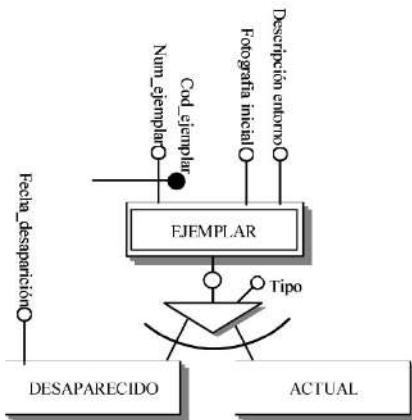


Figura 1.38. Diagrama E/R correspondiente a Parte 1

Puesto que además el subtipo *ACTUAL* no almacena información propia, otra solución válida es la que se muestra en la Figura 1.39 consistente en tener el atributo opcional *Fecha_desaparición* en la entidad *EJEMPLAR*.

PARTE 2

B. Además, el objetivo de la BD es que toda la actividad dedicada al cuidado y mejora que se realiza sobre todos los ejemplares de los árboles quede registrada. Para ello, se almacenará en la BD una serie de actividades a las que puede someterse cualquier ejemplar de un árbol. Por un lado se guardan todas las podas de cada ejemplar, interesando el tipo de poda, comentarios, fotografía y si se trata de una poda o de una tala; esta

información permite comparar de una poda a otra si el trabajo ha sido correcto y permitió mejorar el crecimiento. También se registran las mediciones tomadas a cada ejemplar (altura, tamaño de la copa y la edad relativa), las limpiezas, almacenando el tipo de limpieza (si es del tronco o de la base) y una descripción y, por último, la eliminación de las plagas de la que se guarda una descripción de los daños, la especie atacante (lombrices, pulgones, etc.) y el tratamiento que se aplica para combatirla.

Cada una de estas actividades se realiza en una determinada fecha de tal manera que en una fecha solo puede llevarse a cabo una actividad sobre un ejemplar, almacenándose la hora de inicio y la hora de fin y los operarios de parques y jardines del ayuntamiento (código de empleado, nombre y teléfono de contacto) que las han llevado a cabo. Como máximo participan dos operarios en cualquiera de las actividades que pueden llevarse a cabo sobre un determinado ejemplar de árbol.

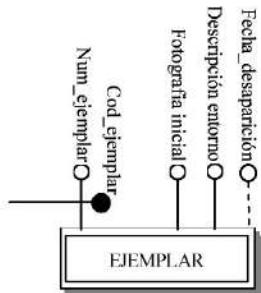


Figura 1.39. Diagrama E/R correspondiente a Parte 2

La Figura 1.40 muestra la solución adoptada en la que se ha creado una jerarquía total y exclusiva con *ACTIVIDAD* como subtipo y con las distintas actividades como subtipos, a saber, *PODA*, *MEDICIÓN*, *LIMPIEZA* y *PLAGA* cada una con atributos propios. El identificador de *ACTIVIDAD* (*Cod_actividad*) que depende en identificación de *EJEMPLAR* es la combinación del *Cod_ejemplar* y la *Fecha*. La interrelación N:M **Participa** representa la asignación de los operarios a las actividades, de tal manera que una actividad tiene asignados como máximo dos operarios.

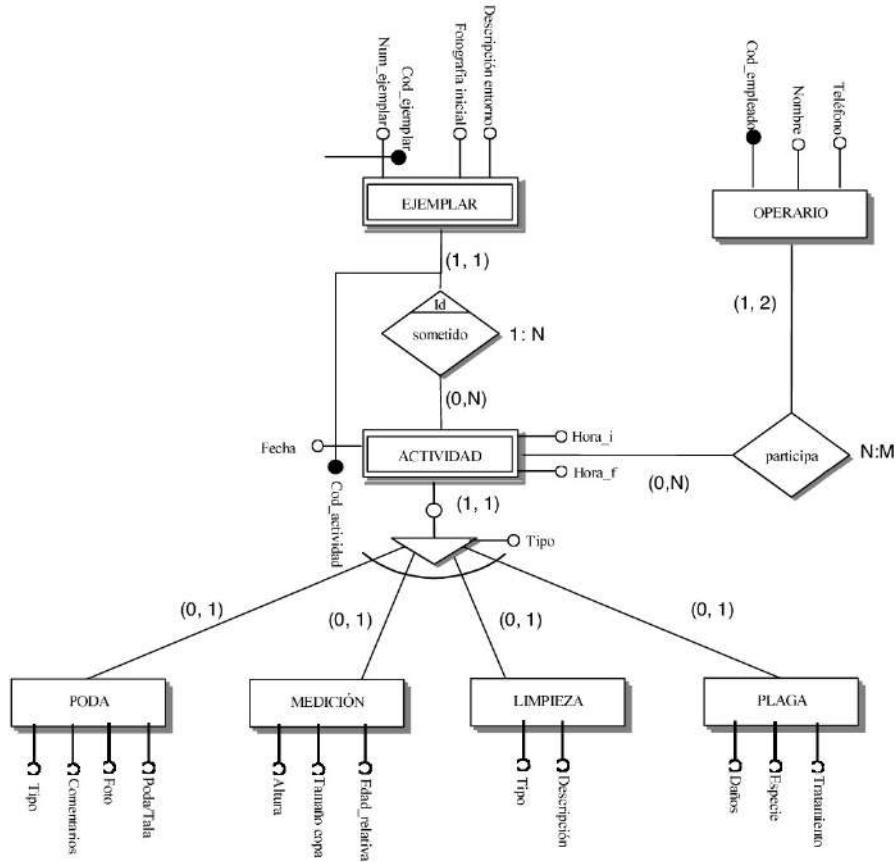


Figura 1.40. Diagrama E/R correspondiente a Parte 3

En los diagramas E/R de esta parte no se han podido reflejar los siguientes supuestos semánticos:

- Un operario no puede trabajar simultáneamente en varias actividades.
- No es posible efectuar ninguna actividad sobre un ejemplar desaparecido, por lo que tendrá que compararse la fecha de desaparición del ejemplar con las fechas de la actividad.

Además, en el diagrama E/R inicial no se pudo comprobar el siguiente supuesto semántico:

- No puede haber dos ejemplares actuales en las mismas coordenadas.

SOLUCIÓN PROPUESTA

A continuación se muestra el esquema E/R propuesto para este problema, compuesto por el diagrama E/R y los supuestos semánticos que no se han podido añadir en el diagrama.

SUPUESTOS ASOCIADOS AL DIAGRAMA E/R

- SS1. Un operario no puede trabajar simultáneamente en varias actividades.
- SS2. No es posible efectuar ninguna actividad sobre un ejemplar desaparecido.
- SS3. No puede haber dos ejemplares actuales en las mismas coordenadas.
- SS4. El atributo *Tipo* de la entidad *LIMPIEZA* toma valores en el dominio {tronco, base}.
- SS5. El atributo *Poda/Tala* de la entidad *PODA* toma valores en el dominio {poda, tala}.
- SS6. El atributo *Tipo* de la generalización *ACTIVIDAD* toma valores en el dominio {poda, medición, limpieza, plaga}.
- SS7. El atributo *Tipo* de la jerarquía de *EJEMPLAR* toma valores en el dominio {desaparecido, actual}.

DIAGRAMA E/R

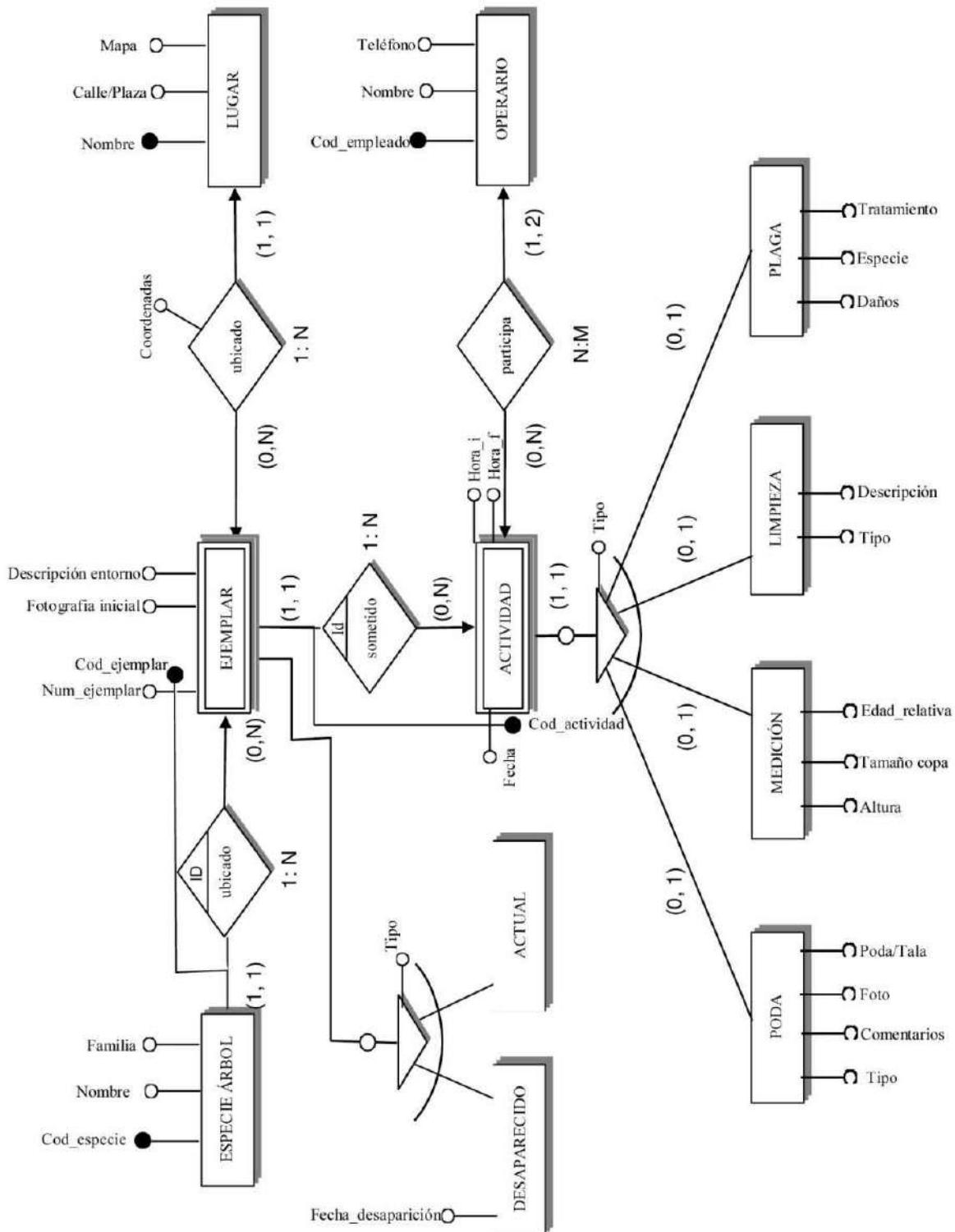


Figura 1.41. Diagrama E/R propuesto para la BD sobre medio ambiente

PROBLEMA 1.8: CLÍNICA OFTALMOLÓGICA

ENUNCIADO

El esquema E/R de la Figura 1.42 corresponde a una parte del diseño de una BD para almacenar la información relativa a una cadena de clínicas oftalmológicas especializadas en el diagnóstico y tratamiento de problemas y trastornos oculares. En este esquema se muestra la información almacenada sobre los pacientes que acuden a la clínica para someterse a un tratamiento (por ejemplo, de cirugía láser para corrección de miopía) y el médico asignado para aplicarle ese tratamiento.

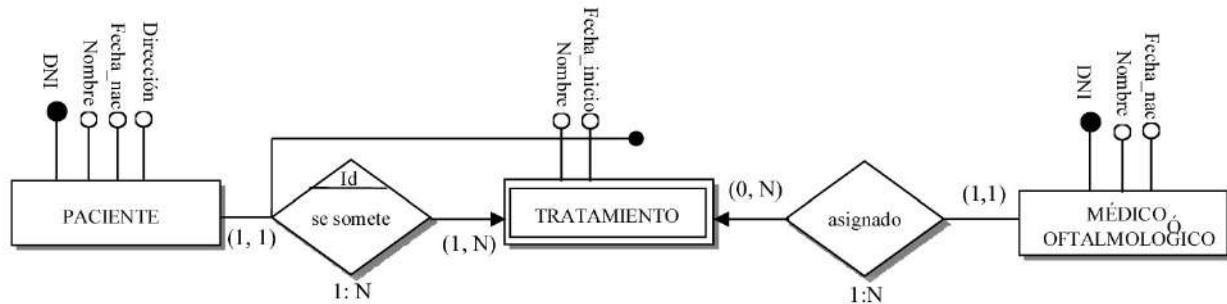


Figura 1.42. Diagrama E/R clínicas oftalmológicas

Se pide extender el esquema E/R de la Figura 1.42 para recoger los siguientes supuestos semánticos (indicando aquellos supuestos que no hayan podido reflejarse en la solución propuesta):

A. Se desea almacenar información sobre las distintas clínicas existentes (Calle, Número, Código postal, Ciudad, Teléfono) así como a qué clínica está asignado cada médico oftalmólogo y en qué períodos de tiempo, manteniendo un histórico. Un médico puede cambiar de clínica a lo largo del tiempo.

B. Además, cada tratamiento incluye una serie de pruebas (por ejemplo, revisión, graduación de la vista, análisis de la córnea, etc.) de tal manera que una prueba se identifica con un nombre dentro de cada tratamiento y se caracteriza por una fecha y hora de realización, por un tipo (obligatoria o solicitada por el paciente), y por una descripción de los resultados. Así mismo, cada prueba la lleva a cabo un médico (que no tiene por qué coincidir con el asignado

originariamente al inicio del tratamiento).

DISCUSIÓN DEL ENUNCIADO

APARTADO A

Se desea almacenar información sobre las distintas clínicas existentes (Calle, Número, Código postal, Ciudad, Teléfono) así como a qué clínica está asignado cada médico oftalmólogo y en qué períodos de tiempo, manteniendo un histórico. Un médico puede cambiar de clínica a lo largo del tiempo.

Se detecta una entidad nueva *CLÍNICA* cuyo atributo identificador principal debe estar compuesto por los atributos *Calle*, *Número* y *Código postal*, pues no hay otro mecanismo para distinguir entre dos clínicas. Como los médicos pueden pertenecer a una misma clínica en distintos períodos de tiempo, es necesario incluir un atributo multivaluado *Periodo*, compuesto por la fecha de inicio y la fecha de fin de pertenencia a una Clínica. Es más, la fecha de fin puede considerarse como un atributo opcional, pues puede que un médico determinado, en el momento actual, no haya cambiado de clínica, por lo que no exista valor de dicho atributo.

Un médico puede pertenecer a varias clínicas a lo largo del tiempo (como mínimo a una) y una clínica puede contar con varios médicos (como mínimo uno), por tanto la correspondencia es N:M para la interrelación **Pertenecce**.

Adicionalmente se debería comprobar que la fecha de inicio de permanencia en una clínica es inferior a la fecha de fin y que no existen solapamientos entre períodos de tiempo, es decir, que un médico no pueda trabajar en varias clínicas al mismo tiempo, ocurriendo que la fecha de inicio en la interrelación **Pertenecce** implica una única fecha de finalización y, al mismo tiempo, la fecha de finalización implica una única fecha de inicio. Como cada médico tiene que estar asignado como mínimo a una clínica, habría que comprobar también que no existen lapsus de tiempo en **Pertenecce** para ningún médico.

Por último habría que comprobar que, lógicamente, la fecha de nacimiento de cada médico es anterior a la fecha de inicio de pertenencia a una clínica.

La Figura 1.43 muestra el subesquema resultante.

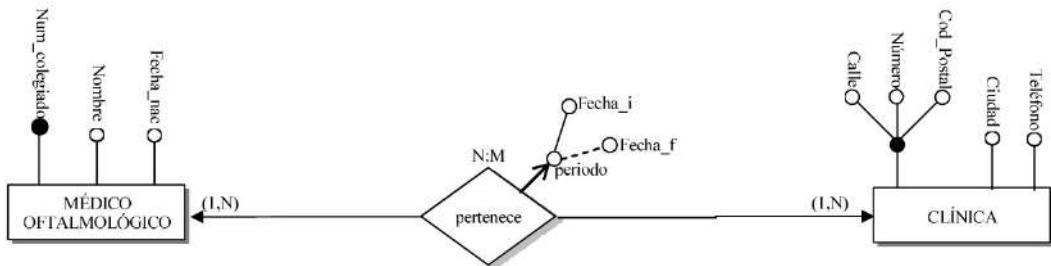


Figura 1.43. Diagrama E/R correspondiente al Apartado A

APARTADO B

“Cada tratamiento incluye una serie de pruebas (por ejemplo, revisión, graduación de la vista, análisis de la córnea, etc.) de tal manera que una prueba se identifica con un nombre dentro de cada tratamiento y se caracteriza por una fecha y hora de realización, por un tipo (obligatoria o solicitada por el paciente), y por una descripción de los resultados. Así mismo, cada prueba la lleva a cabo un médico (que no tiene por qué coincidir con el asignado originalmente al inicio del tratamiento)”.

Cada prueba se realiza en función del tratamiento asignado, por tanto la entidad PRUEBA es débil con respecto a TRATAMIENTO. Es más, toda prueba se identifica por un nombre dentro de cada tratamiento, por lo que la interrelación **Incluye** entre ambas entidades implica una dependencia en identificación.

En el enunciado se dice que los médicos son los que asignan los tratamientos, sin embargo las pruebas correspondientes a cada tratamiento pueden ser realizadas por médicos que no han asignado el tratamiento, por tanto no basta con la interrelación **Asignado** para recoger este supuesto y se debe incluir una nueva **Realizada** entre MEDICO y PRUEBA. El ciclo **Asignado, Incluye, Realizada** no es susceptible de ser redundante porque el médico que realiza una prueba no tiene por qué coincidir con el asignado al tratamiento.

Todos estos supuestos aparecen reflejados en la Figura 1.44.

De entre los supuestos semánticos que no se pueden recoger se incluyen que no es posible controlar que un médico tenga asignadas dos pruebas simultáneamente, y tampoco es posible reflejar que no haya dos pruebas simultáneas en un mismo tratamiento ni que la fecha de una prueba sea mayor que la del tratamiento. Por último se debería controlar la validez de las fechas entre la fecha de nacimiento de un médico y la fecha en la que realiza las

pruebas y que el atributo *Tipo* de la entidad *PRUEBA* toma valores en el dominio {Obligatoria, Solicitada}.

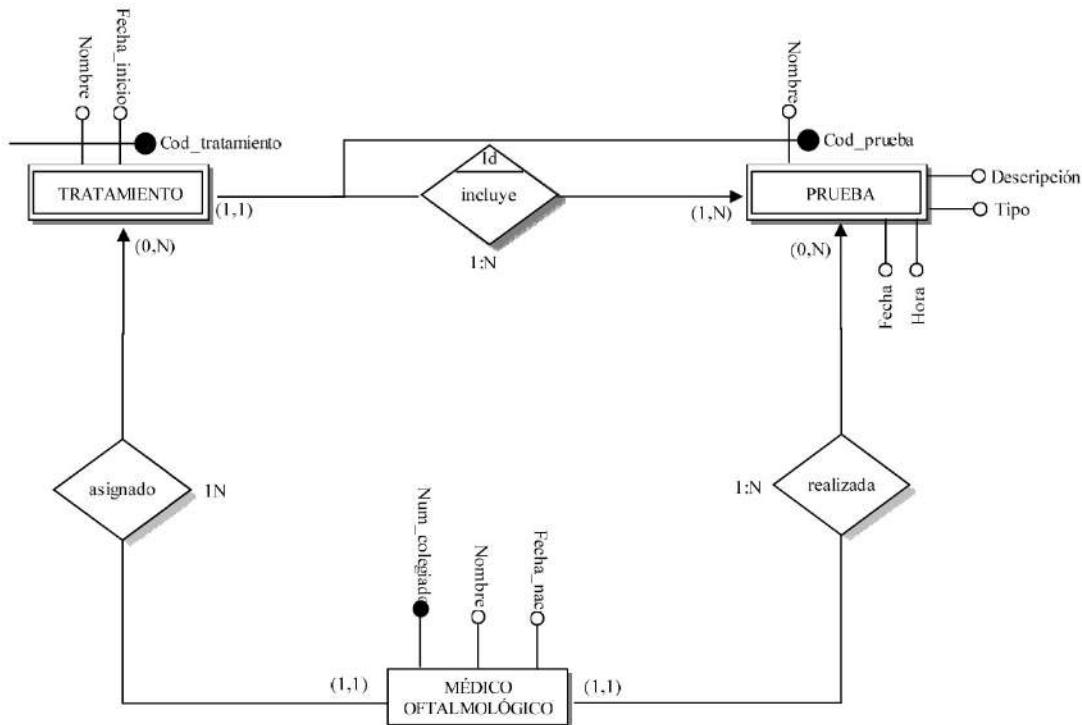


Figura 1.44. Diagrama E/R correspondiente al Apartado B

SOLUCIÓN PROPUESTA

SUPUESTOS ASOCIADOS AL DIAGRAMA E/R

Dominios

SS1. PRUEBA. Tipo = {Obligatoria, Solicitudada}.

Otros

SS2. Comprobar que la fecha de inicio de permanencia en una clínica es inferior a la fecha de fin

SS3. Comprobar no existen solapamientos entre periodos de tiempo en la interrelación Pertenece, es decir, que un médico no pueda trabajar en varias clínicas al mismo tiempo, ocurriendo que la fecha de inicio en la interrelación **Pertenece** implica una única fecha de finalización y, al mismo tiempo, la fecha de finalización implica una única fecha de inicio.

SS4. Comprobar también que no existen lapsus de tiempo en **Pertenece** para ningún médico.

SS5. La fecha de nacimiento de un MÉDICO OFTALMOLÓGICO ha de ser anterior a la fecha de inicio de pertenencia del médico a una clínica.

SS6. Controlar que un médico no tenga asignadas dos pruebas simultáneamente.

SS7. Controlar que no haya dos pruebas simultáneas en un mismo tratamiento.

SS8. La fecha de una determinada prueba ha de ser posterior o igual en el tiempo a la fecha de inicio del tratamiento.

SS9. La fecha de nacimiento de un médico ha de ser anterior a la fecha de las pruebas realizadas por él.

SS10. La fecha de nacimiento del paciente ha de ser anterior o igual a la fecha de inicio del tratamiento.

DIAGRAMA E/R

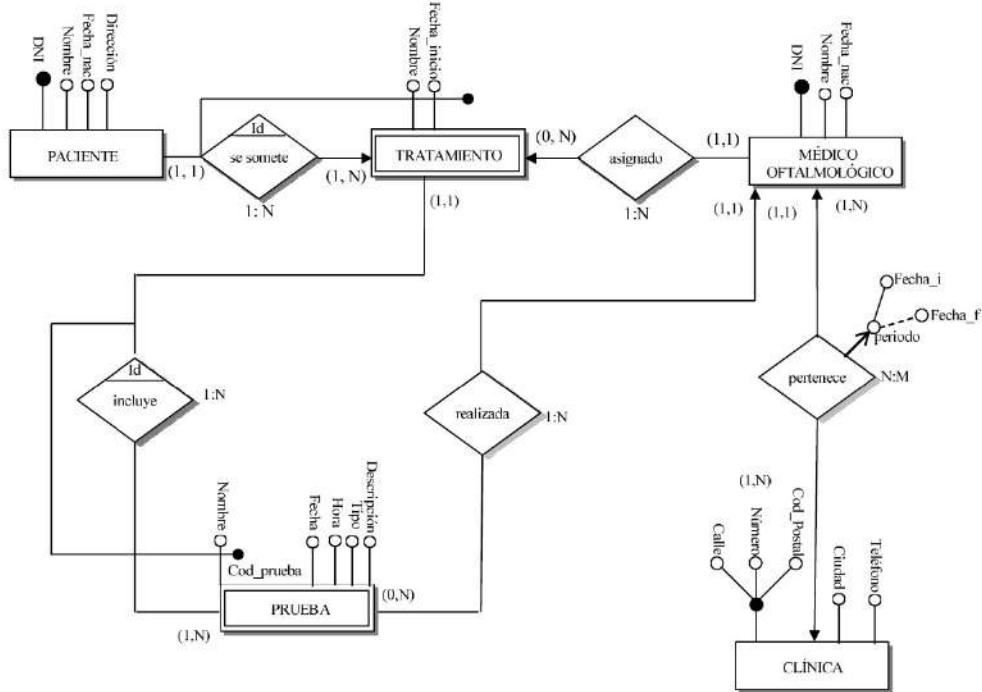


Figura 1.45. Diagrama E/R propuesto para la gestión de clínicas oftalmológicas

PROBLEMA 1.9: LONJA DE PESCADO

Se quiere desarrollar una base de datos para llevar la gestión de la lonja de pescado de un pueblo costero. Los barcos llevan la pesca de cada día a la lonja y allí se subasta a los compradores que generalmente son pescaderías de la zona. Los supuestos semánticos que se deben contemplar en la base de datos son los siguientes:

Una vez llega la pesca de cada día en los barcos, ésta se prepara en los distintos lotes que se subastarán. Cada lote se identifica por un código de lote que se le asigna en la lonja antes de la subasta y consta de un número de cajas de una determinada especie (por ejemplo, pulpo, merluza, gambas, etc.) así como el número de kilos total y la fecha de recepción. Además, también interesa almacenar el precio por kilo de salida y el precio total de salida del lote.

De cada especie se guardará un código, un nombre y un tipo (por ejemplo, marisco, pescado azul, etc.). Se almacenará también información sobre los barcos (matrícula, nombre, clase, nombre del capitán y armador) que entregan la pesca en la lonja para saber qué barco capturó cada lote.

Estos barcos pueden faenar en distintos caladeros en los que capturan las especies que componen los lotes. De cada caladero nos interesa conocer un nombre (que suponemos es único), extensión y ubicación. En la lonja se guarda información relativa a qué barcos y en qué caladeros se han capturado las especies (número de kilos de cada especie y periodo de tiempo de faena representado por una fecha de inicio y otra de fin).

Una vez empezada la subasta, los distintos compradores (código de comprador, nombre, dirección, CIF y cuota anual de pago a la lonja) pujan por los lotes en los que están interesados. Cada lote se asigna al comprador que realiza mejor puja. De cada adquisición de lote se almacena el precio de compra por kilo y el precio total de adjudicación del lote.

En la BD se almacenará información de los pagos que realiza la lonja a los barcos que entregan la pesca diaria y de los pagos que efectúan los compradores por la adquisición de los lotes. En cuanto a los compradores, existen compradores que tienen crédito y realizan los pagos al final de cada mes; de estos compradores se guarda un numero de cuenta bancaria, el último importe acumulado hasta el momento y la fecha de vencimiento del pago (suponemos

que no se guarda histórico de todas las mensualidades pues solo nos interesa la mensualidad en curso). Por otro lado, existen los compradores que realizan los pagos al contado sobre los que no se necesita guardar información adicional. Un comprador no puede ser de ambos tipos a la vez.

Así, la lonja generará una factura por uno o varios lotes que ha adquirido un comprador. De todas las facturas se guarda un número de factura, una fecha de emisión y un importe total. Además, en las facturas emitidas a los compradores se incluyen los lotes que contiene y el comprador que debe abonarla. En las facturas emitidas por los barcos, la lonja almacena además de los datos mencionados de la factura, el CIF del barco y los códigos de lote facturados. En el caso de los compradores sin crédito interesa saber el estado de sus facturas (pendiente o pagado).

Se pide:

Realizar el esquema E/R extendido correspondiente a los supuestos anteriores explicando si se ha considerado algún supuesto semántico adicional. Si alguna especificación del enunciado no ha podido reflejarse en el esquema, hacerlo constar.

DISCUSIÓN DEL ENUNCIADO

Comenzaremos por reflejar la semántica relativa a los lotes, barcos y caladeros (Figura 1.46). En el enunciado se distingue claramente que existen las entidades *LOTE* (identificada por un *Cod_lote*), *ESPECIE* (identificada por un *Cod_especie*), *CALADERO* (identificado por un *Nombre*) y *BARCO* (identificado por una *Matrícula* y que además tiene *CIF* como Identificador Alternativo)

En cuanto a las interrelaciones, la interrelación 1:N **Contiene** refleja que cada lote contiene una única especie. Además, existe una interrelación ternaria N:M:P **Captura** que almacena información sobre qué especie, captura cada barco y en qué caladero, guardánsolo los kilos y en qué periodo de tiempo se hizo la captura. Obsérvese, además, que son atributos multivaluados puesto que se dará el caso de que para una misma especie, barco y caladero se produzcan varias capturas en distintas épocas, siendo la fecha de inicio la que determine el número de kilos capturados cada especie y la fecha de fin. También se puede suponer que la fecha de fin en la que un barco captura una determinada especie en un caladero puede implicar la fecha de inicio en la que comenzó la captura y el número de kilos capturados. El tipo de correspondencia de **Captura** es N:M:P pues dada una especie en un caladero puede ser capturada por varios barcos; un barco en un determinado caladero puede capturar varias especies y un barco puede capturar una determinada especie en varios caladeros

También se almacena información sobre los compradores, por lo que se representa la entidad *COMPRADOR* (idenficada por un *Cod_comprador*). La interrelación N:M **Adquirido** almacena el *Precio/kilo final* y el *Precio total* al que el comprador ha adquirido el lote, que se derivará de multiplicar el valor de los atributos *Kilos* de la entidad *LOTE* por el valor del atributo *Precio/kilo final*. La fórmula que calcula el valor del atributo derivado no se puede representar en el diagrama E/R.

A continuación describiremos el diagrama E/R correspondiente a los distintos tipos de facturas que se manejan en el sistema. Como las facturas tienen atributos en común se ha representado una jerarquía parcial y exclusiva con *FACTURA* como supertipo (con *Num_fact* como identificador principal) y dos subtipos, *FACTURA COMPRADOR* (que tiene como atributo propio *Fecha_pago*, que representa la fecha de vencimiento de la factura y se representa

como un atributo opcional) y *FACTURA BARCO*. En el diagrama E/R ha sido imposible recoger la semántica de que la fecha de pago de una determinada factura de comprador nunca puede ser posterior a la fecha de emisión de la factura.

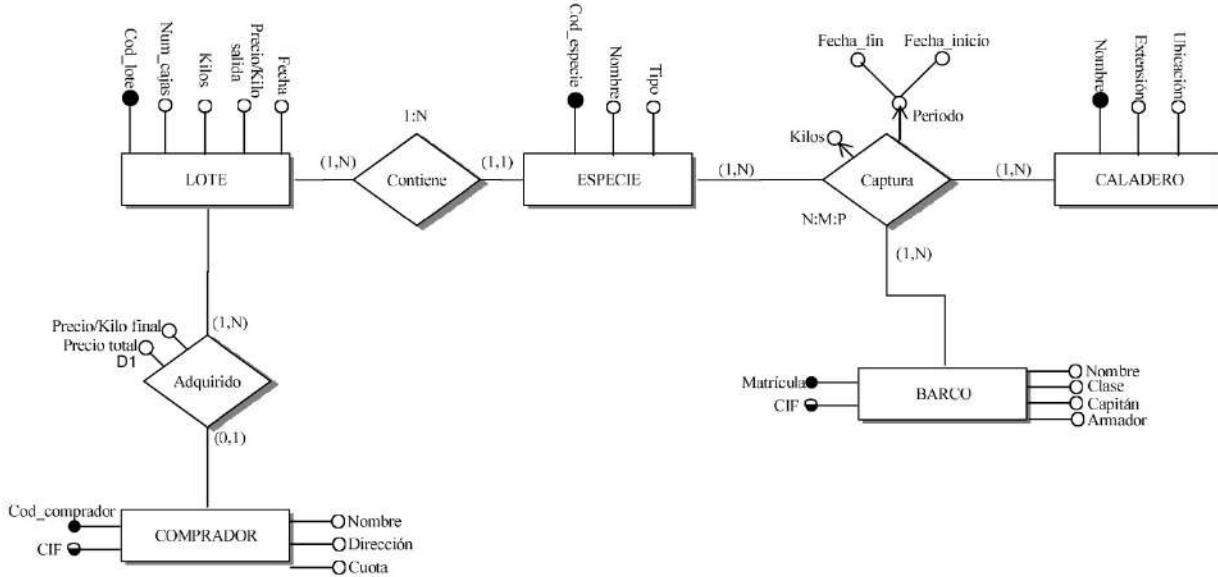


Figura 1.46. Vista parcial del diagrama E/R para la gestión de una lonja de pescado

La Figura 1.47 muestra que un Lote procede de un único Barco (interrelación 1:N **Pescado**), que un Lote se factura como máximo en una Factura Barco (interrelación 1:N **Produce**) y que un Lote se cobra como máximo a una Factura Comprador (interrelación 1:N **Genera**).

Se observa que hay un ciclo formado por las interrelaciones **Pescado**, **Produce** y **Emite**. Estas tres interrelaciones están asociadas semánticamente, lo que permite estudiar una posible redundancia. Se comprueba que la semántica de interrelación **Emite** se puede obtener a partir de **Pescado** y **Produce** ya que podemos conocer las facturas que emite un barco a través de los lotes que ha pescado y en qué facturas aparecen estos lotes. Así mismo, podemos saber qué barco emitió una determinada factura a través de los lotes que aparecen en la factura y comprobando de qué barco son en la interrelación **Pescado**.

También se puede observar que existe otro ciclo formado por las interrelaciones **Pescado**, **Contiene** y **Captura**. En este caso, las interrelaciones también están asociadas semánticamente, pero esta vez no existe ninguna redundancia. Si se eliminara la interrelación **Pescado** no se recogería en ningún sitio cuáles son los lotes pescados por cada barco, ya que la interrelación

Contiene solamente posee información de las especies que contiene cada lote, pero no qué barco las capturó. Si se eliminara la interrelación **Contiene** no se sabría qué especies contiene cada lote de entre todas las capturadas por el barco. Por último, la interrelación **Captura** no se puede eliminar ya que, entre otras cosas, relaciona los caladeros donde se capturan las especies.

Lo que sí que se observa es que será necesario comprobar que un lote no puede haber sido pescado por un barco que nunca haya capturado especies que contenga el lote. En este caso no se puede realizar ningún supuesto acerca de las fechas de captura y ventas de los lotes ya que no se aclara explícitamente en el enunciado. Si se supusiera, por ejemplo, que toda especie capturada se ha de vender en el mismo día, sería necesario comprobar que un lote no puede contener más kilos de una determinada especie de los que ha capturado el barco en el mismo día.

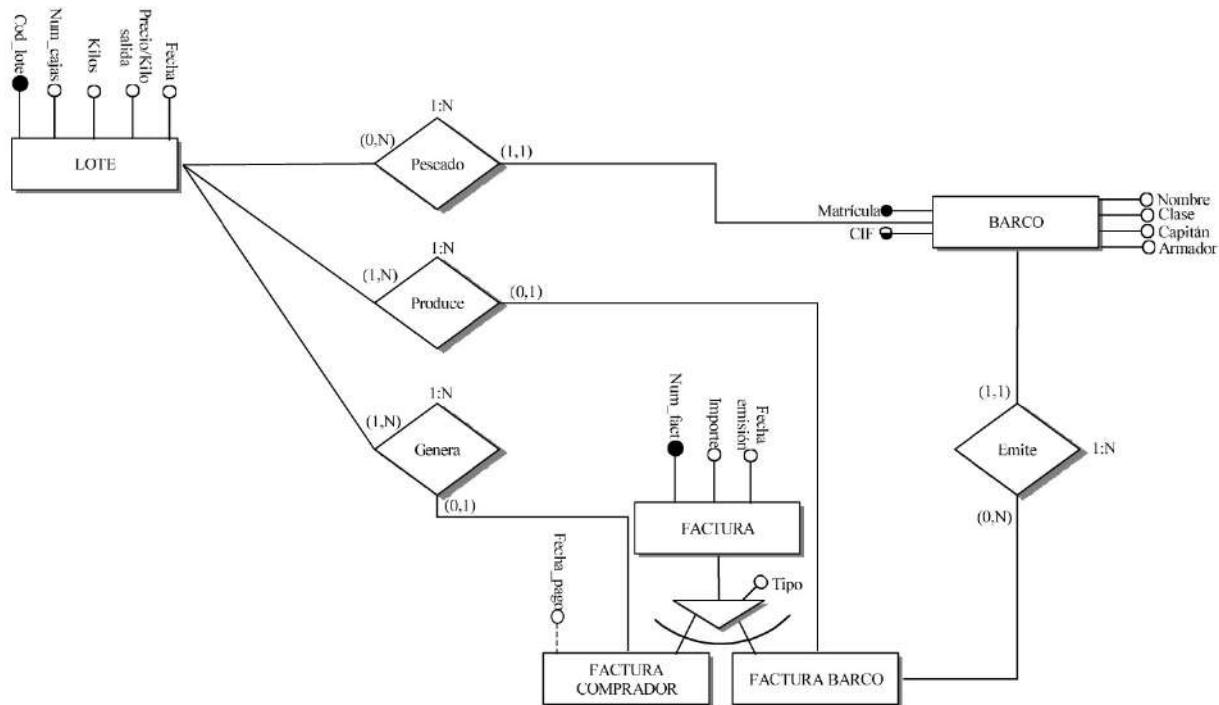


Figura 1.47. Vista parcial del diagrama E/R para la gestión de una lonja de pescado

La última parte del diagrama E/R se muestra en la Figura 1.48 y se corresponde con la representación de los tipos de compradores que asisten a la lonja. Se ha utilizado una jerarquía para reflejar los dos tipos de compradores, **CON CRÉDITO** y **SIN CRÉDITO**. Ambas entidades se asocian mediante dos interrelaciones exclusivas con la entidad **FACTURA COMPRADOR**, es decir, una factura de comprador solo puede asignarse a un determinado tipo de cliente. No

se puede representar en el diagrama E/R que toda ocurrencia de la entidad *FACTURA COMPRADOR* ha de ser pagada por un comprador (a crédito o al contado). Además, también será necesario comprobar que la fecha de vencimiento de un determinado comprador con crédito nunca puede ser anterior a la fecha de pago de sus facturas. Otro supuesto semántico que se no puede incorporar al diagrama E/R es que la fecha de emisión de una determinada factura de un comprador a crédito nunca debería ser posterior a la fecha de vencimiento del crédito del comprador.

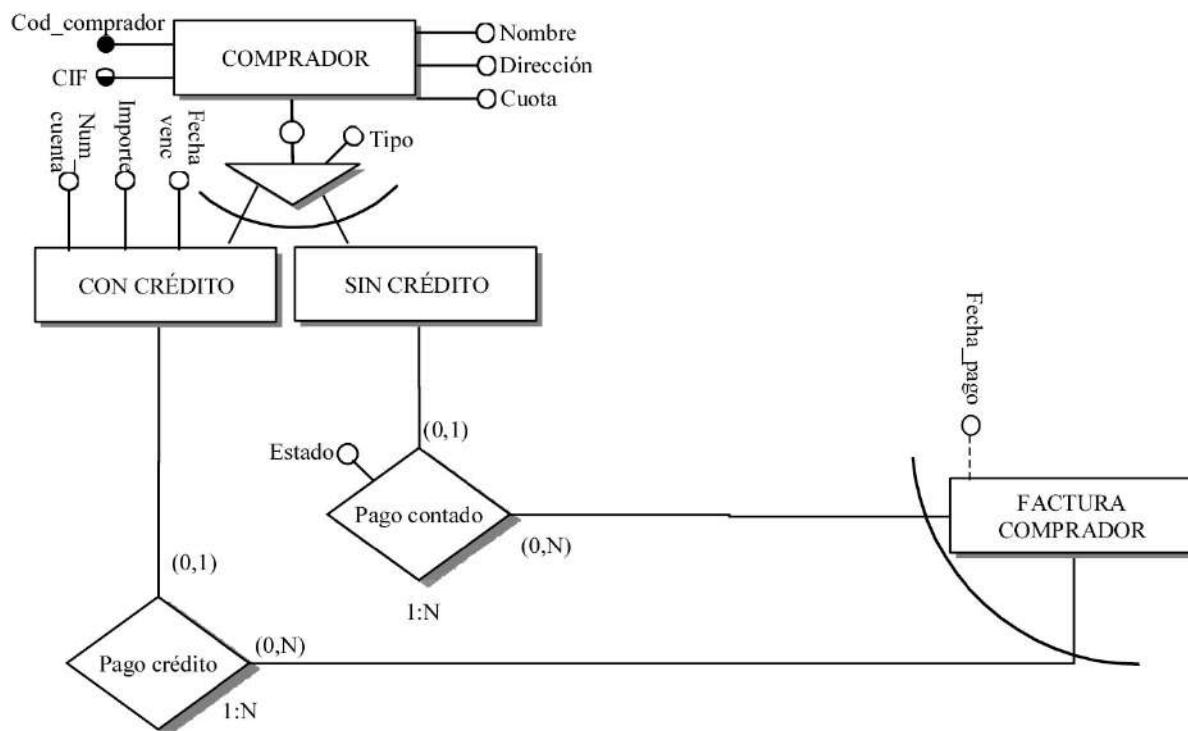


Figura 1.48. Vista parcial del diagrama E/R para la gestión de una lonja de pescado

SOLUCIÓN PROPUESTA

DIAGRAMA E/R

La Figura 1.49 muestra el diagrama E/R completo que se propone como solución.

SUPUESTOS SEMÁNTICOS NO RECOGIDOS

- SS1. Todos los lotes que aparecen en una factura de un barco se corresponden a un mismo barco.
- SS2. Los periodos de tiempo en los que un barco captura una determinada especie en un caladero no se pueden solapar. Además, la fecha de inicio en la que un barco captura una determinada especie en un caladero implica una única fecha de fin de captura y el número de kilos capturados. De igual manera, la fecha de fin en la que un barco captura una determinada especie en un caladero puede implicar la fecha de inicio en la que comenzó la captura y el número de kilos capturados.
- SS3. En la interrelación Adquirido el valor del atributo *Precio_total* se calcula multiplicando el valor del atributo *Precio/kg_final* por el atributo *Kilos* de la entidad *LOTE*.
- SS4. Un lote no puede haber sido pescado por un barco que nunca haya capturado especies que contenga ese lote.
- SS5. Toda ocurrencia de la entidad *FACTURA COMPRADOR* ha de ser pagada por un comprador (a crédito o al contado).
- SS6. La fecha de pago de una determinada factura de comprador nunca puede ser posterior a la fecha de emisión de la factura.
- SS7. La fecha de emisión de una determinada factura de un comprador a crédito nunca debería ser posterior a la fecha de vencimiento del crédito del comprador.
- SS8. El atributo *Tipo* de la jerarquía de *FACTURA* toma valores en el domino *TIPO_FACTURA*=*{factura_comprador, factura_banco}*.

SS9. El atributo *Tipo* de la jerarquía *COMPRADOR* toma valores en el domino
TIPO_COMPRADOR={con_crédito, sin_crédito}.

 No se puede realizar ningún supuesto acerca de la fecha de captura de las especies que componen los lotes y la fecha de los lotes debido a que el enunciado no especifica el tiempo de venta de los mismos.

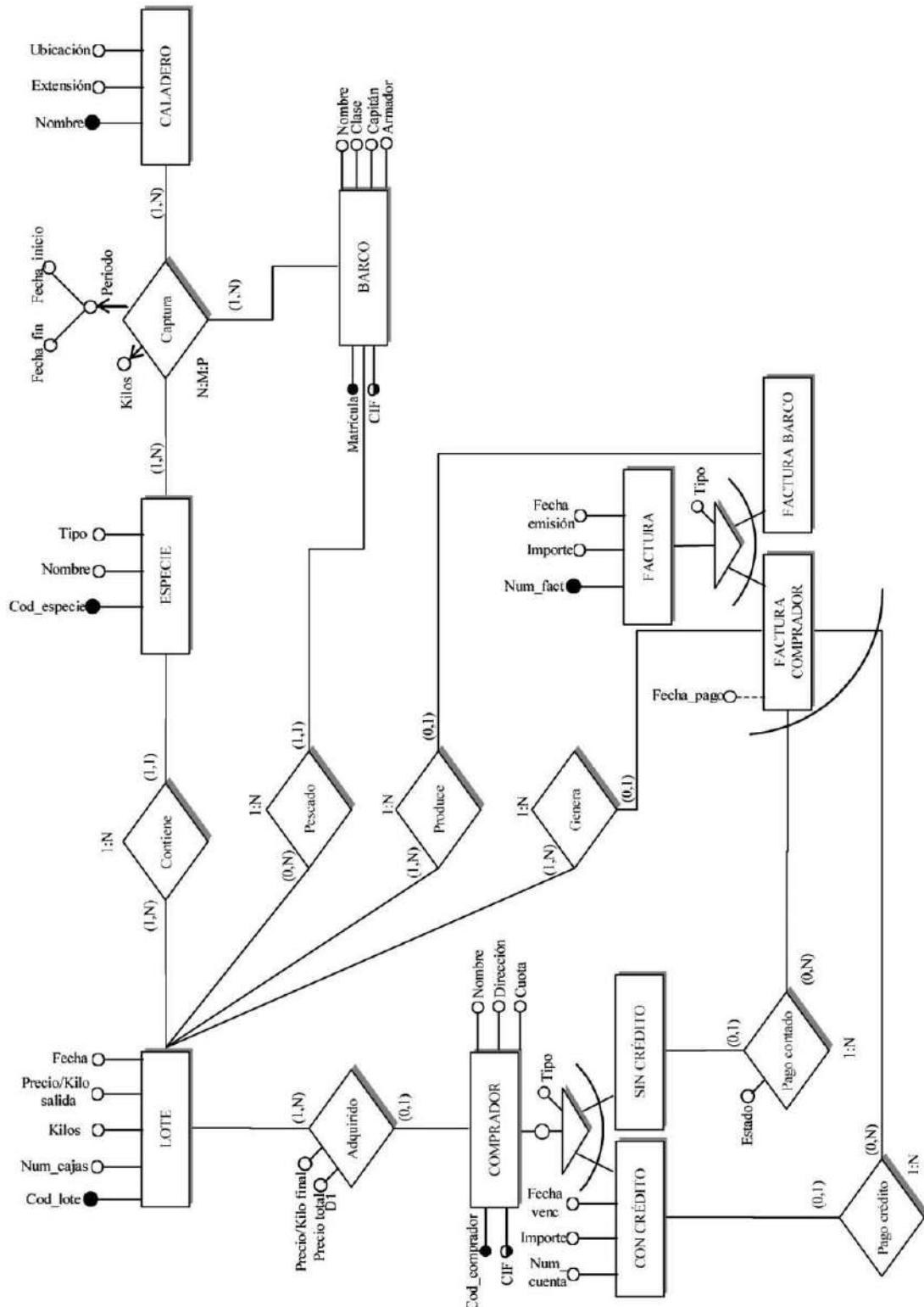


Figura 1.49. Diagrama E/R para la gestión de una lonja de pescado

PROBLEMA 1.10: VIVEROS

Se desea diseñar una Base de Datos para gestionar los empleados y productos a la venta de una cadena de viveros dedicados a la venta diversos productos relacionados con la jardinería. Los supuestos que hay que recoger en la BD son los siguientes:

La cadena de viveros dispone de varios viveros en la provincia de Madrid identificados por un código de tienda y de los que se almacenará un teléfono, una dirección y un responsable que será uno de los empleados que trabaja en el vivero (es necesario almacenar durante qué periodos de tiempo ha sido responsable cada empleado).

Los productos que se venden tienen asignado un código de producto nos interesa guardar el precio y el *stock* que hay de cada producto en cada uno de los viveros y pueden ser de tres tipos: plantas de las que se guardará su nombre, y una breve descripción de los cuidados que requiere; accesorios de jardinería y artículos de decoración. Estos productos se distribuyen en zonas dentro de cada vivero cada una de ellas identificadas por un nombre dentro de cada vivero (zona exterior regadio, interior climatizada, zona de caja, etc.). Se desea conocer el *stock* de cada producto de acuerdo a las zonas del vivero.

Los empleados estarán asignados a una determinada zona en un vivero la cual podrá cambiar a lo largo del tiempo (se guardará histórico de ello) y además, los empleados pueden moverse de un vivero a otro según las necesidades en distintos periodos de tiempo. De los empleados se quiere conocer su DNI, su nombre y un teléfono de contacto.

En cuanto al proceso de venta de los distintos productos, solo se almacenarán los pedidos que realizan los clientes pertenecientes al Club VIP que es una promoción especial que permite a los clientes obtener descuentos según las cuantías de sus compras. De estos clientes se almacena su DNI, su nombre, dirección, teléfono y la fecha de incorporación al club así como los datos de sus pedidos que incluyen un número de pedido, la fecha de realización, los productos adquiridos junto con las unidades y el descuento realizado; por último, también se incluye el precio de los portes en caso de que se hayan contratado. De cada cliente se almacenarán todos los pedidos que haya realizado hasta la fecha.

En cuanto a estos pedidos de clientes pertenecientes al Club VIP interesa

también guardar quién fue el empleado que lo gestionó y en qué vivero se realizó el pedido teniendo en cuenta que un pedido en un determinado vivero lo gestiona un único empleado.

Se pide:

Realizar el esquema E/R extendido correspondiente a los supuestos anteriores explicando si se ha considerado algún supuesto semántico adicional. Si alguna especificación del enunciado no ha podido reflejarse en el esquema, hacerlo constar.

DISCUSIÓN DEL ENUNCIADO

La Figura 1.50 muestra el diagrama E/R propuesto para recoger lo supuestos semánticos dados en el enunciado. Comenzaremos con la representación de los viveros y las zonas que los componen. La entidad fuerte **VIVERO** almacena la información de cada una de las tiendas y **ZONA** es una entidad débil con una dependencia en identificación respecto a **VIVERO** debido a que los nombres de las zonas se repiten en los distintos viveros de la empresa.

Además también almacena información de los productos que se venden en el vivero en la entidad **PRODUCTO** para la que existe un jerarquía con un único subtipo **PLANTA** que almacena información propia (*Nombre* y *Descripción*); los tipos de producto accesorios de jardinería y artículos de decoración no tienen atributos propios ni tampoco van a participar en interrelaciones en el esquema y por ello no deben aparecer como subtipos de **PRODUCTO**.

La interrelación **Ubicado** representa en qué zona de un viviero está un determinado producto y en qué cantidad (atributo *Stock*) de tal manera que un producto puede estar en varias zonas de un vivero (en el enunciado no se establece ninguna restricción al respecto).

En cuanto a los empleados del vivero se desea mantener un histórico de asignaciones de empleados a zonas en los viveros y en qué períodos de tiempo. Para ello se tiene la interrelación N:M **Asignado** entre **EMPLEADO** y **ZONA** y que contiene un atributo multivaluado de tipo periodo de tiempo para contemplar los distintos períodos de tiempo que un empleado puede estar asignado a una misma zona de un vivero. Se supone que la fecha de finalización del contrato siempre es conocida, por lo que el atributo será obligatorio. También se supone que la fecha de inicio en la que un empleado se encuentra asignado a una zona implica una única fecha de finalización; de igual forma se supone que la fecha de finalización de la asignación de un empleado a una zona también implica una única fecha de inicio, siendo siempre la fecha de inicio anterior o igual a la fecha de finalización del periodo. Además, un empleado nunca puede estar asignado a más de una zona al mismo tiempo, es decir, los períodos de tiempo en los que un empleado está asignado a una zona nunca se pueden solapar. Estos supuestos semánticos no se pueden representar en el diagrama E/R.

Por otro lado, la interrelación **Responsable** también tiene un atributo multivaluado de tipo periodo de tiempo para almacenar el histórico de

responsable. Esta interrelación representa que un empleado puede no haber sido nunca responsable de un vivero y que un vivero a lo largo del tiempo puede haber tenido varios responsables en distintas fechas (además el mismo empleado puede haber sido responsable varias veces). Al igual que en la interrelación anterior, se supone que la fecha de finalización siempre es conocida (atributo obligatorio) y también se supone que la fecha de inicio implica una única fecha de finalización y la fecha de finalización implica una única fecha de inicio, siendo la fecha de inicio siempre anterior o igual a la fecha de fin. De igual forma, se ha de comprobar que un empleado nunca puede ser responsable de más de un vivero al mismo tiempo y que un vivero no tiene a más de un responsable al mismo tiempo. Estos supuestos semánticos tampoco se pueden representar en el diagrama E/R.

La entidad *PEDIDO* almacena la información concerniente al número, fecha y descuento y contiene varios productos (interrelación **Incluye**), lo realiza un único *CLIENTE VIP* (interrelación 1:N **Realiza**), se realiza en un único *VIVERO* (interrelación 1:N **Hecho**) y lo gestiona un único *EMPLEADO* (interrelación **Gestiona**).

SOLUCIÓN PROPUESTA

SUPUESTOS NO RECOGIDOS EN EL DIAGRAMA E/R:

A continuación se enumeran los supuestos que no han podido reflejarse en el diagrama E/R:

- SS1. En un determinado momento un vivero solo tiene un empleado responsable.
- SS2. Los periodos de tiempo en los que un empleado es responsable de un vivero no se pueden solapar.
- SS3. Además, la fecha de inicio en la que un empleado es responsable de un vivero implica una única fecha de finalización y la fecha de finalización implica una única fecha de inicio.
- SS4. La fecha de inicio en la que un empleado es responsable de un vivero ha de ser anterior o igual a la fecha de fin.
- SS5. El empleado responsable de un vivero está asignado a la zona de ese vivero.
- SS6. Los periodos de tiempo en los que un empleado está asignado a las zonas de un vivero (o de distintos viveros) no pueden solaparse, es decir, un empleado no puede estar asignado a más de una zona a la vez.
- SS7. Además, la fecha de inicio en la que un empleado se encuentra asignado a una zona implica una única fecha de finalización; de igual forma se supone que la fecha de finalización de la asignación de un empleado a una zona también implica una única fecha de inicio.
- SS8. La fecha de inicio en la que un empleado está asignado a una zona ha de ser anterior o igual a la fecha de fin.
- SS9. Un empleado no puede ser responsable de más de vivero a la vez (este supuesto semántico se controla con los supuestos semánticos SS5 y SS6).
- SS10. La fecha de incorporación de un cliente VIP tiene que ser anterior a todos sus pedidos.
- SS11. El empleado que gestiona un pedido trabaja en el vivero en el que ha

realizado el pedido en las fechas del mismo.

DIAGRAMA E/R:

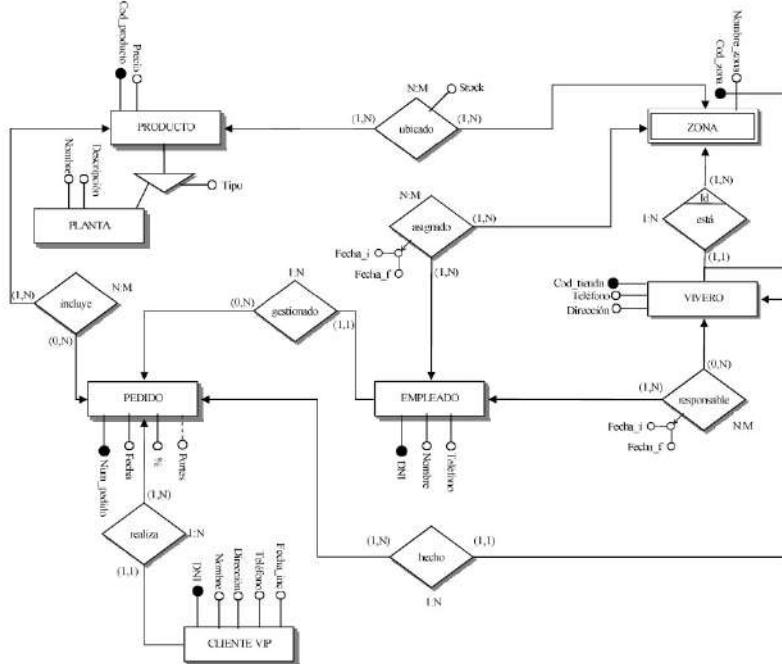


Figura 1.50. Diagrama E/R propuesto para la gestión de un vivero

¹ Que, obviamente, siempre implica una Dependencia en Existencia.

² Otras posibles restricciones de cardinalidad son $(0, Num1)$, $(1, Num1)$, $(Num1, N)$, $(Num1, Num2)$ con $Numi = 2,3,\dots$

³ En el Modelo Relacional, Clave Primaria.

⁴ En el Modelo Relacional, Clave Alternativa.

⁵ A la etiqueta se le podrá asignar la fórmula o procedimiento de derivación.

⁶ Esta aproximación se utilizará en el primer ejercicio de este tema.

⁷ Aunque en la realidad existen otros gastos sobre los que habría que guardar información (comisiones bancarias, gastos extras, etc.), en este caso solo se tendrán en cuenta los aquí expuestos.

⁸ Se define unidosis como la unidad de consumo de cada fármaco, es decir, si un envase de Aspirinas tiene 30 comprimidos, entonces son 30 unidosis.

⁹ Suponemos que los medicamentos recetados son los consumidos, es decir, no hay que almacenar por separado los consumos y por otro lado los recetados.

Capítulo 2

DISEÑO LÓGICO

2.1 RECORDATORIO DE LA ESTÁTICA DEL MODELO RELACIONAL

El elemento central del modelo relacional es la **Relación**. Una relación tiene un nombre, un conjunto de atributos que representan sus propiedades y un conjunto de tuplas que incluyen los valores que cada uno de los atributos toma para cada elemento de la relación. Otro elemento importante dentro del modelo es el **Dominio**, que también tiene entidad propia al igual que la relación, y que describe los valores válidos que un atributo definido sobre este dominio puede tomar. Una relación puede representarse como una tabla de dos dimensiones (las columnas son los atributos de la relación y las filas son las tuplas) con un único valor en cada celda intersección. Esta manera de representación se denomina por extensión, y va variando a lo largo del tiempo puesto que las tuplas de una Relación pueden verse afectadas por operaciones de actualización. A continuación se muestra un ejemplo de relación, denominada *ALUMNOS* y representada por extensión esta relación consta de los atributos *cod_matricula*, *nombre*, *ciudad* y *cod_grupo* (nombres de las columnas de la tabla). Cada fila de la tabla muestra una tupla correspondiente a los datos de un alumno (describe la ocurrencia de un individuo en el mundo real).

ALUMNOS

COD_MATRICULA	NOMBRE	CIUDAD	COD_GRUPO
101	Juan Montero	Alcorcón	11
102	Alicia Cristóbal	Leganés	11
202	Ana Vallejo	Leganés	21
300	Ignacio López	Móstoles	31
103	Leticia Martínez	Alcorcón	--

Los atributos de la relación se definen sobre dominios formados por un conjunto de valores homogéneos y atómicos. Por ejemplo, el dominio del *cod_matricula* es *CODIGOS*, formado por el conjunto de cadenas de longitud 3 caracteres. También es posible definir dominios por extensión (enumerando los posibles valores de los que consta el dominio) o bien indicando un rango de

valores sobre un tipo base.

Existe otra forma de representar una relación en el modelo relacional que se denomina por intención o grafo de relación, en la cual solo se exponen las partes estáticas de la misma. De manera detallada, estas partes son: el nombre de la relación, el nombre de los atributos, y el nombre del dominio sobre el cual están definidos (aunque este último, en general, se omite). Para el ejemplo anterior, el grafo de relación sería:

ALUMNOS (cod-matricula, nombre, ciudad, cod-grupo)

El modelo relacional impone una serie de *restricciones inherentes*:

- a) En una relación no puede haber dos tuplas iguales (obligatoriedad de clave primaria).
- b) El orden de las tuplas y el de los atributos no es relevante.
- c) Cada atributo solo puede tomar un único valor del dominio sobre el cual está definido (no hay grupos repetitivos).
- d) Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo (regla de integridad de entidad).

Por otra parte, los mecanismos que proporciona el modelo relacional para recoger *restricciones semánticas* o de *usuario* son:

- a) La restricción de *clave primaria* (*PRIMARY KEY*) permite declarar un atributo o conjunto de atributos como la clave primaria de una relación que identifica únicamente cada tupla de la relación. Por la restricción inherente (a) es obligatorio definir en toda relación una clave primaria, y ésta no puede adoptar valores nulos. La notación utilizada en este libro para representar la clave primaria en una relación es subrayar los atributos que forman parte de ella. En el ejemplo mostrado anteriormente de la relación *ALUMNO*, la clave primaria es el atributo *cod_matricula*.
- b) La restricción de *unicidad* (*UNIQUE*) nos permite definir claves de identificación alternativas (los valores de uno o varios atributos no pueden repetirse en diferentes tuplas de una relación). La notación que se seguirá para representar la unicidad es la del subrayar de manera discontinua el atributo o atributos que la componen.

- c) La restricción de *obligatoriedad* (*NOT NULL*) permite declarar si uno o varios atributos de una relación deben tomar siempre un valor, es decir, no pueden tomar valores nulos. Por defecto, en el grafo relacional los atributos son obligatorios, por lo que la notación utilizada para los atributos opcionales se muestra con un asterisco.
- d) La restricción de *clave ajena* (*FOREIGN KEY*), también denominada *integridad referencial*, se utiliza para, mediante claves ajenas (conjunto de atributos en una relación que es una clave primaria en otra o la misma relación) enlazar relaciones de una base de datos. La integridad referencial indica que los valores de la clave ajena en la relación que referencia deben corresponderse con alguno de los valores existentes de la clave (primaria o alternativas) definida (o tener valor nulo, si se permite) en la relación referenciada. Si no se define una clave en la relación referenciada, por defecto se asumirá la referencia sobre la clave primaria. Los atributos que son clave ajena en una relación no necesitan tener los mismos nombres que los atributos de la clave primaria con la cual ellos se corresponden. En el grafo relacional, la notación utilizada es a través de una flecha, que desde la clave ajena termina en la relación a la que referencia (si los valores son tomados de la clave primaria) o en los atributos que forman parte de la clave alternativa a la que referencia.

A través de las claves ajenas se pueden expresar los distintos tipos de vínculos que pueden aparecer entre dos relaciones. Estos vínculos se pueden tipificar en vínculos de uno a varios o de varios a varios, principalmente. A continuación se muestra un ejemplo de un tipo de vínculo uno a varios con las relaciones *EMPLEADOS* y *DEPARTAMENTOS*:

EMPLEADOS (dni, nombre, apellidos, direccion, telefono, departamento)

→ DEPARTAMENTOS (numero-dept, nombre)

Figura 2.1. Tipo de vínculo uno a varios

Las claves primarias de la relación *EMPLEADOS* y *DEPARTAMENTOS* son *DNI* y *numero_dept*, respectivamente. El atributo *departamento* de la relación *EMPLEADO* es una clave ajena que referencia a la relación *DEPARTAMENTOS*, es decir, los valores del atributo *departamento* deben corresponderse con los valores del atributo de la relación a la que referencia, que

en este caso, es la clave primaria *numero_dept* de la relación *DEPARTAMENTOS*. Algunas tuplas de estas relaciones se muestran a continuación:

EMPLEADOS

NOMBRE	DEPARTAMENTO	SALARIO	FECHA_NAC	EXT_TLF
Pablo Montero	14	600	10-11-67	6543
Beatriz Cristóbal	13	3000	20-9-68	6577
J. Luis Martín	11	700	25-6-77	6433
Almudena López	13	1000	4-5-60	6422
Ángel Vallejo	14	2000	15-4-72	6321
Pedro García	11	700	12-3-70	6323

DEPARTAMENTOS

NUMERO_DEPT	NOMBRE
11	Contabilidad
13	Marketing
14	Informática

La integridad referencial asegura que los empleados de la relación *EMPLEADOS* solo pueden trabajar en departamentos que hayan sido dados de alta en la relación *DEPARTAMENTOS*.

Además de la integridad referencial, que permite enlazar relaciones entre sí dando lugar a la estructura de la BD, el modelo relacional permite también definir las opciones de borrado y modificación en las claves ajenas. Estas opciones indican las acciones que hay que llevar a cabo cuando se produce un

borrado o modificación de una tupla en la relación referenciada por una relación (lo que haría peligrar la integridad de la BD, al dejar de existir valores en la clave referenciada). Las posibilidades para una operación de actualización (borrado o modificación) son:

- **Borrado/modificación sin acción (NOT ACTION):** SQL3 permite definir la opción *RESTRICT* y *NOT ACTION*. La primera actúa como: si existen tuplas en la relación hija relacionadas con la tupla de la relación padre sobre la que se realiza la operación, entonces no se permitirá llevar a cabo dicha operación. Y la opción *NOT ACTION* es similar a la anterior (pero puede diferir su comprobación). Por lo que solo se impide la operación si al final de la misma el resultado rompe la integridad referencial. En el ejemplo anterior, no se podría borrar un departamento que tenga empleados que trabajen en él.
- **Borrado/modificación en cascada (CASCADE):** el borrado (o modificación) de una tupla en la relación padre ocasiona un borrado (o modificación) de todas las tuplas relacionadas en la relación hija (tuplas cuya clave ajena coincide con el valor de la clave primaria de la tupla eliminada o modificada en la relación padre). Así, en el ejemplo anterior, el borrado de un departamento supone un borrado de todos los empleados que trabajan en él.
- **Borrado/modificación con puesta a nulos (SET NULL):** esta posibilidad nos permite poner el valor de la clave ajena referenciada a *NULL* cuando se produce el borrado o modificación de una tupla en la relación padre. Continuando con el ejemplo anterior, el borrado de un departamento implicaría que todos los empleados que trabajan en él pasarán a contener *NULL* en el atributo *Departamento*. Evidentemente, esta opción está permitida siempre y cuando la clave ajena admita nulos.
- **Borrado/modificación con puesta a un valor por defecto (SET DEFAULT):** su funcionamiento es similar al caso anterior, con la excepción de que el valor al que se ponen las claves ajenas referenciadas es un valor por defecto que se habrá especificado en la definición de la tabla correspondiente. En este caso, en el ejemplo anterior, cuando se borra un determinado departamento es posible asignar sus empleados a un departamento ficticio (que debe encontrarse evidentemente presente en la relación *DEPARTAMENTOS*).

Las opciones de borrado y modificación pueden ser distintas para una determinada clave ajena de una relación; por ejemplo, es posible definir el borrado en cascada y la modificación con puestas a nulos.

- **Restricciones de verificación:** en algunos casos puede ocurrir que sea necesario especificar una condición que deben cumplir los valores de determinados atributos de una relación de la BD aparte de las restricciones ya vistas de clave primaria, unicidad, obligatoriedad y clave ajena. Para ello se definen las restricciones de verificación *CHECK* que, al igual que las ya estudiadas, siempre llevan implícitas un rechazo en caso de que no se cumpla la condición especificada y que también se comprueban ante una inserción, borrado o modificación. Como las restricciones de verificación siempre van ligadas a un único elemento de la BD (generalmente una relación) no es necesario que tengan un nombre. Por ejemplo, para la relación *EMPLEADOS* podría definirse una restricción que comprobara que el sueldo de un empleado siempre ha de ser mayor o igual al salario mínimo (600 euros). Así, si se va a insertar un empleado con un sueldo de 300 euros, la operación se rechazaría.

Otro tipo de restricción, que generaliza la anterior, lo forman las aserciones (*ASSERTION*) en las que la condición se establece sobre elementos de distintas relaciones. Se define como un objeto relacional que tiene existencia por sí mismo. Los *CHECK* sin embargo siempre están asociados a una relación. Su funcionamiento es similar al de las restricciones de verificación *CHECK*. La única diferencia consiste en que el ámbito de un *CHECK* es una única relación a nivel de tupla. Continuando con nuestro ejemplo, se podría especificar una asercción para la restricción que establezca que “no hay ningún empleado que trabaje en el departamento de contabilidad que gane más de 10.000 euros”.

En las restricciones que involucran algún tipo de subconsulta en la condición (por ejemplo, si se utiliza alguna función de agregación sobre las tuplas de una relación) puede ocurrir que puedan implementarse como un *CHECK*. Sin embargo, es necesario estudiar que la restricción no pueda ser incumplida en ningún caso por cualquiera de las operaciones que puedan realizarse sobre el esquema. Por ello, a veces es necesario implementarlas como una Aserción

(Ullman y Widom (1997)).

- **Disparadores (TRIGGER):** por último, a veces puede interesar especificar una acción distinta del rechazo cuando no se cumple una determinada restricción semántica¹⁰. En estos casos, se recurre al uso de los disparadores o *triggers* que nos permiten, además de indicar una condición, especificar la acción que queremos se lleve a cabo si la condición se hace verdadera. Los disparadores como reglas de tipo evento-condición-acción (ECA) que son, pueden interpretarse como reglas que especifican que cuando se produce un evento, si se cumple una condición, entonces se realiza una determinada acción. Los disparadores están contemplados en el SQL3 y existen productos relacionales que los soportan, aunque con algunas diferencias en el modo de funcionamiento. En nuestro ejemplo, se podría definir un disparador que “informará al administrador de la BD cuando haya un empleado que trabaje en el departamento de Marketing que gane más de 7.000 euros”. Como veremos más adelante también son convenientes para mantener algunas restricciones del esquema E/R que no tienen una transformación directa a un esquema relacional.

2.1.1 Notación

Un esquema relacional se representará mediante un grafo, conocido como grafo o esquema relacional. Básicamente, se trata de un grafo dirigido cuyos nodos son las relaciones de la BD y los arcos representan las restricciones de clave ajena y en el que aparecerán además de las distintas relaciones con sus atributos y las restricciones de clave ajena las restricciones de clave primaria, unicidad y obligatoriedad. Las convenciones empleadas para la representación de este grafo son:

1. El nombre de las tablas está representado en mayúsculas y el de los atributos en minúsculas. Primero aparece el nombre de la relación y a continuación sus atributos entre paréntesis.
2. Las claves primarias aparecen subrayadas.
3. Las claves alternativas aparecen subrayadas con una línea discontinua.
4. Las claves ajenas apuntan a través de una flecha a la tabla que

referencian (si la clave toma valores de la clave primaria de esta relación) o a una clave alternativa (si toma valores de esta clave).

5. Los atributos que pueden tomar valores nulos aparecen con un asterisco.

6. Las opciones para la integridad referencial son:

- DNA, DC, DSN, DSD para el borrado sin acción, en cascada, con puestas a nulos o con puesta a valor por defecto, respectivamente.
- UNA, UC, USN, USD para la modificación sin acción, en cascada, con puestas a nulos o con puesta a valor por defecto, respectivamente.

2.2 REGLAS DE TRANSFORMACIÓN DE UN ESQUEMA E/R A UN ESQUEMA RELACIONAL

Las tres reglas básicas empleadas para transformar un esquema conceptual E/R en un esquema relacional son:

1. Toda entidad se transforma en una relación.
2. Las interrelaciones N:M se transforman en una relación.
3. Las interrelaciones 1:N dan lugar o bien a una propagación de clave o bien a una relación.

A continuación, se describirán brevemente estas tres reglas, empleando para ello el esquema E/R que se muestra en la Figura 2.2 y que se corresponde con la siguiente descripción:

Una compañía ferroviaria necesita una base de datos para gestionar su taller en el que se llevan a cabo las revisiones periódicas y reparaciones de las locomotoras. La información que debe contener la base de datos se describe en los siguientes supuestos semánticos.

El taller dispone de varias áreas de reparación (motores diésel, motores eléctricos, transmisiones, etc.) caracterizadas por un número, una localización y un tamaño en metros cuadrados.

Las reparaciones de las locomotoras (código de locomotora, fecha de comienzo de funcionamiento y nombre) se desglosan en un conjunto de tareas a las que se asignan diversas áreas de reparación (zonas del taller). Una tarea se caracteriza por su nombre (desmontaje, limpieza, engrase, montaje, etc.), su descripción y su duración y se puede realizar en una o varias áreas de reparación. Cada área de reparación puede ocuparse para realizar distintas tareas de reparación.

Las tareas de reparación son llevadas a cabo por varios operarios (tanto especializados como aprendices). Interesa conocer quiénes son los operarios asignados a una determinada tarea de reparación y área de reparación. De igual manera, hay que recoger las áreas de reparación

ocupadas para llevar a cabo las tareas y los operarios involucrados en ellas (incluyendo las horas dedicadas por cada operario en cada tarea y área de reparación).

De cada reparación se desea guardar un código, el nombre, la duración en horas y la fecha de comienzo. Es importante, además, saber los recambios (tuercas, válvulas, etc.) que se han empleado en la reparación de una determinada locomotora, queriendo conocer de cada recambio el código, descripción, nombre de los proveedores que suministran cada recambio y número de unidades empleadas en cada reparación. Así mismo, se quiere almacenar las herramientas (código de herramienta, descripción y peso) empleadas en cada tarea de reparación junto con las fechas de inicio y fin de utilización de esas herramientas.

Existen operarios especializados y aprendices; de los especializados se quiere conocer los años de experiencia, su DNI, nombre, fecha de nacimiento y número de la Seguridad Social. Los operarios especializados supervisan a los aprendices, interesando almacenar esta información. Un operario aprendiz solo tiene un supervisor y se quiere guardar la fecha en que se inició este seguimiento. De los operarios aprendices se almacenará su DNI, nombre, fecha de nacimiento y número de la Seguridad Social.

Por último, los aprendices asisten a cursos de formación (código de curso, nombre y fecha de inicio) y en cada curso los aprendices son evaluados con varias puntuaciones.

2.2.1 Transformación de entidades, atributos y dominios

Cada **Entidad** del esquema E/R dará lugar a una nueva relación cuya clave primaria es el Identificador Principal de la Entidad.

Cada **Atributo** de una entidad se transforma en un atributo de la relación creada para la Entidad aunque hay que tener en cuenta sus distintos tipos de restricciones semánticas:

- **Atributos Univaluados:** dan lugar a un atributo de la relación. En la Figura 2.2, la entidad *LOCOMOTORA* da lugar a una relación con el mismo nombre y con tres columnas: *cod_locomotora*, *fecha_inicio* y *nombre*.

- **Atributos Multivaluados:** dan lugar a una nueva relación cuya clave primaria es la concatenación de la clave primaria de la entidad en la que se sitúa el atributo multivaluado más el nombre del atributo multivaluado. En ocasiones, si el atributo multivadado no admite repeticiones, es suficiente éste como clave primaria. Por ejemplo, en la Figura 2.2, la entidad *RECAMBIO* tiene el atributo multivaluado *Proveedor*; su transformación al modelo relacional daría lugar a una relación *RECAMBIOS* cuya clave primaria es *cod_recambio* y una relación *PROVEEDORES* cuya clave primaria es la concatenación de *cod_recambio* y *proveedor*.

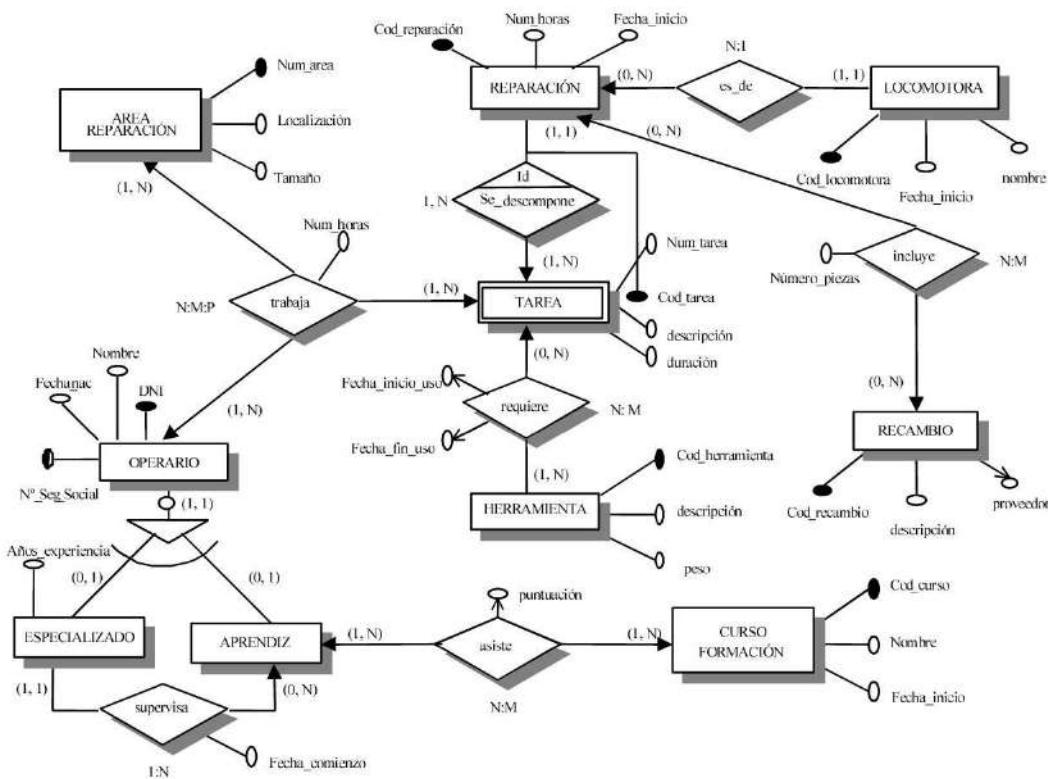


Figura 2.2. Esquema E/R

- **Atributos Obligatorios:** atributos con la restricción de *NOT NULL*.
 - **Atributos Opcionales:** atributos que pueden tomar valores nulos.
 - **Identificador Principal:** atributo(s) que forman la clave primaria.
 - **Identificador Alternativo:** atributo con la restricción de *UNIQUE*. El atributo *Nº_Seg_Social* de la entidad *OPERARIO* (Figura 2.2) es un identificador alternativo de operario.

- **Atributos Derivados:** atributos cuyos valores se obtienen como resultado de algún cálculo sobre otros atributos. En la Figura 2.2, la entidad *REPARACIÓN* tiene el atributo derivado *Num_horas* que se obtiene de la suma de la duración de todas las tareas que tiene asociadas una determinada reparación.
- **Atributos Compuestos:** se transforman en los atributos que los componen (no hay forma de representar un atributo compuesto en el modelo relacional).
- **Dominios:** se transforman en los dominios del Modelo Relacional.

2.2.2 Transformación de interrelaciones N:M

Las interrelaciones N:M dan lugar a una relación cuya clave será la concatenación de los Identificadores Principales de las Entidades que enlaza la interrelación. De esta forma los atributos que forman la clave primaria en esta nueva relación son claves ajenas respecto a las relaciones en las que estos atributos son clave primaria. Por ejemplo, en la Figura 2.2 existe la interrelación N:M **Incluye** y su transformación al Modelo Relacional da lugar a una relación *INCLUYEN* cuya clave primaria es la concatenación de las claves primarias de *REPARACIONES* y *RECAMBIOS*, es decir, *cod_reparación* y *cod_recambio*.

Si la interrelación tiene atributos, estos pasarán a formar parte de la relación creada para la interrelación. Por ejemplo, la interrelación **Incluye** tiene el atributo *Número_piezas*; la relación *INCLUYEN* creada para esta interrelación contiene, además de los atributos que forman la clave primaria, una columna denominada *numero_piezas*.

En el caso de que la interrelación contenga un atributo multivaluado que no denote una dimensión temporal, la clave de la relación deberá incluir también este atributo. Por ejemplo, en la Figura 2.2 la interrelación **Asiste** entre *APRENDIZ* y *CURSO FORMACIÓN* contiene el atributo multivaluado Puntuación que refleja las distintas calificaciones obtenidas por un aprendiz en cada uno de los cursos a los que ha asistido. La transformación al modelo relacional de esta interrelación da lugar a una relación *ASISTEN* cuya clave primaria contiene la concatenación de *DNI*, *cod_curso* y *puntuacion*; sin embargo, puesto que un aprendiz podría obtener iguales calificaciones en un determinado curso, sería necesario añadir un atributo *fecha_evaluación* para distinguirlas y que también formaría parte de la clave primaria.

Sin embargo, en el caso de atributos con una dimensión temporal (generalmente atributos que denotan fechas, horas o intervalos de tiempo), tanto si son multivaluados como univaluados, es necesario estudiar la semántica del universo del discurso con el fin de determinar cuáles van a ser los atributos que formen la clave primaria de la relación a la que da lugar la interrelación. Para ello habrá que estudiar si se trata de interrelaciones que representan información histórica o no. Por ejemplo, en la Figura 2.2, la interrelación **Requiere** entre *TAREA* y *HERRAMIENTA* tiene dos atributos multivaluados (*Fecha_inicio_uso* y *Fecha_fin_uso*) que nos indican que una herramienta puede utilizarse en una tarea de reparación en diferentes intervalos de tiempo. La transformación de esta interrelación da lugar a la relación *REQUIEREN* cuya clave primaria está compuesta por los atributos *cod_reparación* y *num_tarea* procedentes de *TAREA*¹¹, *Cod_herramienta* procedente de *HERRAMIENTA* y el atributo *Fecha_inicio_uso*, es decir, *tarea*, *herramienta* y *fecha de inicio* determinan únicamente cada tupla de la relación *REQUIEREN*. Si en una fecha determinada una herramienta solo pudiera utilizarse en una tarea, entonces la clave de *TAREAS* no formaría parte de la clave de la relación *REQUIEREN*.

También es necesario tener en cuenta cuáles son las cardinalidades mínimas y máximas en cada lado de la interrelación para no perder semántica en la transformación. Estos casos se irán estudiando a medida que vayan surgiendo en los ejercicios propuestos.

2.2.3 Transformación de interrelaciones 1:N

En el caso de las interrelaciones 1:N existen dos posibilidades de transformación:

1. Crear una nueva relación para la interrelación, cuyo tratamiento sería igual que el de las interrelaciones N:M, con la salvedad de que la clave primaria de la nueva relación constará del Identificador Principal de la Entidad que se encuentra en el lado N de la interrelación. Si la interrelación tiene atributos propios, ésta es una solución razonable. Por ejemplo, en la Figura 2.2, la interrelación **Supervisa** podría transformarse en una relación *SUPERVISAN* cuya clave primaria es el atributo *DNI_aprendiz* (pues un operario aprendiz tiene un único supervisor) y que consta además de los atributos *DNI_especializado* y *fecha_comienzo*.

2. Propagar el Identificador Principal desde la Entidad que se encuentra en el lado 1 a la Entidad que se encuentra en el lado N (propagación de clave); si existen atributos en la interrelación, estos también se propagarán. En ausencia de información relativa al volumen de valores nulos que se manejarán o de si existen posibilidades de que la interrelación evolucione o no a una interrelación N:M, esta solución de propagación de clave será la que se adopte en la resolución de los ejercicios propuestos en este libro. Por ejemplo, en la Figura 2.2, la interrelación **Es_de** entre las entidades *REPARACIÓN* y *LOCOMOTORA* se transforma en una propagación de clave (*cod_locomotora*) desde la relación *LOCOMOTORAS* hacia la relación *REPARACIONES* que constaría de los siguientes atributos: *cod_reparacion* (clave primaria), *num_horas*, *fecha_inicio*, *cod_locomotora* (clave ajena).

No hay que olvidar cómo controlar las cardinalidades mínimas y máximas de la interrelación para no perder semántica. Estos casos se irán estudiando a medida que vayan surgiendo en los ejercicios propuestos.

2.2.4 Transformación de otros elementos del Modelo E/R Extendido

A continuación se describen brevemente la transformación de las extensiones del modelo E/R. Un análisis más detallado se encuentra en De Miguel *et al.* (1999).

2.2.5 Transformación de Dependencias en Existencia y en Identificación

Una interrelación 1:N de Dependencia en Existencia origina que la clave ajena, propagada desde la entidad fuerte a la entidad débil, deba tener la opción de borrado en Cascada, es decir, no puede existir ninguna ocurrencia de la entidad hijo que no esté relacionada con una ocurrencia de la entidad padre.

Una interrelación 1:N de Dependencia en Identificación¹² da lugar a una propagación de clave desde la entidad fuerte a la entidad débil de tal forma que la entidad débil requiere de la clave de la entidad fuerte para su identificación. En la Figura 2.2, la dependencia en identificación denotada por la interrelación

Se_decompone hace que la clave de *REPARACIONES* (*cod_reparacion*) se propague a la relación *TAREAS* de tal forma que la clave primaria de esta última está formada por los atributos *cod_reparacion* y *num_tarea*. La relación *TAREAS* consta, además, de los atributos *descripción* y *duración*.

2.2.6 Transformación de Generalizaciones

Aunque existen tres posibilidades para la transformación de jerarquías al modelo relacional, en los casos propuestos en este libro se adoptará preferentemente la que se describe a continuación. Se creará una relación por cada Entidad participante en la jerarquía (una relación para el supertipo y una para cada uno de los subtipos), de tal forma que el supertipo propaga su Identificador Principal a cada uno de los subtipos que pasan a identificarse por el mismo identificador (como clave ajena). La relación creada para el supertipo podrá contener el atributo discriminante de la jerarquía. Por ejemplo, en la Figura 2.2, la generalización entre el supertipo *OPERARIO* y los subtipos *ESPECIALIZADO* y *APRENDIZ* da lugar a tres relaciones; la relación *OPERARIOS* contiene los atributos *DNI* (clave primaria), *nombre*, *fecha_nac*, *n_seg_social* y *tipo*. La relación *ESPECIALIZADOS* tiene como clave primaria el atributo *DNI* (clave ajena) y, además, tiene el atributo propio *años_experiencia*. Por último, la relación *APRENDICES* tiene como único atributo su clave primaria, *DNI*, que también es clave ajena.

Las restricciones semánticas de totalidad/parcialidad y exclusividad/solapamiento también hay que controlarlas en la transformación al Modelo Relacional. La totalidad se controla prohibiendo las inserciones en el supertipo y con un disparador que ante una inserción en un subtipo inserte también en el supertipo (además el atributo discriminante no podría tomar valores nulos). La generalización de la Figura 2.2 es total, pues no puede haber un operario que no sea ni especializado ni aprendiz; así, cuando se hace una inserción en la relación *OPERARIOS* hay que forzar a una inserción en el subtipo correspondiente. En el caso de la parcialidad, el atributo discriminante puede tomar valores nulos y no sería necesario ningún disparador.

En cuanto a la exclusividad se requiere una aserción que compruebe que, si un ejemplar pertenece a uno de los subtipos, entonces no puede pertenecer a los demás. Si se permite solapamiento también es necesaria una aserción similar a la de la exclusividad pero que dé cabida a nuevos valores del atributo discriminante para los casos de solapamiento comprobando que las ocurrencias están en los

subtipos adecuados. Por ejemplo, en la Figura 2.2, la exclusividad entre *ESPECIALIZADOS* y *APRENDICES* puede controlarse con una aserción de la siguiente forma:

```
CREATE ASSERTION Controlar-exclusividad-generalización
    Check ((Tipo='especializado' AND EXIST ESPECIALIZADOS
AND NOT EXIST APRENDICES) OR
        (Tipo='aprendiz' AND EXIST APRENDICES
        AND NOT EXIST ESPECIALIZADOS))
```

2.2.7 Transformación de Interrelaciones de grado superior a dos

Hasta el momento solo se han tratado interrelaciones binarias. Si en el esquema E/R se encuentran interrelaciones de grado superior (ternarias, cuaternarias, etc.) se requiere hacer un estudio exhaustivo de las cardinalidades mínimas y máximas para llevar a cabo la transformación. Se mostrarán las reglas de transformación para el caso de las interrelaciones ternarias aunque podría extenderse a las de grado superior a 3.

El caso general de transformación de interrelaciones ternarias N:M:P establece que la interrelación da lugar a una relación cuya clave primaria es la concatenación de los Identificadores Principales de las entidades que relaciona. En el caso de que las cardinalidades sean (1,N), (1,N) y (1,N) no es necesario ningún mecanismo adicional para no perder semántica en la transformación al relacional. La interrelación ternaria **Trabaja** de la Figura 2.2 da lugar a una relación *TRABAJAN* cuya clave primaria es la concatenación de las claves primarias de *AREAS REPARACIONES*, *OPERARIOS* y *TAREAS* y que son *num_area*, *DNI*, *cod_reparacion* y *num_tarea*. Además, esta relación contiene también el campo *num_horas*.

Sin embargo, es necesario estudiar las cardinalidades mínimas y máximas en los tres lados de la interrelación para realizar la transformación al relacional sin perder semántica, por ejemplo, en el caso de que alguna cardinalidad mínima sea 0 o en el caso de que algún lado de la interrelación tenga cardinalidad (1,1). Estas distintas posibilidades se estudiarán en la resolución de los problemas propuestos en este libro.

2.2.8 Transformación de Interrelaciones Exclusivas

En el caso de interrelaciones exclusivas, éstas se transforman como se ha

visto en este mismo apartado, pero es necesario añadir un *CHECK* que compruebe que si un ejemplar de la entidad participa ya en una ocurrencia de una interrelación, entonces no puede participar en ninguna ocurrencia de la otra interrelación.

2.2.9 Algunos aspectos sobre la pérdida de semántica en la transformación al Modelo Relacional

Existen algunas restricciones que es necesario controlar con mecanismos externos al modelo relacional. Aquí es necesario establecer la diferencia entre el modelo relacional y los SGBD comerciales que no implementan el modelo relacional completo. Por ello, como se ha mencionado en los apartados anteriores dedicados a la transformación al modelo relacional, aunque aquí se hable de *CHECKS* y aserciones como mecanismos del modelo relacional para implementar ciertas restricciones, estos mecanismos no siempre están disponibles en los sistemas gestores de BD, lo que implica que hay que recurrir a otros medios para recoger esas restricciones (por ejemplo, disparadores presentes en la BD, procedimientos almacenados, aplicaciones externas, etc.). Se dará aquí enumeración de las restricciones de los esquemas E/R que es necesario contemplar en la transformación al modelo relacional mediante *checks*, aserciones o disparadores:

- Cardinalidades mínimas de 1 en interrelaciones N:M y 1:N (excluyendo aquellas que se controlan con la restricción *NOT NULL* cuando se realiza propagación de clave).
- Cardinalidades máximas conocidas en interrelaciones binarias N:M y 1:N e interrelaciones ternarias.
- Exclusividad en las generalizaciones.
- Inserción y borrado en las generalizaciones.
- Atributos derivados.
- Exclusividad entre interrelaciones.
- Atributos multivaluados obligatorios.

Por último, puede mejorarse la calidad del diseño resultante introduciendo otras restricciones que no figuraban en el enunciado original pero que se consideran adecuadas o convenientes (por ejemplo, restricciones que implican

operadores de comparación de fechas).

2.3 DINÁMICA DEL MODELO RELACIONAL: ÁLGEBRA RELACIONAL

En los apartados anteriores se ha estudiado la parte estática del modelo relacional. Repasaremos ahora brevemente la dinámica del mismo desde el punto de vista del álgebra relacional. No estudiaremos aquí la sintaxis del lenguaje SQL3, pues en los problemas propuestos en este capítulo se mostrarán sentencias para la creación y manipulación de una BD en este lenguaje. Remitimos al lector a la bibliografía recomendada para profundizar en estos aspectos.

Los operadores del álgebra relacional se aplican a relaciones dando como resultado nuevas relaciones. Originariamente Codd definió ocho operadores en dos grupos:

1. Las operaciones algebráicas tradicionales de conjuntos: unión, intersección, diferencia y producto cartesiano.
2. Las operaciones relacionales especiales: restricción, proyección, combinación y división.

Por otro lado, la intersección, combinación y división se pueden derivar de otros operadores, y por ello se denominarán operaciones derivadas (frente al resto que serán primitivas). Una operación del álgebra relacional consiste en:

O - Operador del álgebra relacional (unión, combinación, etc.).

r - Relación(es)¹³ la(s) que se aplica el operador O.

r' - Relación resultado de la operación.

tal que: $O(r) = r'$

La Figura 2.3 muestra los operadores primitivos y derivados del álgebra relacional y una representación gráfica de estos operadores.

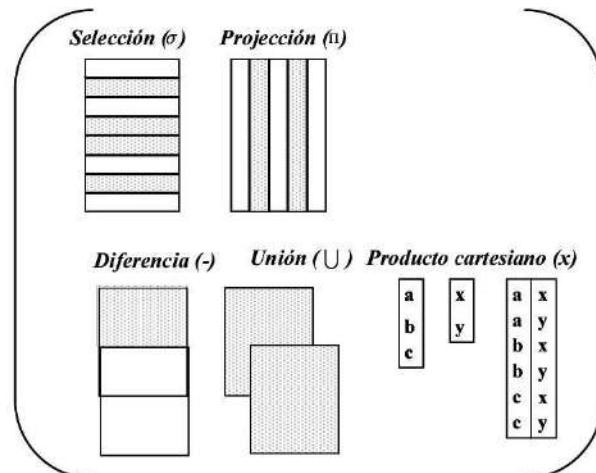
- **Restricción o selección (σ):** extrae las tuplas de una relación dada que satisfagan una condición especificada. Por ejemplo, seleccionar los empleados cuya $f_nac < 01/01/1972$ de la tabla *EMPLEADOS*:

$$\sigma_{f-nac < 01/01/1970}(EMPLEADOS)$$

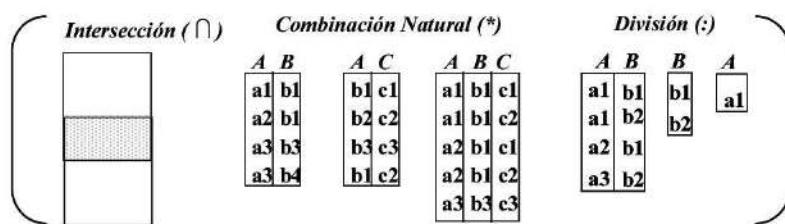
- **Proyección (Π):** extrae los atributos especificados de una relación dada eliminando las tuplas duplicadas. Por ejemplo, seleccionar los nombres de los empleados de la relación *EMPLEADOS*:

$$\pi_{\text{nombre}}(EMPLEADOS)$$

- **Unión (U):** construye una relación con todas las tuplas que aparezcan en cualquiera de las dos relaciones especificadas. Para usar este operador es necesario que las dos relaciones especificadas tengan los mismos atributos definidos sobre los mismos dominios.
- **Diferencia (-):** construye una relación con las tuplas de la primera relación que no aparecen en la segunda. Para usar este operador es necesario que las dos relaciones especificadas tengan los mismos atributos definidos sobre los mismos dominios.



Operadores primitivos



Operadores derivados

Figura 2.3. Operadores del álgebra relacional principales

- **Producto Cartesiano (x):** a partir de dos relaciones especificadas, construye una relación concatenando cada tupla de la primera con cada una de las tuplas de la segunda. La relación resultante está definida sobre la unión de los atributos de las dos relaciones especificadas.
- **Combinación (θ):** a partir de las relaciones especificadas, construye una relación concatenando cada tupla de la primera relación con cada una de las tuplas de la segunda, siempre que ambas tuplas satisfagan la condición dada. Si la condición es la igualdad y en la relación resultante se elimina el atributo común, se denomina combinación natural. En la combinación externa se incluyen las tuplas de una relación que no están relacionadas con otra relación, concatenando con nulos. La combinación (θ) es un producto cartesiano seguido de una restricción. La combinación natural (*) es un producto cartesiano seguido de restricción por igualdad y proyección. Un ejemplo se muestra en la Figura 2.4.

Combinación Natural ($R1 * R2$)

R1	A	B	*	R2	A	C	=	R	A	B	C
	a	u			a	x			a	u	x
	b	v			a	y			a	u	y
	c	w			b	z			b	v	z

Figura 2.4. Combinación natural

- **Intersección (\cap):** construye una relación con aquellas tuplas que aparezcan en las dos relaciones especificadas. Para usar este operador es necesario que las dos relaciones especificadas tengan los mismos atributos definidos sobre los mismos dominios. La intersección se puede definir en función de la unión y la diferencia.
- **División (:):** construye una relación con los valores de un atributo de la primera relación (dividendo) que concuerda con todos los valores de los atributos de la segunda relación (divisor). Se puede definir en función de la proyección, del producto cartesiano y de la diferencia.
- **Agrupación (GROUP BY):** agrupa tuplas por valores comunes de ciertos atributos y aplica una función de agregación (frecuencia, suma, media...) a cada subconjunto. La Figura 2.5 muestra un ejemplo, en donde los atributos por los que se agrupa son *sexo*, *ec* dando lugar a cuatro grupos y de estos grupos se selecciona la media de la *edad*.

PERS

DNI	NOMBRE	SEXO	EDAD	EC
251.336	JUAN	V	24	S
676.876	PEDRO	V	14	S
789.908	ELENA	M	56	C
678.987	MARÍA	M	38	C
753.098	ELISA	M	42	V
458.179	ANA	M	50	V
261.126	LUIS	V	30	C
35.245	JULIAN	V	26	C

RES = IT SEXO, EC, MEDIA(EDAD) (GROUP BY SEXO, EC (PERS))

RES

SEXO	EC	MEDIA (EDAD)
V	S	19
V	C	28
M	C	47
M	V	46

Figura 2.5. Ejemplo de Agrupación

2.4 CÓMO SE ESTRUCTURAN LOS PROBLEMAS

Los problemas de este capítulo se estructuran en dos grandes bloques: por un lado, aquellos cuyo objetivo es diseñar un esquema relacional a partir de una serie de supuestos semánticos (problemas 2.1 hasta 2.10) y, por otro lado, los problemas cuyo objetivo es obtener un esquema relacional a partir de un esquema E/R (para los que se utilizarán las reglas de transformación vistas en el apartado *Reglas de transformación de un esquema E/R a un esquema relacional*).

Dentro del primer bloque de problemas se han seleccionado un conjunto de problemas resueltos a distintos niveles de detalle. Los más detallados en su resolución son los cinco primeros. Los cinco últimos son enunciados con solución y algunos breves comentarios que hemos creído importantes para un mejor entendimiento de la solución propuesta.

Con la experiencia adquirida, sobre todo en los dos primeros problemas que muestran de una manera metódica cómo resolver problemas de diseño relacional, se presentarán una serie de problemas extraídos de exámenes o ejercicios propuestos durante estos años de labor docente del grupo LABDA. Hemos seleccionado dos tipos de problemas. El primero de ellos es una sucesión de supuestos que se deben abstraer en un grafo relacional. En ellos se comentarán las partes más complicadas del diseño, se advertirá sobre otras formas de diseñarlas y por supuesto, se hará hincapié en los supuestos que se han añadido y aquéllos que no se han podido reflejar en el grafo. El otro tipo de problemas parte de un subgrafo al que se va modificando según se especifican nuevas restricciones. La mayoría de los problemas contienen también un apartado para resolver consultas tanto en álgebra relacional como en SQL3. En la resolución de estas consultas se ha intentado que la expresión en SQL3 fuera una traducción casi directa de la escrita en álgebra relacional. La eficiencia de estas consultas dependerá fuertemente del SGBD que se utilice en la fase de implementación y, por ello, la consulta SQL3 dada como solución puede que no sea la más eficiente para el gestor de trabajo.

Para el segundo tipo de problemas se aplicarán las reglas de transformación de esquemas E/R a esquemas relacionales explicadas anteriormente y que se enumeran a continuación junto con las restricciones semánticas que hay que contemplar en un esquema relacional:

1. Transformación de entidades, atributos y dominios.
2. Transformación de interrelaciones N:M.
3. Transformación de interrelaciones 1:N y de Dependencias en Existencia y en Identificación.
4. Transformación de Generalizaciones.
5. Transformación de Interrelaciones de grado superior.
6. Transformación de Interrelaciones exclusivas.
7. Incorporar las opciones de borrado y modificación en las claves ajena
8. Incorporar las restricciones de obligatoriedad y unicidad.
9. Incorporar las restricciones de usuario que no podían recogerse en el esquema E/R y que ahora es necesario incluir en el esquema relacional (por ejemplo, mediante restricciones de verificación).

PROBLEMA 2.1: ALOJAMIENTOS RURALES

La Comunidad de Madrid desea guardar información sobre los alojamientos rurales que existen en dicha comunidad. Para ello decide crear una base de datos que recoja las siguientes consideraciones:

Un alojamiento rural se identifica por un nombre (“Villa Aurora”, “Las Rosas”, etc.) tiene una dirección, un teléfono y una persona de contacto que pertenece al personal del alojamiento. En cada alojamiento trabajan una serie de personas que se identifican por un código de personal. Se requiere conocer el nombre completo, la dirección y el NIF. Aunque en un alojamiento trabajen varias personas, una persona solo puede trabajar en un alojamiento. Los alojamientos se alquilan por habitaciones y se desea conocer cuántas habitaciones componen el alojamiento, de qué tipo es cada una de estas habitaciones (individuales, dobles, triples), si poseen cuarto de baño y el precio.

En algunos de estos alojamientos se realizan actividades multiaventura organizadas para huéspedes (senderismo, bicicleta de montaña, etc.). Estas actividades se identifican por un código. Es de interés saber el nombre de la actividad, la descripción y el nivel de dificultad de dicha actividad (de 1 a 10). Estas actividades se realizan un día a la semana, por ejemplo: en la casa “El pazo de Paco” se practica senderismo los jueves y se desea guardar esta información. Pero puede haber algún día en el que no se practique ninguna actividad.

Se pide:

1. Diseñar el esquema relacional. Indicar en él las claves primarias, claves alternativas y las claves ajenas.
2. Escribir las siguientes consultas en SQL y en Álgebra Relacional:
 - Nombre y descripción de las actividades que se realizan en el alojamiento denominado “La Huerta”.
 - Nombre de los alojamientos que tienen habitaciones dobles y realizan actividades de senderismo (el nombre de la actividad es senderismo).
 - Nombres de las personas de contacto que trabajan en alojamientos con más de 10 habitaciones.

- Número medio de actividades que ofertan los alojamientos rurales de la Comunidad de Madrid.

Discusión del enunciado

Diseñar el esquema relacional. Indicar en él las claves primarias, claves alternativas y las claves ajenas. De estas últimas, especificar los modos de borrado y modificación.

IDENTIFICACIÓN DE LAS RELACIONES BÁSICAS

Un alojamiento rural se identifica por un nombre (“Villa Aurora”, “Las Rosas”, etc.) tiene una dirección, un teléfono y una persona de contacto.

Debemos construir una primera relación **ALOJAMIENTOS** donde se almacenen todos los alojamientos rurales de la Comunidad de Madrid. Su clave primaria será el nombre del alojamiento (*nombre_aloja*). La persona de contacto, como se verá en el párrafo siguiente de las especificaciones, forma parte del personal del alojamiento, y como un empleado no puede trabajar en varios alojamientos, este atributo será una clave alternativa (y cuando se estudien los tipos de vínculos se verá que además es ajena).

ALOJAMIENTOS (nombre_aloja, direccion, telefono, contacto)

En cada alojamiento trabajan una serie de personas que se identifican por un código de personal. Se requiere conocer el nombre completo, la dirección y el NIF.

Consideramos una nueva relación **PERSONAL** cuya clave primaria es el código de personal (*codigo_p*) y cuya clave alternativa es el *NIF*.

PERSONAL (codigo_p, nombre_p, direccion, NIF)

Los alojamientos se alquilan por habitaciones y se desea conocer cuántas habitaciones componen el alojamiento, de qué tipo es cada una de estas habitaciones (individuales, dobles, triples), si poseen cuarto de baño y el precio.

Debemos considerar una relación **HABITACIONES**. El número de habitación (*n-habita*) no identifica cada una de las tuplas de esta relación, habrá

habitaciones con el mismo número en distintos alojamientos. Sin embargo, si se incluye el nombre del alojamiento, formando una clave primaria compuesta, se consigue la unicidad de cada tupla. Esto equivale a decir que el número de habitación no tiene sentido si no se le asocia un alojamiento.

Se desea recoger el número de habitaciones de que dispone cada alojamiento, por lo que debemos introducir un nuevo atributo *nº-hab* en la relación ALOJAMIENTOS, como se muestra en la figura.

HABITACIONES (alojamiento, n-habita, tipo, baño, precio)

ALOJAMIENTOS (nombre_aloja, direccion, telefono, contacto, *nº-hab*)

En algunos de estos alojamientos se realizan actividades multiaventura organizadas para huéspedes (senderismo, bicicleta de montaña, etc...). Estas actividades se identifican por un código.

Es de interés saber el nombre de la actividad, la descripción y el nivel de dificultad de dicha actividad (1_10).

De este último párrafo se deduce la necesidad de una nueva relación ACTIVIDADES, cuya clave primaria será el código de actividad (*codigo_act*).

ACTIVIDADES (*codigo_act*, nombre_act, descripcion, nivel)

ANÁLISIS DE LOS TIPOS DE VÍNCULOS

A través del texto podemos identificar en el primer párrafo de las especificaciones que existe un tipo de vínculo uno a varios entre las relaciones ALOJAMIENTOS y PERSONAL.

Un alojamiento rural se identifica por un nombre (“Villa Aurora”, “Las Rosas”, etc.). y una persona de contacto que pertenece al personal del alojamiento.... Aunque en un alojamiento trabajen varias personas, una persona solo puede trabajar en un alojamiento.

En un alojamiento pueden trabajar una serie de personas, pero una misma persona solo puede trabajar en un alojamiento, por tanto debemos introducir un atributo *nombre_aloja* en la relación PERSONAL, que será clave ajena de esta relación y referenciará a ALOJAMIENTOS. De esta manera queda totalmente reflejado el vínculo que existe entre estas tablas.

Además todo alojamiento tiene una persona de contacto que pertenece al personal del alojamiento, luego una de las posibilidades puede ser que el atributo *contacto* sea clave ajena que refencie a PERSONAL.



El tipo de vínculo entre alojamiento y persona de contacto es del tipo uno a uno ya que un alojamiento solo puede tener una persona de contacto y una persona de contacto solo lo será de un alojamiento. Existen dos formas de representar este tipo de vínculo en el modelo relacional: como un caso particular de un tipo de vínculo uno a varios o como un caso particular de un vínculo varios a varios. Se elegirá una solución u otra siempre teniendo en cuenta que el diseño debe evitar los valores nulos, y debe ser intuitivo y lo más sencillo posible.

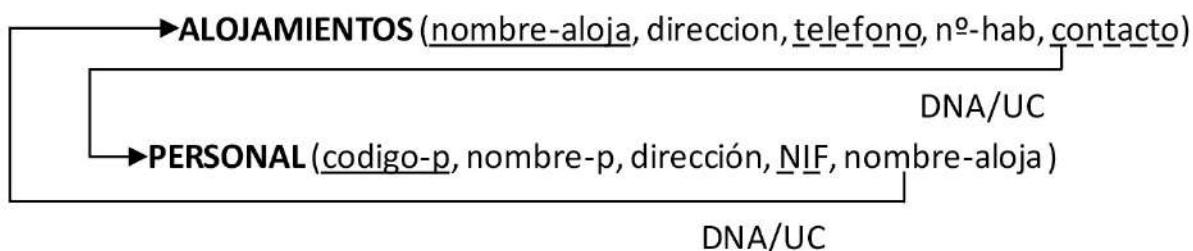


Figura 2.6. Representación de un vínculo uno a uno

Si lo representamos con un tipo de vínculo uno a uno, la clave podría propagarse a cualquiera de las relaciones. En la Figura 2.6 muestra el caso en que la clave propagada es la persona de contacto a la relación *ALOJAMIENTOS*. La otra posibilidad, no representada, sería propagar el nombre del alojamiento a *PERSONAL*, para indicar qué personas son de contacto y de qué alojamiento. En este caso, se tendrían dos claves ajenas que referenciarían a *ALOJAMIENTOS* que necesariamente tendrían el mismo valor, o la segunda tendría valor nulo (puesto que no todo personal es de contacto).



Se podría pensar en sustituir la segunda clave ajena por un atributo booleano, como por ejemplo, *es-persona-de-contacto*. Desafortunadamente con esta solución se sacrifica la semántica de que exista siempre una y solo una persona de contacto. Por otra parte, si la persona de contacto de un alojamiento trabaja en ese alojamiento, la representación elegida en la Figura 2.6 no asegura el cumplimiento de esta restricción, que deberá controlarse mediante una aserción.

Por lo tanto, la mejor solución (dentro del tipo de vínculo uno o varios) es la presentada en la Figura 2.6. Entre las relaciones *ALOJAMIENTOS* y *PERSONAL*, existe un ciclo referencial que impide la definición de las claves ajenas en el mismo momento que se crean las tablas y la ejecución de operaciones de actualización. Dicho de otro modo, la restricción de clave ajena en el atributo *contacto* de la relación *ALOJAMIENTOS* obliga a que, a la hora de introducir un nuevo alojamiento, se haya incluido previamente la persona de contacto correspondiente en la relación *PERSONAL*. Por otro lado, dar de alta a un nuevo empleado en *PERSONAL*, dado que existe una restricción de clave ajena en el atributo *nombre-aloha*, implica haber introducido previamente el alojamiento correspondiente. Parece que se ha llegado a una situación contradictoria. Este problema se puede solucionar creando las tablas sin las claves ajenas, modificando posteriormente la definición de la relación, como se muestra en el *script* de SQL al final de la resolución del ejercicio. También habrá que deshabilitar algunas de las claves cuando se realicen operaciones de actualización. Además, con esta solución, *contacto* produce una redundancia puesto que la persona de contacto debe trabajar en el mismo alojamiento donde desempeña este puesto, por lo que aparece como valor en la clave ajena de la relación *PERSONAL*. Esta restricción deberá ser controlada a través de una aserción.

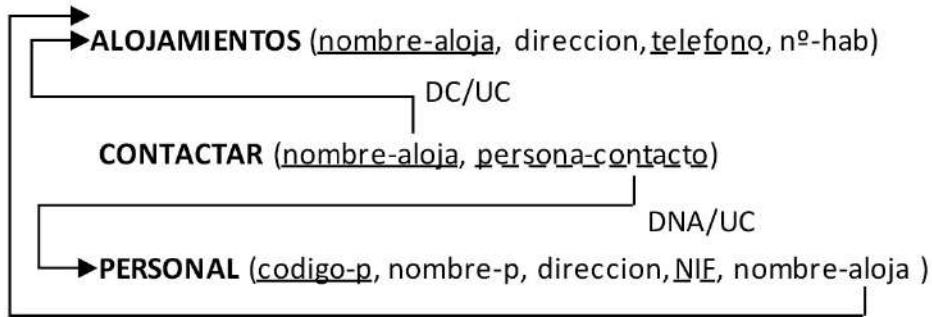


Figura 2.7. Representación de un vínculo varios a varios

La segunda posibilidad es crear una nueva relación, tratando este vínculo como si fuera de tipo varios a varios. La clave primaria de esta relación en uno de los dos atributos y otro debe ser clave alternativa, véase la Figura 2.7. Con esta solución evitamos el ciclo referencial que aparece en el otro caso pero como contrapartida se puede dar el caso de alojamientos en los que no aparezca una persona de contacto, a menos que se incluya en la fase de implementación un disparador para controlar que todo alojamiento tenga siempre una persona de contacto y además, que esta persona pertenezca al personal del alojamiento.

Elegimos la primera opción por abarcar distintas opciones en el diseño y para que el lector sepa cómo afrontar este tipo de ciclos en la fase de implementación.

Los supuestos semánticos que se han incluido en este subesquema se corresponden con las restricciones de integridad para las claves ajenas. Como se puede observar en la Figura 2.7, los borrados son no acción, siguiendo la política de borrados controlados y las modificaciones en cascada.

Con la solución adoptada, según se ha explicado anteriormente, se debe controlar en la fase de implementación que toda persona de contacto de un alojamiento debe pertenecer al personal del mismo (esta restricción no puede quedar reflejada en el grafo relacional).

En el siguiente extracto del texto se detecta un tipo de vínculo uno a varios.

Los alojamientos se alquilan por habitaciones y se desea conocer cuántas habitaciones componen el alojamiento....

Un alojamiento, por lo tanto, se compone de una a varias habitaciones y una habitación solo es de un hotel. Como ya se explicó anteriormente, se necesita el alojamiento, no solo para identificar cada tupla de *HABITACIONES* sino también

para expresar que esta relación no tiene sentido si no está asociada con un alojamiento. El atributo *nombre_aloja*, además de formar parte de la clave primaria, será clave ajena que referencia a la tabla *ALOJAMIENTOS*.

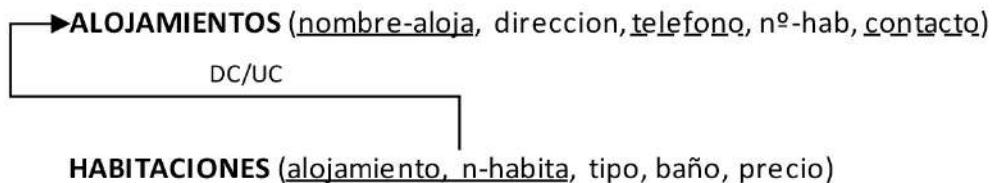


Figura 2.8. Restricción de clave ajena entre habitaciones y alojamientos

Como se observa en la Figura 2.8, la opción de borrado que se ha considerado es en cascada, puesto que en este caso se quiere recoger la existencia de cierta dependencia de la relación *HABITACIONES* con *ALOJAMIENTOS*. Si desaparece un alojamiento, no tendrá sentido mantener información acerca de sus habitaciones. La opción de modificación también es en cascada.

Se debe tener en cuenta que no es posible reflejar las siguientes restricciones. Un alojamiento debe tener al menos una habitación. Aunque esta restricción no aparezca en la especificación de requisitos, el conocimiento del dominio nos hace añadir este supuesto. Durante la fase de implementación se tendrá que añadir un disparador para la inserción que controle esta restricción. Además, como se desea conocer el número de habitaciones de las que consta cada alojamiento, deberíamos tener un disparador que, cada vez que se dé de alta o se elimine una habitación, se incremente o decremente en una unidad el valor del atributo *nº_hab*, puesto que este atributo es derivado (lo que incluye cierta redundancia en el grafo).



El conjunto de especificaciones dadas en el enunciado de este ejercicio no indica que sea importante la información acerca de los alquileres de estos alojamientos por lo que no se ha incluido.

Por último, en el siguiente párrafo aparece otro tipo de vínculo o asociación entre las relaciones *ACTIVIDADES* y *ALOJAMIENTOS*.

En algunos de estos alojamientos se realizan actividades multiaventura organizadas para huéspedes (senderismo, bicicleta de montaña, etc.). Estas actividades se realizan un día a la semana, por ejemplo: en la casa “Villa Aurora” se practica senderismo los jueves y se desea guardar esta

información. Pero puede haber algún día en el que no se practique ninguna actividad.

Cada alojamiento puede proporcionar ciertas actividades multiaventura y, además, una misma actividad puede ser propuesta por varios alojamientos. Como vemos, es un tipo de vínculo varios a varios y, por tanto, tenemos que crear una nueva relación *REALIZA_ACTIVIDAD*, cuya clave primaria estará compuesta por las claves primarias de las relaciones *ALOJAMIENTOS* y *ACTIVIDADES*. Cada una de ellas será a su vez clave ajena que referenciará a la relación de la cual provienen.

Debemos introducir el atributo *dia_semana* en la relación *REALIZA_ACTIVIDAD*, pues en el enunciado se dice que se desea guardar la información relativa al día de la semana en que se realiza cada actividad.

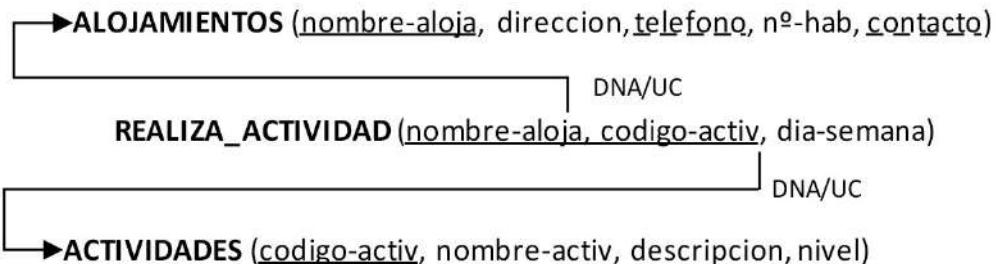


Figura 2.9. Representación de la interrelación entre alojamientos y actividades

Si se opta por el diseño de la relación *REALIZA_ACTIVIDAD* según el subesquema presentado en la Figura 2.9, lo que se refleja es que no puede darse la misma actividad en un alojamiento en varios días.

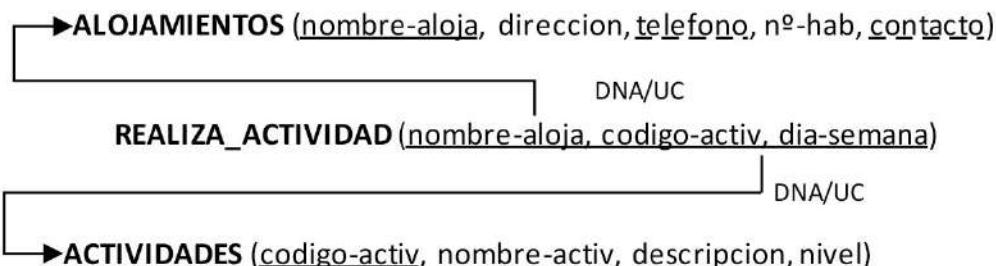


Figura 2.10. Representación de la restricción temporal

Si la relación *REALIZA_ACTIVIDAD* posee como clave primaria todos los atributos que la componen (Figura 2.10) entonces lo único que no está permitido es que una actividad se practique varias veces a lo largo de un mismo día en un

alojamiento. Como en las especificaciones del dominio no aparece reflejado ninguno de estos supuestos, se tomará el menos restrictivo (Figura 2.10).

Al no aparecer en las especificaciones nada acerca de las restricciones de integridad a tener en cuenta para cada una de las claves ajenas definidas en la relación *REALIZA_ACTIVIDAD*, se supone el borrado sin acción y la modificación en cascada para ambas. De esta manera, se controlan los borrados y las modificaciones.

GRAFO RELACIONAL

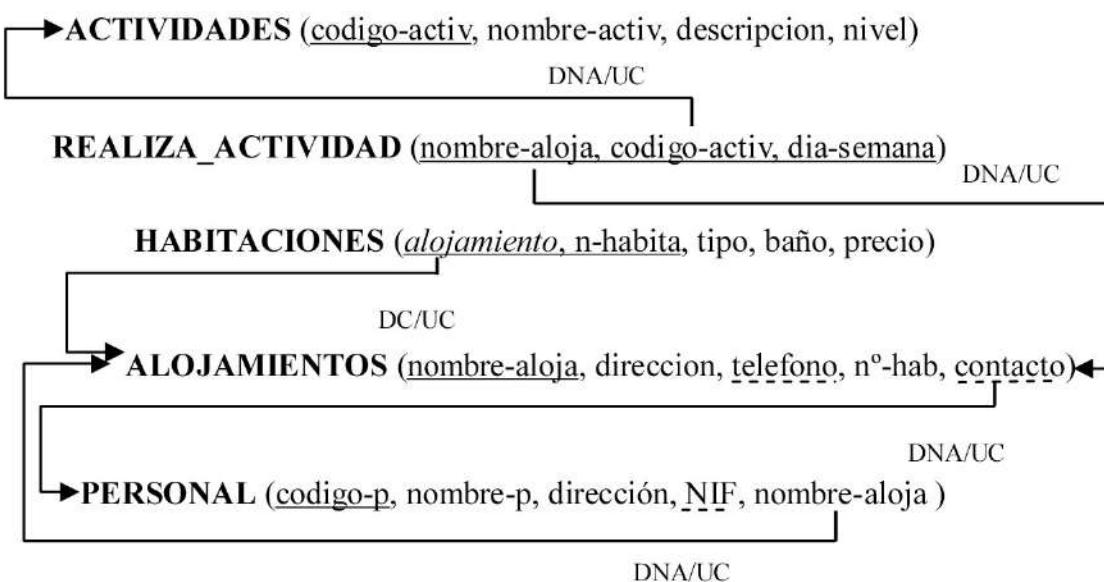


Figura 2.11. Grafo relacional completo para el problema de los alojamientos rurales

SCRIPT EN SQL3

```
CREATE DOMAIN Nombre-Valido CHAR(30);
CREATE DOMAIN Tipo-Codigo CHAR(3);
CREATE DOMAIN Tipo-Baño CHAR(2)
    CHECK (VALUE IN ('SI', 'NO'));
CREATE DOMAIN Tipo-Habitacion CHAR(10)
    CHECK (VALUE IN ('individual', 'doble', 'triple'));
CREATE DOMAIN Tipo-Nivel-Dificultad INTEGER BETWEEN 1 AND 10;

CREATE TABLE ALOJAMIENTOS (
    nombre-aloja Nombre-Valido,
    direccion     CHAR(25) NOT NULL,
    telefono      CHAR (9) NOT NULL,
    contacto      Tipo-Codigo NOT NULL,
    n°_hab        INTEGER NOT NULL,
```

```

CONSTRAINT pk-alojamientos PRIMARY KEY (nombre-aloha)
CONSTRAINT uk-alojamientos1 UNIQUE (contacto)
CONSTRAINT uk-alojamientos2 UNIQUE (telefono)) ;

CREATE TABLE PERSONAL (
    codigo-p      Tipo-Codigo,
    nombre-p      Nombre-Valido NOT NULL,
    direccion     CHAR(25) NOT NULL,
    NIF           CHAR(10) NOT NULL,
    nombre-aloha Nombre-Valido NOT NULL,
    CONSTRAINT pk-personal PRIMARY KEY (codigo-p),
    CONSTRAINT uk-personal UNIQUE (NIF),
    CONSTRAINT fk-personal-alojamientos FOREIGN KEY
        (nombre_aloha) REFERENCES ALOJAMIENTOS ON UPDATE
        CASCADE ) ;

ALTER TABLE ALOJAMIENTOS
    ADD CONSTRAINT fk-alojamientos-personal FOREIGN KEY
        (contacto) REFERENCES PERSONAL
        ON UPDATE CASCADE;

CREATE TABLE HABITACIONES (
    alojamiento   Nombre-Valido,
    n-habita      INTEGER(3),
    tipo          Tipo-Habitacion NOT NULL,
    baño          Tipo-Baño NOT NULL,
    precio        INTEGER NOT NULL,
    CONSTRAINT pk-habitaciones PRIMARY KEY (alojamiento, n-
    habita),
    CONSTRAINT fk-habitaciones-alojamientos FOREIGN KEY
        (nombre_aloha) REFERENCES ALOJAMIENTOS
        ON UPDATE CASCADE ON DELETE CASCADE);

CREATE TABLE ACTIVIDADES (
    codigo-activ   Tipo-Codigo,
    nombre-activ   Nombre-Valido NOT NULL,
    descripcion    CHAR(50) NOT NULL,
    nivel          Tipo-Nivel-Dificultad NOT NULL,
    CONSTRAINT pk-actividades PRIMARY KEY (codigo-activ));

CREATE TABLE REALIZA_ACTIVIDAD (
    codigo-act     Tipo-Codigo,
    Nombre-aloha  Nombre-Valido,
    Dia-semana    CHAR(10) NOT NULL,

```

```
CONSTRAINT pk-realiza-actividad PRIMARY KEY (codigo-act,  
        nombre-aloja, dia-semana),  
CONSTRAINT fk-realiza-actividad-actividades FOREIGN KEY  
(codigo_act) REFERENCES ACTIVIDADES ON UPDATE CASCADE  
CONSTRAINT fk-realiza-actividad-alojamientos FOREIGN KEY  
(nombre_aloja) REFERENCES ALOJAMIENTOS ON UPDATE CASCADE  
);
```



Aquellas restricciones de integridad que no aparecen especificadas para la clave ajena se deben a que la opción por defecto tanto para el borrado como la modificación es sin acción, por lo que no es necesario definirlas.



Es importante destacar la definición de los dominios incluida al comienzo del script. Estos dominios se han identificado durante la fase de elaboración del grafo relacional.

CONSULTAS

Nombre y descripción de las actividades que se realizan en el alojamiento denominado “La Huerta”.

ÁLGEBRA RELACIONAL

$$\Pi_{\text{nombre-activ,descripcion}} [ACTIVIDADES * \Pi_{\text{codigo-activ}} (\sigma_{\text{nombbre='La Huerta}} (\text{REALIZA_ACTIVIDAD}))]$$

SQL

```
SELECT A.nombre-activ, A.descripcion  
FROM ACTIVIDAD A, REALIZA_ACTIVIDAD B  
WHERE nombre-aloja = "La Huerta" AND  
A.codigo-activ = B.codigo-activ;
```

A continuación se muestra la misma consulta en SQL empleando el operador *JOIN*.

```
SELECT nombre-activ, descripcion  
FROM ACTIVIDAD JOIN REALIZA_ACTIVIDAD WHERE nombre-aloja =  
"La Huerta");
```



La sintaxis del operador *JOIN* en SQL3 permite omitir la condición de combinación cuando existen atributos con el mismo nombre en las relaciones que intervienen en la operación.

Nombre de los alojamientos que tienen habitaciones dobles y realizan actividades de senderismo.

ÁLGEBRA RELACIONAL

$$\begin{aligned} & \Pi_{\text{número-aloja}} \\ & \left[\sigma_{\text{tipo}='doble'}(\text{HABITACIONES})^* \right]_{\text{alojamiento}=\text{número-aloja}} \\ & \left[\Pi_{\text{número-aloja}} \left[\text{REALIZA_ACTIVIDAD}^* \right. \right. \\ & \left. \left. \left(\Pi_{\text{codigo-activ}} \left(\sigma_{\text{número-activ}='Senderismo'}(\text{ACTIVIDADES}) \right) \right) \right] \right] \end{aligned}$$

Utilizando un operador de conjuntos, como es la intersección, se puede redefinir la consulta como se muestra a continuación:

$$\begin{aligned} & \Pi_{\text{número-aloja}} \left(\sigma_{\text{tipo}='doble'}(\text{HABITACIONES}) \right) \\ & \cap \Pi_{\text{número-aloja}} \left[\text{REALIZA_ACTIVIDAD}^* \right. \\ & \left. \left(\Pi_{\text{codigo-activ}} \left(\sigma_{\text{número-activ}='Senderismo'}(\text{ACTIVIDADES}) \right) \right) \right] \end{aligned}$$

SQL

```
SELECT DISTINCT B.nombre-aloja  
  
FROM ACTIVIDAD A, REALIZA_ACTIVIDAD B, HABITACIONES C  
  
WHERE A.nombre-activ = "Senderismo" AND C.tipo = "doble" AND  
(B.codigo-activ = A.codigo-activ AND B.nombre-aloja =  
C.nombre-aloja);
```

Otra forma de abordar la consulta SQL siguiendo la filosofía de la consulta con el operador de conjuntos Intersección en Álgebra Relacional, se presenta a continuación.

```
SELECT nombre-aloja  
  
FROM HABITACIONES WHERE tipo='doble'  
  
INTERSECT  
  
SELECT nombre-aloja  
  
FROM REALIZA_ACTIVIDAD JOIN (SELECT nombre-aloja FROM  
ACTIVIDAD WHERE nombre-activ = 'Senderismo');
```

Nombres de las personas de contacto que trabajan en alojamientos con más de 10 habitaciones.

ÁLGEBRA RELACIONAL

$$\prod_{nombre - p} [(\sigma_{n^o-hab > 10}(ALOJAMIENTOS)) *_{nombre - aloja} PERSONAL]$$

SQL

```
SELECT nombre-p  
FROM ALOJAMIENTOS A JOIN PERSONAL P  
ON A.nombre-aloja = P.nombre-aloja  
WHERE A.nº-hab > 10;
```

Siguiendo estrictamente la sintaxis de la consulta expresada en álgebra relacional se obtendría la siguiente expresión SQL.

```
SELECT nombre-p FROM PERSONAL P, (SELECT * FROM ALOJAMIENTOS  
A WHERE A.nº-hab > 10) Q  
WHERE Q.nombre-aloja = P.nombre-aloja;
```

Número medio de actividades que ofertan los alojamientos rurales de la Comunidad de Madrid.

ÁLGEBRA RELACIONAL

$$\prod_{AVG(num-activ)} \left[\prod_{COUNT(*) AS num-activ} [GROUP BY_{nombre-aloja} (REALIZA_ACTIVIDAD)] \right]$$

SQL

```
SELECT AVG(num-activ)
FROM (SELECT COUNT(*) AS num-activ
      FROM REALIZA_ACTIVIDAD
     GROUP BY nombre-aloja);
```

Como puede verse, en primer lugar se obtiene el número de actividades diferentes que se ofertan en cada alojamiento para después proceder a calcular la media. En este caso es imprescindible hacer uso de subconsultas para obtener el cálculo pedido.

PROBLEMA 2.2: LA TIENDA DE REGALOS

El esquema relacional mostrado en la Figura 2.12 contiene parte de los elementos necesarios para modelar la semántica de una tienda de regalos de ámbito nacional. El cliente es la persona que quiere realizar un regalo a otra persona que se denomina receptor. La tienda tiene los regalos agrupados en categorías y el cliente deberá indicar cuáles de estos regalos son los preferidos del receptor.

CLIENTES (cod_cliente, DNI, nombre_completo, direccion, telefono)

RECEPTORES (nombre_completo, direccion, codigo postal, provincia, telefono)

→ CATEGORIAS (id_categoria, categoria, descripcion)

REGALOS (categoria, nº_referencia, unidades, precio/uni)

Figura 2.12. Grafo relacional inicial para el problema de la tienda de regalos

A continuación se presenta una lista de supuestos semánticos. Se pide indicar si ya están contemplados en el esquema, si se pueden añadir al mismo y cómo, o si no es posible reflejarlos.

- a) La descripción de la categoría por la que se agrupan los regalos no es obligatoria.
- b) Cada cliente tiene un DNI que lo identifica.
- c) Se puede eliminar una categoría pero no se eliminarán los regalos de esa categoría.
- d) Si el valor correspondiente a *id_categoria* es modificado en la tabla *REGALOS* esa modificación se propagará a categoría.
- e) Es necesario almacenar en el esquema de la base de datos las preferencias que el cliente señala acerca del receptor del regalo. Las preferencias indican qué regalos son los favoritos de la persona a la que se quiere regalar.
- f) La compra siempre será de un regalo y queremos almacenar el día en

que se efectúo la misma y la dedicatoria (si la tiene). Un regalo es comprado por un cliente para un receptor y no puede ocurrir que un mismo regalo sea enviado a la misma persona el mismo día.

- g) Cada vez que se realiza una compra de un determinado regalo se debe disminuir el número de unidades disponibles de ese regalo en la tienda (atributo *unidades* en la relación *REGALOS*).
- h) El regalo comprado debe aparecer en las preferencias señaladas por el cliente para la persona que va a recibir el regalo elegido, es decir, no se podrá seleccionar un regalo que no haya sido primero introducido en las preferencias de ese receptor.
- i) No se permitirá la modificación del nombre de un receptor si se ha indicado alguna preferencia para él.
- j) Si borramos un cliente o receptor debe desaparecer toda la información asociada a ellos.

Sobre el grafo relacional obtenido en el apartado anterior, escribir en Álgebra Relacional y SQL las siguientes consultas:

- Nombres de los receptores ubicados en la provincia de Cuenca.
- Importe medio de las compras realizadas en los últimos 90 días.
- Nombres de categorías y número de unidades vendidas para cada una de ellas en el último año ordenadas de mayor a menor número de unidades.
- Nombre del cliente que más ha comprado durante el último año.

Discusión del enunciado

La manera en la que se abordará la resolución de este ejercicio será la siguiente: partiendo del grafo relacional inicial aportado en el enunciado, se extenderá o modificará según las especificaciones.

- a) La descripción de la categoría por la que se agrupan los regalos no es obligatoria.

Este requisito no se contempla en el grafo inicial y se puede reflejar a través

del mecanismo de opcionalidad (Figura 2.13(a)).

- b) Cada cliente tiene un DNI que lo identifica.

Esta restricción no se recoge en el grafo relacional resultante de la inclusión de la restricción del apartado anterior, pero puede ser añadida a través de la definición de clave alternativa para el atributo *DNI* en la tabla *CLIENTES* (Figura 2.13 (b)).



También se podría haber considerado en la modificación de la clave primaria de la tabla *CLIENTES*, y añadir el *DNI* como clave primaria y el *Cod_Cliente* como alternativa. Esta opción necesita realizar más cambios en el grafo relacional que la anterior sin ningún beneficio en el diseño, razón por la que se ha elegido la primera opción explicada.

- c) Se puede eliminar una categoría, pero no se eliminarán los regalos de esa categoría.

Para que se pueda contemplar este requisito en el grafo relacional, ya que no se refleja, es necesario especificar la opción de borrado de la clave ajena categoría en la relación *REGALOS* como puesta a nulos o a valor por defecto. Esta última exigiría marcar una categoría para todos los regalos descalificados (no se especifica), y por ello se adoptará la otra opción. Para ello, se debe modificar este atributo para que admita valores nulos y posteriormente poder definir la restricción de integridad de borrado con puesta a nulos. En la Figura 2.13 (c) muestra la opción elegida (aparecen dos restricciones añadidas, la opcionalidad del atributo *ca158nvía158ría* la opción de borrado con puesta a nulos).



Es muy importante recordar que para poder utilizar la opción de borrado con puesta a nulos o con valor por defecto, la clave ajena donde se define debe ser opcional o tener un valor por defecto definido, si no esta opción no tendrá sentido.

- d) Si el valor correspondiente a *id_categoria* es modificado en la tabla *REGALOS* esa modificación se propagará a categoría.

No es posible añadir al grafo relacional ningún elemento que recoja la restricción especificada en este apartado, ya que el valor correspondiente al

id_categoria en la tabla *REGALOS* es clave ajena que referencia a la tabla *CATEGORIAS* (*id_referencia*), y no al revés. Así, modificar la opción de integridad referencial en la clave ajena de la relación *REGALOS* no afecta a esta restricción.



Las restricciones de integridad se definen en las claves ajenas e indican cómo debe actuar el SGBD si se producen operaciones de borrado y de modificación en la tabla a la que referencian las claves ajenas, pero no al contrario.

- e) Es necesario almacenar en el esquema de la base de datos las preferencias que el cliente señala acerca del receptor del regalo. Las preferencias indican qué regalos son los favoritos de la persona a la que se quiere regalar.

Esta restricción no se recoge en el esquema relacional y para incluirla se debería definir una nueva relación donde se especificara el receptor y los regalos que un cliente indica como preferentes para este receptor. Realmente, el cliente en sí no es necesario que sea incluido (será el que introduzca esta información, pero quién introdujo estas preferencias no es relevante). En la Figura 2.13 (se muestra la nueva relación, la clave primaria y las claves ajenas definidas para contemplar esta restricción (sin las opciones de integridad referencial pues en esta especificación no se indican; se esperará a ver si en las siguientes son definidas).

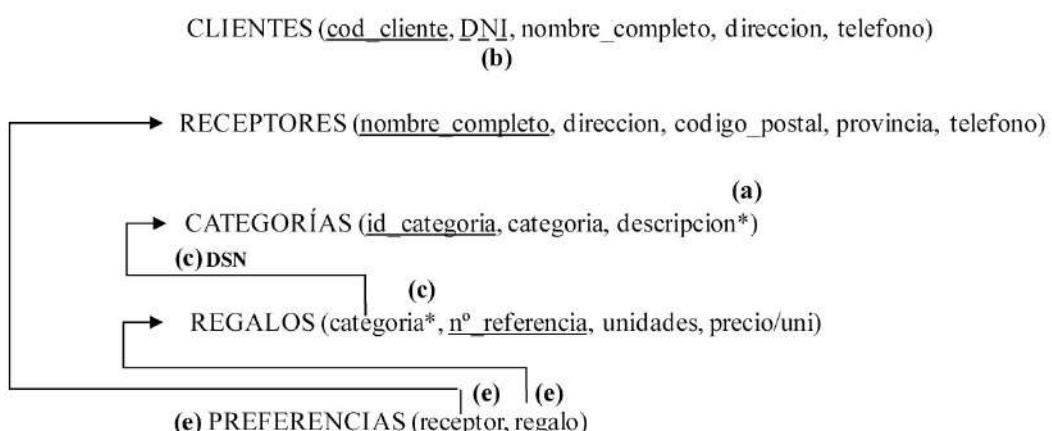


Figura 2.13. Grafo relacional resultante de la actualización del grafo inicial con las especificaciones desde la a) hasta la e)

- f) La compra siempre será de un regalo y queremos almacenar el día en

que se efectúo la misma y la dedicatoria (si la tiene). Un regalo es comprado por un cliente para un receptor y no puede ocurrir que un mismo regalo sea enviado a la misma persona el mismo día.

El requisito especificado no se presenta en el grafo relacional mostrado en la Figura 2.13 pero puede ser añadido a través de la creación de una nueva relación denominada COMPRAS donde se almacene el cliente, el receptor, el regalo, la fecha en la que se envía regalo y la dedicatoria (opcional, Figura 2.14). Para recoger todas las especificaciones presentadas en este apartado se define (véase la Figura 2.14) la clave primaria compuesta de los atributos *receptor*, *regalo*, *fecha* de esta manera se contempla la restricción *Un regalo es comprado por un cliente para un receptor y no puede ocurrir que un mismo regalo sea enviado a la misma persona en un mismo día*. También se debe añadir que el cliente, el receptor y el regalo son claves ajenas que referencian a las relaciones *CLIENTES*, *RECEPTORES* y *REGALOS*, respectivamente.

- g) Cada vez que se realiza una compra de un determinado regalo se debe disminuir el número de unidades disponibles de ese regalo en la tienda (unidades en Regalos).

Esta especificación no se encuentra reflejada en el grafo relacional de la Figura 2.14 y no se puede reflejar. Lo que se indica con esta restricción es que el valor del atributo *unidades* en la relación *REGALOS* debe ser modificado de manera automática cada vez que se realice una compra. En la fase de implementación, se podría contemplar definiendo un disparador asociado a la tabla *COMPRAS* para el evento de inserción, que actuaría modificando el atributo *unidades* del regalo que ha sido comprado.

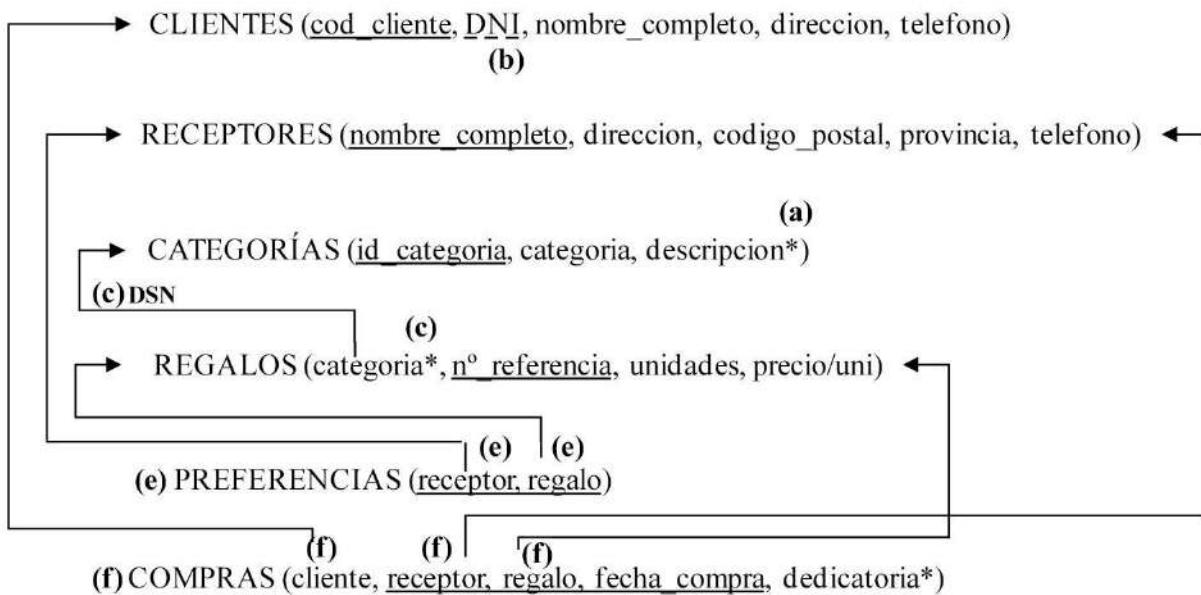


Figura 2.14. Grafo relacional con la inclusión de las restricciones especificadas hasta el apartado (f)

- h) El regalo comprado debe aparecer en las preferencias señaladas por el cliente para la persona que va a recibir el regalo elegido, es decir, no se podrá seleccionar un regalo que no haya sido primero introducido en las preferencias de ese receptor.

La restricción especificada no se contempla en el grafo relacional de la Figura 2.14 y para tenerla en cuenta se debe modificar la relación a la que referencian las claves ajenas *receptor* y *regalo* en la relación *COMPRAS*. En vez de referenciar a las tablas *RECEPTORES* y *REGALOS* por separado, se hará que conjuntamente referencien a la tabla *PREFERENCIAS*. De esta manera, no se podrá insertar una nueva compra si antes no se ha introducido la tupla correspondiente en las preferencias del receptor. Con esto se evita que un receptor reciba un regalo que no le guste (Figura 2.15 (h)).

- i) No se permitirá la modificación del nombre de un receptor si se ha indicado alguna preferencia para él.

Claramente se está definiendo la opción de modificación de la clave ajena *receptor* en la tabla *PREFERENCIAS*. La opción que se indica es la no acción que se añade al esquema presentado en la Figura 2.15 (i).

- j) Si borramos un cliente o receptor queremos que desaparezca toda su información.

Con esta restricción se indican los modos de borrado, que son en cascada, para todas aquellas claves ajenas que refieren a las tablas *CLIENTES* y *RECEPTORES*. Véase la Figura 2.15 (j).

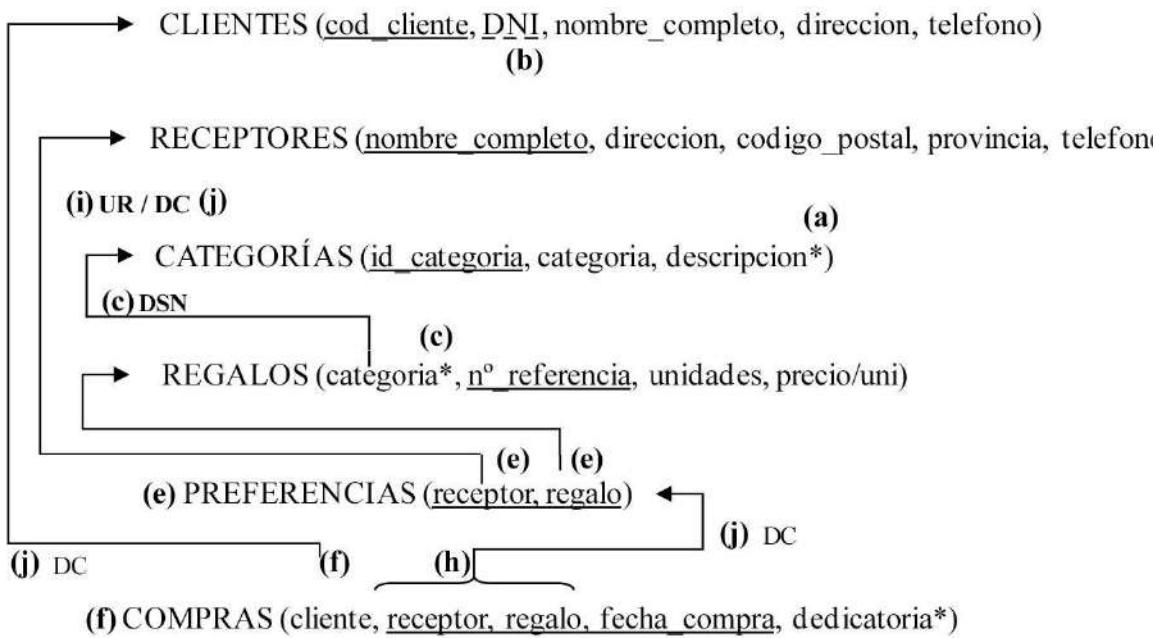


Figura 2.15. Grafo relacional resultante de la actualización del grafo inicial con las nuevas especificaciones

CONSULTAS

Nombres de los receptores ubicados en la provincia de Cuenca.

ÁLGEBRA RELACIONAL

$$\prod_{\text{nombre_completo}} \left(\sigma_{\text{provincia}='Cuenca'}(\text{RECEPTOR}) \right)$$

SQL

```
SELECT nombre_completo  
FROM RECEPTOR  
WHERE provincia = 'Cuenca' ;
```

Importe medio de las compras realizadas en los últimos 90 días.

ÁLGEBRA RELACIONAL

$$\prod_{AVG(\text{precio/uni})} [\sigma_{fecha_compra > hoy - 90}(COMPRAS)]$$

SQL

```
SELECT AVG(precio/uni)
FROM COMPRAS
WHERE fecha_compra > hoy - 90;
```

Nombres de categorías y número de unidades vendidas para cada una de ellas ordenadas de mayor a menor número de unidades.

ÁLGEBRA RELACIONAL

Primero se combinarán las relaciones *REGALOS* y *COMPRAS* agrupando el resultado según categorías, de manera que pueda aplicarse la función de agregado *COUNT* y obtener el número de unidades vendidas por categorías.

$$\prod_{\text{COUNT(*) AS unidades}} [\text{GROUP BY}_{\text{categoria}} [\text{COMPRAS *}_{\text{regalo = n}^{\text{a}} \text{referencia}} \text{REGALOS}]] \quad (1)$$

El resultado de (1) estará formado por los identificadores de las categorías y el número de unidades vendidas para cada categoría. Es necesario combinar este resultado con la tabla *CATEGORÍA* con el fin de obtener el nombre de la categoría en lugar del identificador.

$$\prod_{\text{CATEGORIA.categoría,unidades}} [CATEGORÍA *_{\text{id_categoria=categoría}} (1)]$$

SQL

```
SELECT C.categoría, Unidades  
FROM CATEGORIA C JOIN  
(SELECT categoría, COUNT(*) AS Unidades  
FROM COMPRAS C JOIN REGALOS R  
ON C.regalo=R.nº_referencia  
GROUP BY categoría ) Q  
ON C.id_categoria = Q.categoría;
```

Nombre del cliente que más ha comprado durante el último año.

ÁLGEBRA RELACIONAL

Primero se combinarán las relaciones *COMPRAS*, *CLIENTES* y *REGALOS* obteniendo una relación de cada compra realizada por cada cliente junto con el precio del regalo durante el último año.

$$\sigma_{\text{fecha_compra} > \text{hoy} - 365} (\text{COMPRAS}) \times_{\text{cod_cliente} = \text{cliente}} \text{CLIENTES} \times_{\text{regalo} = \text{nº_referencia}} \text{REGALOS} \quad (1)$$

Sobre el resultado de (1) se realizará un agrupamiento sobre el cliente, de manera que pueda aplicarse una función de agregado que calcule la suma de los precios de todos los productos comprados por cada cliente.

$$\sigma_{\text{nombre_completo}, \text{MAX(total)}} \left(\prod_{\text{nombre_completo}, \text{SUM(precio/uni)} \text{ AS total}} [\text{GROUP BY}_{\text{cod_cliente}}((1))] \right)$$

SQL

```
SELECT nombre_completo, MAX(total) FROM  
  
    (SELECT nombre_completo, SUM(precio/uni) AS  
     total  
  
      FROM COMPRAS C JOIN  
  
        (CLIENTES CL JOIN REGALOS R  
  
          ON CL.regalo=R.nº_referencia)  
  
          ON C.cliente=CL.cod_cliente  
  
WHERE fecha_compra > hoy - 365  
  
GROUP BY CL.cod_cliente);
```

PROBLEMA 2.3: COMPAÑÍA TEATRAL

La compañía teatral “Thaliapatos” necesita organizar la información que posee sobre sus actividades. Estas, naturalmente, se centran en el mundo de la farándula.

En primer lugar, se debe registrar la información de un/a artista: nombre, apellidos, nombre artístico, edad, caché, descripción, y papeles con los que es afín (que ya ha interpretado alguna vez), y cuántas veces ha interpretado cada papel. Naturalmente, un artista tiene en mucho orgullo que el nombre artístico es único y no puede ser copiado por nadie.

Los “papeles” son personajes en una obra concreta. De cada papel hay que almacenar el nombre del personaje (único en su obra), la duración del mismo en escena, y una descripción del atrezo. Respecto de la obra, es preciso registrar su título (siempre único), autor, año de publicación, y veces que se ha representado. De los autores es preciso observar su nombre, apellidos, nombre artístico (opcional), edad, y cuánto suelen cobrar por libreto y cuánto por representación.

Desgraciadamente, los artistas y escritores son gente temperamental y en ocasiones ocurre que deciden apartarse de una compañía. Si un artista se apartara de nuestra compañía, se perdería su información, incluyendo qué papeles es capaz de representar, si bien esos papeles tienen que permanecer en la base (pueden existir otros actores a los que les sean afines). Por otro lado, si es un escritor el que se despide, además de la información del mismo, se ha de prescindir de toda la información concerniente a sus obras (incluyendo los papeles de cada obra en su caso).

Por otro lado, la compañía cuenta con sus propios locales (teatros), que vienen caracterizados por un nombre, dirección, localidad, provincia, teléfono, categoría (opcional) y aforo. En estos teatros se desarrollan funciones, que son la representación de una obra en una fecha concreta. Como es natural, una función tiene asociados unos artistas para que pueda ser representada, si bien este trabajo lo va desarrollando un gestor a medida que habla con los artistas. Por lo tanto, pueden existir funciones que aún no tengan asignados todos los artistas necesarios. Para que un artista sea asociado a una obra, es imprescindible que se haya registrado anteriormente una afinidad del mismo con el papel (aunque el número de veces que lo haya interpretado pueda ser 0).

Si una obra es vetada en nuestra compañía, no se van a cancelar todas las funciones de la misma si bien habrá que escoger otra obra para representar. En estos casos, la obra que se representará queda en incógnita temporal (la función sigue registrada, pero la obra adopta el valor nulo). Si se cierra un local, deberán desaparecer las funciones que estaban programadas en el mismo. Si una obra cambia su título, deberán desprogramarse todas las funciones previstas (la obra se queda con el valor “Reprogramada”).

Se pide:

1. Confeccionar un grafo relacional que se adapte a las características del problema.
2. Realiza en Álgebra Relacional y en SQL las siguientes consultas:
 - Nombre y apellidos del artista al que llaman “El brujo”.
 - Títulos de las obras representadas en el teatro Lope de Vega en las que aparece el personaje “Arlechino”.
 - Títulos de las obras que han sido vistas por más de 10.000 personas (como es habitual, hay que suponer que el aforo siempre está lleno).
 - Funciones, representadas por el nombre del teatro y la fecha en que están programadas, que no tienen asignados todos los artistas que precisan.

Discusión del enunciado

El grafo relacional resultante después de tener en consideración las especificaciones aparece en la Figura 2.16.



En el grafo de la Figura 2.16 conviene notar que, en algunos casos, como el atributo *autor* de la relación *OBRAS*, es en realidad un atributo compuesto, es decir, representa los dos atributos que constituyen la clave primaria de *AUTORES*. De la misma forma, podría sustituirse ese atributo *autor* en *OBRAS* por dos atributos *nombre_a* y *apellido_a*.

A continuación se describen algunas de las consideraciones que se han tenido en cuenta para modelar los requisitos especificados en el texto.

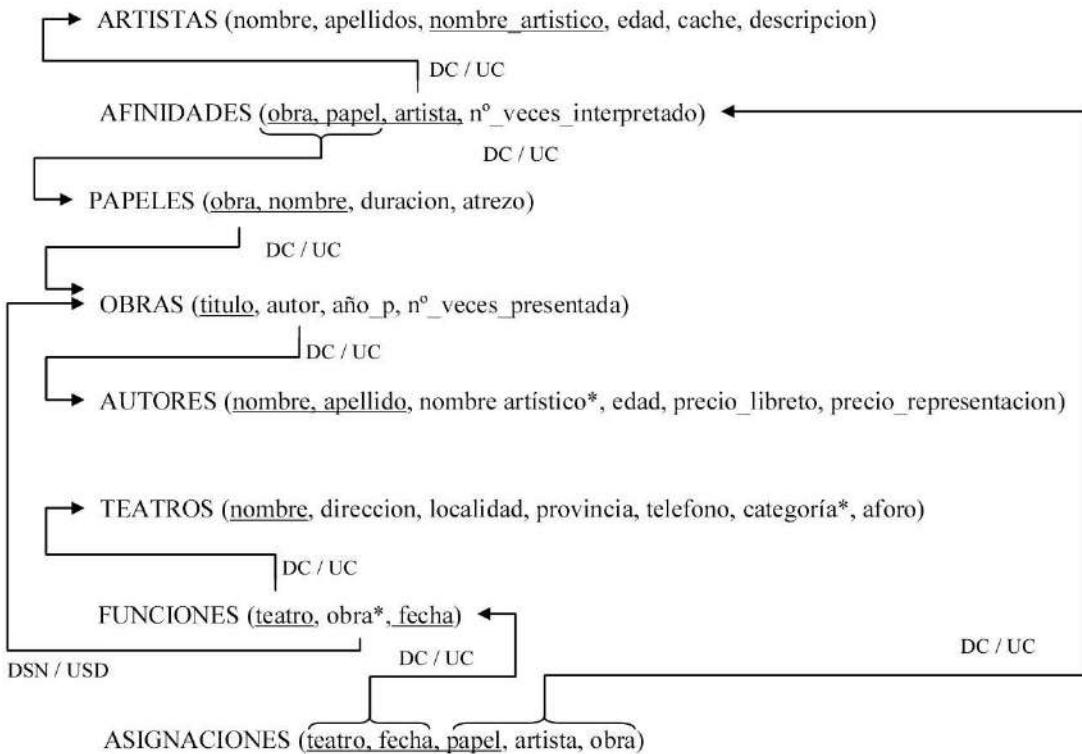


Figura 2.16. Grafo relacional correspondiente al Problema 2.3

Para recoger la siguiente especificación ...*Para que un artista sea asociado a una obra, es imprescindible que se haya registrado anteriormente una afinidad del mismo con el papel (aunque el número de veces que lo haya interpretado pueda ser 0)...* es necesario definir una clave ajena en la relación ASIGNACIONES que haga referencia a la relación AFINIDADES, de esta manera el mecanismo de clave ajena hará que nunca se asigne un papel de una obra a un artista si antes no lo ha interpretado (es decir, debe existir una tupla en la tabla AFINIDADES). La Figura 2.17 muestra la relación descrita.

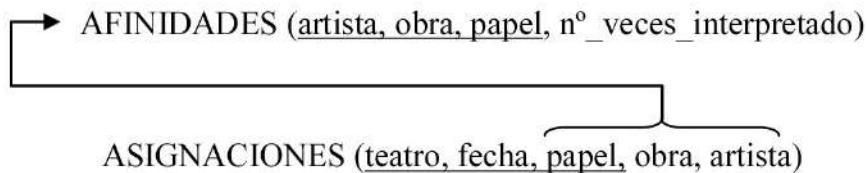


Figura 2.17. Representación de la restricción 1

Debemos añadir que el atributo nº_veces_interpretado tomará el valor por defecto 0 y se deberá controlar posteriormente, en la fase de implementación, que este valor se incremente en 1 cada vez que un artista sea asignado a un papel.

en la tabla *ASIGNACIONES*.

Las opciones de borrado y de modificación de claves ajena que han sido recogidas del texto son analizadas en este apartado. De las consideraciones... *Desgraciadamente, los artistas y escritores son gente temperamental y en ocasiones ocurre que deciden apartarse de una compañía. Si un artista se apartara de nuestra compañía, se perdería su información, incluyendo qué papeles es capaz de representar, si bien esos papeles tienen que permanecer en la base (pueden existir otros actores a los que les sean afines). Por otro lado, si es un escritor el que se despide, además de la información del mismo, se ha de prescindir de toda la información concerniente a sus obras (incluyendo los papeles de cada obra en su caso)...* se deduce que el borrado de la clave ajena artista en *AFINIDADES* es en cascada, permaneciendo los papeles en la base de datos ya que esta opción no afecta a la relación *PAPELES*. Las claves ajena correspondientes al autor en *OBRAS*, y obra en *PAPELES* también tendrán un borrado en cascada (Figura 2.18) pues toda la información relacionada con el autor debe desaparecer. De igual modo, aunque no se expone de manera explícita en el texto, también se escoge un borrado en cascada para la clave ajena obra, papel en *AFINIDADES* y para la clave ajena *artista obra papel* en *ASIGNACIONES*. De esta manera se eliminarán todas las tuplas referentes a la obra de un autor, cuando éste se aparte de la compañía.

El otro párrafo del texto donde se indican opciones de borrado y modificación dice: ...*Si una obra es vetada en nuestra compañía, no se van a cancelar todas las funciones de la misma si bien habrá que escoger otra obra para representar. En estos casos, la obra que se representará queda en incógnita temporal (la función sigue registrada, pero la obra adopta el valor nulo)...* Según esto el borrado de la clave ajena obra en *FUNCIONES* debe ser con puesta a nulos (*Set Null*), de esta manera si una obra es eliminada de la relación *OBRAS* y todavía tiene funciones, estas funciones aparecen sin tener asignada obra (véase la Figura 2.19 (a)).

...*Si se cierra, un local, deberán desaparecer las funciones que estaban programadas en el mismo...* Para ello, la clave ajena teatro en *FUNCIONES* tendrá la opción de borrado en cascada, así cuando se elimine una tupla en la relación *TEATRO* se borrarán todas las tuplas que referencian a ese teatro en la relación *FUNCIONES* (véase la Figura 2.19 (b)).

...*Si una obra cambia su título, deberán desprogramarse todas las funciones previstas (la obra se queda con el valor Reprogramada)...* La opción de modificación más adecuada para la clave ajena obra en *FUNCIONES* es con

puesta a un valor por defecto que, en este caso, será *reprogramada*. Por lo tanto, tendremos que definir que este atributo obra tome este valor por defecto (véase la Figura 2.19 (c)).

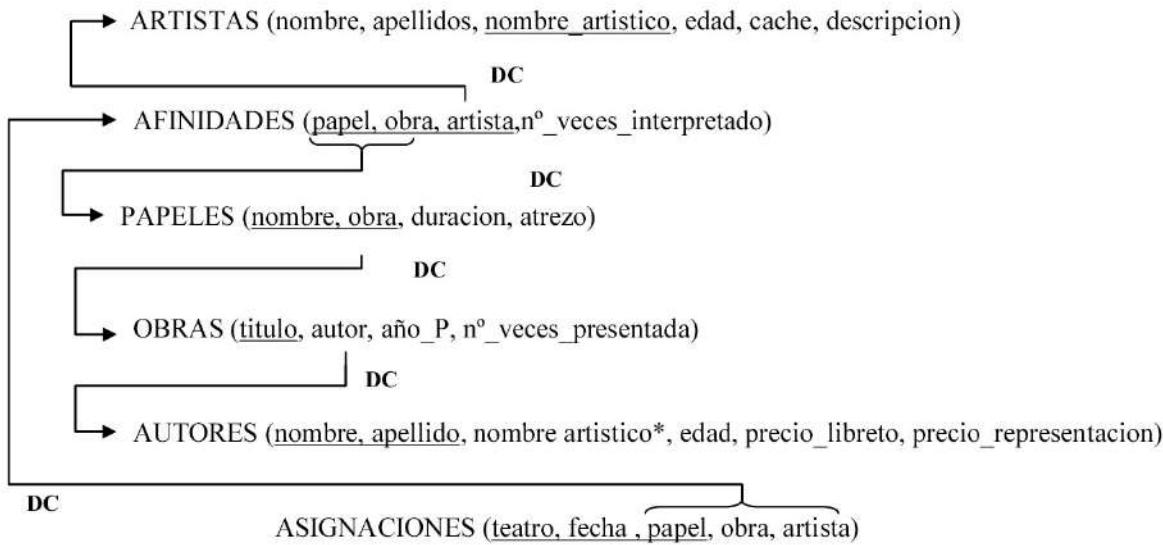


Figura 2.18. Opciones de integridad especificadas en el enunciado

En la fase de implementación de la base de datos, en la tabla **AFINIDADES** hay que incluir una aserción que asegure que un artista no interviene en la misma fecha en varias obras y teatros.

Finalmente, se presentan los supuestos semánticos que se han tenido en cuenta para completar la fase de diseño lógico.

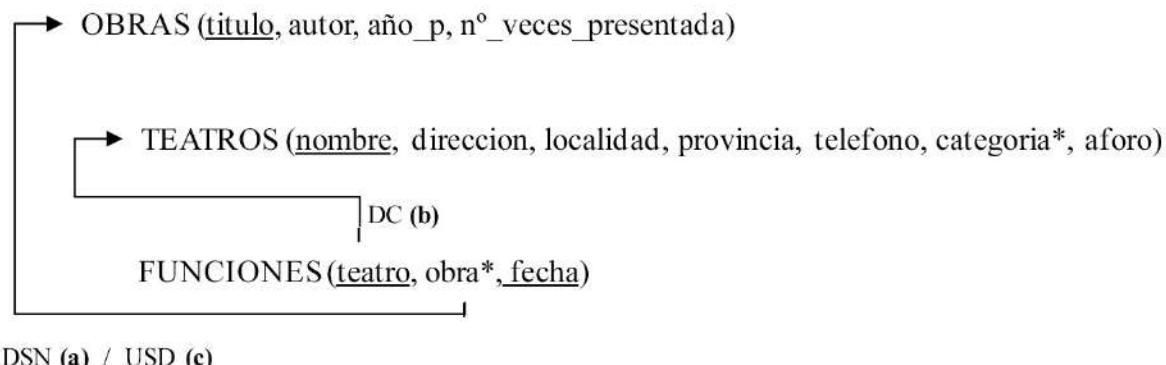


Figura 2.19. Opciones de borrado y modificación destacadas

Supuestos semánticos añadidos por el diseñador:

- Todas las opciones de modificación de las claves ajena que no han sido descritas anteriormente, han sido supuestas. Se ha tomado la modificación en cascada para todos estos casos considerando que si no se ha indicado lo contrario en las especificaciones del dominio es la opción menos restrictiva.
- El nombre de los teatros es único y, por ello, se ha elegido como clave primaria de esta relación.

CONSULTAS

Nombre y apellidos del artista que llaman “El Brujo”.

ÁLGEBRA RELACIONAL

$$\prod_{\text{nombre ,apellidos}} [\sigma_{\text{nombre-a_artistico}='El Brujo'}(\text{ARTISTAS})]$$

SQL

```
SELECT nombre, apellidos  
FROM ARTISTAS  
WHERE nombre_artistico = 'El Brujo';
```

Títulos de las obras representadas en el teatro Lope de Vega en las que aparece el personaje “Arlechino”.

ÁLGEBRA RELACIONAL

$$\prod_{obra} \left[\sigma_{nombre='Arlechino'}(PAPELES) *_{obra} \left(\sigma_{teatro='Lope de Vega'}(FUNCIONES) \right) \right]$$

Conviene destacar que el atributo *teatro* de la relación *FUNCIONES* contiene directamente el nombre del teatro, no es necesaria una combinación con la relación *TEATROS* para filtrar únicamente el teatro de nombre “Lope de Vega”.

SQL

```
SELECT P.obra AS Obra  
  
FROM PAPELES P JOIN FUNCIONES F  
  
ON P.obra = F.obra  
  
WHERE P.nombre = 'Arlechino' AND  
  
F.teatro = `Lope de Vega`;
```

Otra forma de expresar esta consulta en SQL:

Títulos de las obras que han sido vistas por más de 10.000 personas (como es habitual, hay que suponer que el aforo siempre está lleno).

ÁLGEBRA RELACIONAL

$$\prod_{obra} \left[\sigma_{SUM(aforo) > 10000} \left[GROUP\ BY_{obra} [TEATROS^*_{\text{nombbre=teatro}} FUNCIONES] \right] \right]$$

SQL

```
SELECT f.Obra  
  
FROM FUNCIONES f JOIN TEATROS t  
  
ON (f.teatro = t.nombre)  
  
GROUP BY f.Obra  
  
HAVING SUM(aforo)>10.000;
```

Funciones, representadas por el nombre del teatro y la fecha en que están programadas, que no tienen asignados todos los artistas que precisan.

ÁLGEBRA RELACIONAL

Esta consulta se calculará como la diferencia entre el conjunto formado por todas las funciones y el constituido por aquellas funciones que tienen ya todos los papeles asignados. En la primera parte de la consulta es necesaria la combinación de *PAPELES* con *FUNCIONES* para asegurar que se tienen en cuenta todos los papeles que hacen falta para una función, en caso contrario no se obtendrían aquellas funciones que tienen solo alguno de sus papeles asignados.

En las consultas donde se emplean los operadores diferencia y unión es necesario tener en cuenta que los esquemas de las relaciones que intervienen deben ser compatibles en sus esquemas, es decir, deben tener el mismo número de atributos y estos deben estar definidos sobre los mismos dominios.

$$\prod_{teatro, fecha} \left[\prod_{nombre, obra, teatro, fecha} (PAPELES *_{obra} FUNCIONES) - \prod_{papel, obra, teatro, fecha} (ASIGNACIONES) \right]$$

SQL

```
SELECT TF.teatro, TF.fecha FROM  
  
(SELECT nombre, obra, teatro, fecha  
FROM PAPELES P JOIN FUNCIONES F  
ON P.obra = E.obra) TF  
  
EXCEPT  
  
(SELECT papel,obra,teatro,fecha  
FROM ASIGNACIONES)
```

PROBLEMA 2.4: CAMPEONATO DE AJEDREZ

La federación internacional de ajedrez desea disponer de una base de datos con todas las partidas de las que tenga noticia así como de las federaciones nacionales existentes.

- Cualquier partida viene caracterizada por una fecha (de inicio), una duración (en horas), jugadores que se enfrentaron en la misma, y si lo hicieron con piezas blancas o negras.
- Si bien dos jugadores cualesquiera pueden enfrentarse varias veces, ha de ser siempre en fechas distintas (solo un enfrentamiento por fecha y contendientes).
- También hay que recoger cada movimiento de la partida, que básicamente tiene un número ordinal, casilla de origen, y casilla destino. La existencia de estos movimientos está ligada a la de la partida (si ésta fuera eliminada de la base, también habrían de eliminarse aquellos).
- Una vez comenzada una partida (cuando se haya realizado el primer movimiento) ésta no podrá ser aplazada.
- En una partida de ajedrez solo intervienen dos jugadores, uno con piezas blancas, y el otro con piezas negras.
- De los jugadores es necesario recoger sus datos personales: nombre y apellido, dirección postal (que no siempre se conoce) y dirección electrónica (opcional). También hay que recoger a qué federación pertenece y el número de federado (que es único dentro de cada federación).
- De una federación, que es única en cada país, es necesario registrar el nombre, país al que pertenece, teléfono de contacto, y fecha de fundación.
- Una federación no puede ser clausurada hasta que sus jugadores no hayan sido reasignados a otras federaciones o eliminados de la base.
- Las partidas se encuadran a menudo en una edición de un torneo (aunque no siempre). Estos torneos tienen un nombre característico, periodicidad (anual, bienal...) fecha de creación, y están organizados

siempre por una federación.

- Cada vez que se celebra una edición (ordinal de un torneo), se registran todos los enfrentamientos y el nombre del ganador. Otra información útil es la fecha de inicio, cuantía del premio y jugador que lo ha ganado.
- Si una federación desaparece, desaparecen también sus torneos y, por ende, sus ediciones.
- Aunque un jugador desaparezca, sus partidas no deben desaparecer (puesto que son información válida para otro jugador).

Se pide:

1. Grafo Relacional que describa todas estas necesidades (que recoja todos estos supuestos semánticos).
2. Escribir la(s) instrucciones SQL para añadir la descripción de una tabla *PRÓXIMOS ENFRENTAMIENTOS*, que recoja todos los enfrentamientos (nombres de los jugadores y fecha) programados para ser celebrados hoy.
3. Describe en Álgebra Relacional y en SQL las siguientes consultas:
 - a) Nombre de los torneos que en alguna edición hayan tenido un premio mayor de 100.000 €. Aunque varias ediciones de un torneo cumplieran la condición, el nombre del torneo solo debe aparecer una vez.
 - b) Partidas con menos de 20 movimientos.
 - c) Número de partidas jugadas por Robert Fisher.
 - d) Nombre del jugador que ha empleado el menor número de movimientos para ganar una partida.

Discusión del enunciado

Grafo Relacional que describa todas estas necesidades (que recoja todos estos supuestos semánticos).

El grafo resultante se muestra en la Figura 2.20.

A continuación se comentan aquellas decisiones de diseño más destacadas.

La clave primaria de la relación *PARTIDAS* se compone de tres atributos que describen la fecha en la que se jugó la partida (*fecha_ini*), el jugador que participa con fichas blancas (*jugador_B*) y el jugador que participa con negras (*jugador_N*). De esta manera se cumple la restricción impuesta en los requisitos que especifica *Si bien dos jugadores cualesquiera pueden enfrentarse varias veces, ha de ser siempre en fechas distintas (solo un enfrentamiento por fecha y contendientes)*.

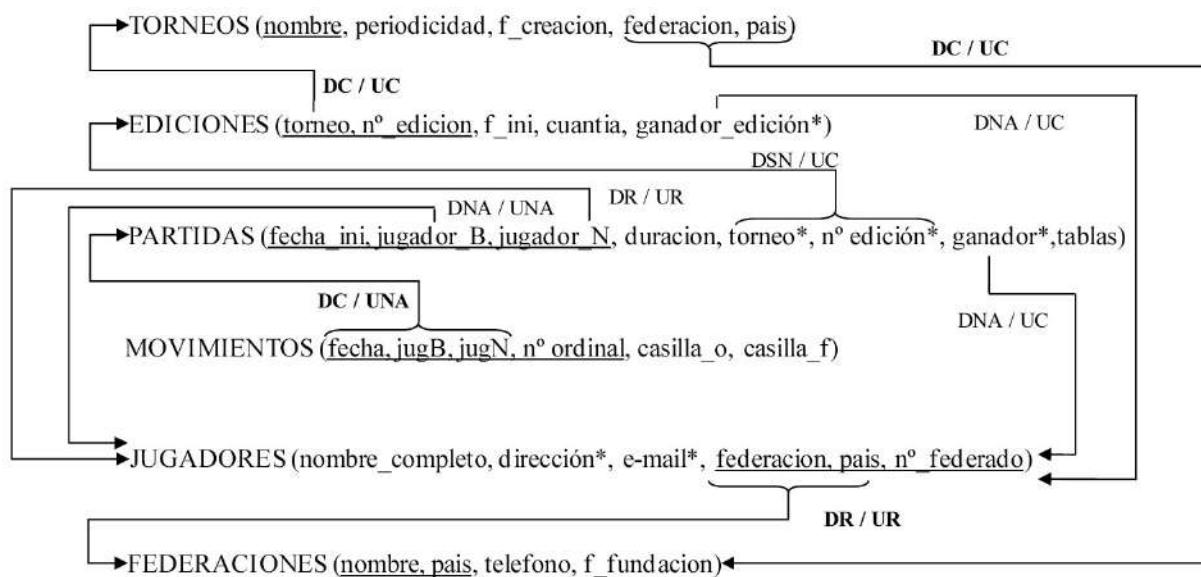


Figura 2.20. Grafo relacional

Las opciones de borrado y modificación que pueden obtenerse a través de las especificaciones del texto se han indicado en la Figura 2.20 en negrita. Para la relación *MOVIMIENTOS* éstas se derivan de la frase *La existencia de estos movimientos está ligada a la de la partida (si ésta fuera eliminada de la base, también habrían de eliminarse estos)*. La cual nos indica que la clave ajena *fecha jugB jugN* tendrá la opción de borrado en cascada. La opción de modificación

para esta clave, por el contrario, será sin acción ya que se nos indica que *Una vez comenzada una partida (cuando se haya realizado el primer movimiento) ésta no podrá ser aplazada* por lo que ni la fecha de la partida ni los jugadores podrán ser modificados si existe algún movimiento que les haga referencia. Para la clave ajena *federacion, pais* de la relación *JUGADORES* se especifica que *Una federación no puede ser clausurada hasta que sus jugadores no hayan sido reasignados a otras federaciones o eliminados de la base* por lo que la restricción de integridad tanto para el borrado como para la modificación será sin acción. Y, por último, para recoger el supuesto de *Si una federación desaparece, desaparecen también sus torneos y por ende sus ediciones* la opción de borrado para la clave ajena *federación, pais* de *TORNEOS* será en cascada y lo mismo ocurrirá para la clave ajena torneo de *EDICIONES*.

Supuestos semánticos añadidos por el diseñador:

- Las restricciones de integridad que no aparecen en las especificaciones han sido supuestas como en los problemas anteriores, los borrados sin acción, para que esta operación sea controlada y las modificaciones en cascada.
- En las relaciones *JUGADORES* y *EDICIONES* los atributos *ganador* y *ganador_edicion* admiten valores nulos con el fin de poder recoger situaciones en las que no se conoce al ganador de alguna de las partidas o ediciones.
- Si en una partida se producen tablas debe ponerse el atributo *tablas* a valor *si* debiendo quedar el atributo *ganador* con valor nulo. Esta situación puede validarse mediante el siguiente *CHECK* en la relación *PARTIDAS* que, además, asegura que el ganador es uno de los jugadores de la partida:

```
ALTER TABLE PARTIDAS ADD CONSTRAINT ganador_partida  
CHECK ((ganador=jugador_B) OR (ganador=jugador_N)  
OR ((ganador=NULL) AND (tablas='si')));
```



Es importante notar que si se eliminase el atributo *tablas* y se incluyese un valor nulo en el atributo *ganador* cuando se producen tablas sería imposible distinguir las partidas que han acabado en tablas de aquellas para las que no se sabe quién ganó.

Supuestos semánticos no recogidos en el grafo relacional:

- En la relación *PARTIDAS*, las claves ajenas *jugador_B* y *jugador_N* deben tener un valor distinto para cada tupla ya que no se puede dar el caso de que un único jugador participe en una partida. Para controlar esta restricción se podría añadir un *CHECK*, una vez se haya creado la relación, tal y como se muestra a continuación:

```
ALTER TABLE PARTIDAS ADD CONSTRAINT Jugadores_distintos  
CHECK ( jugador-B <> jugador-N) ;
```



Se advierte al lector que la notación empleada en un grafo relacional permite agrupar varios atributos bajo un único nombre. Por ejemplo, en la Figura 2.20 el atributo clave ajena *ganador* de la relación *PARTIDA* equivale a los tres atributos que componen la clave primaria de *JUGADORES*, que son *federacion*, *pais* y *nº_federado*. En la fase de implementación habrá que tener en cuenta que, en lugar del mencionado atributo *ganador* habrá que crear tres atributos en su lugar, de lo contrario obtendremos un error.

- Tanto el ganador de una edición de un torneo como el de cada partida son jugadores cuyos datos figuran en la relación *JUGADORES* pero es necesario asegurar que el ganador de una edición de un torneo ha participado en alguna de las partidas de ese torneo. Si se conociese el número de partidas que debe ganar un jugador para hacerse con un torneo podría tenerse en cuenta no solo que el ganador haya participado en alguna partida de ese torneo si no que podría comprobarse si ha ganado el número necesario de partidas. En la fase de implementación estas restricciones se validarían mediante aserciones y disparadores asociados a la relación *EDICIONES*. En la relación *PARTIDAS*, los atributos *torneo* y *edicion* son ambos nulos o ninguno (un *CHECK* controla esta restricción).
- Con los mismos jugadores, dos partidas no empiezan el mismo día (según la clave primaria de *PARTIDAS*) aunque sí puede darse que un jugador tenga dos partidas simultáneas. Para controlar esta restricción se debería crear una aserción en la fase de implementación.

Escribir la(s) instrucciones SQL para añadir la descripción de una tabla *PRÓXIMOS ENFRENTAMIENTOS*, que recoja todos los enfrentamientos

(nombres de los jugadores y fecha) programados para ser celebrados hoy.

```
CREATE VIEW PROXIMAS_PARTIDAS (SELECT fecha_ini,  
jugador_B, jugador_N FROM PARTIDAS  
WHERE fecha_ini =  
SYSDATE);
```

CONSULTAS

Nombre de los torneos que en alguna edición hayan tenido un premio mayor de 100.000 €. Aunque varias ediciones de un torneo cumplieran la condición, el nombre del torneo solo debe aparecer una vez.

ÁLGEBRA RELACIONAL

$$\Pi_{torneo} (\sigma_{cuantía > 100.000} (EDICIONES))$$

Hay que tener en cuenta que el atributo *torneo* de *EDICIONES* tiene una restricción de clave ajena con *TORNEOS*, con lo que el nombre del torneo, que es clave primaria de esa relación, se encuentra también en la relación *EDICIONES*. Por otro lado, el operador de proyección elimina los elementos repetidos del conjunto de resultados, cumpliéndose la restricción pedida en el enunciado de la consulta.

SQL

```
SELECT DISTINCT torneo  
FROM EDICION  
WHERE cuantia > 100.000;
```

Partidas con menos de 20 movimientos.

ÁLGEBRA RELACIONAL

$\Pi_{fecha, jugB, jugN} [\sigma_{COUNT(*) < 20} (GROUP BY_{fecha, jugB, jugN} (MOVIMIENTO S))]$

SQL

```
SELECT fecha, jugB, jugN  
FROM PARTIDAS  
GROUP BY fecha, jugB, jugN  
HAVING COUNT (*) < 20;
```

ÁLGEBRA RELACIONAL

$$\begin{aligned} & \sigma_{\text{COUNT}(*)} \left(\sigma_{\text{nombre_completo} = 'Robert Fisher'} (JUGADORES *_{\text{jugador_B}} \text{PARTIDAS}) \right. \\ & \left. \cup \sigma_{\text{nombre_completo} = 'Robert Fisher'} (JUGADORES *_{\text{jugador_N}} \text{PARTIDAS}) \right) \end{aligned}$$

En las condiciones de combinación de la consulta se han utilizado únicamente los atributos *jugador_B* y *jugador_N* por claridad. Ambos atributos estarían formados en realidad por tres atributos, los que identifican a cada jugador que son: *federacion*, *pais* y *nº_federado*, de manera que en la condición de combinación se compara cada uno de ellos por separado.

SQL

```
SELECT COUNT(*) FROM
((SELECT * FROM
JUGADORES J JOIN PARTIDAS P
ON J.federacion = P.jugador_B.federacion AND
J.pais = P.jugador_B.pais AND
J.n°_federado = P.jugador_B.n°_federado)
UNION
(SELECT * FROM
JUGADORES J JOIN PARTIDAS P
ON J.federacion = P.jugador_N.federacion AND
J.pais = P.jugador_N.pais AND
J.n°_federado = P.jugador_N.n°_federado))
```

En este tipo de consultas es necesario recordar que en los operadores para la unión y la diferencia de relaciones es necesario que los operandos sean compatibles en sus esquemas, es decir, que tengan el mismo número de atributos y que estos estén definidos sobre los mismos dominios.

Nombre del jugador que ha empleado el menor número de movimientos para ganar una partida.

ÁLGEBRA RELACIONAL

Primero se obtienen los movimientos de las partidas:

$$\prod_{COUNT(*) AS Cuenta, fecha, jugB, jugN, ganador} [GROUP BY_{fecha, jugB, jugN, ganador} [(1)]] \quad (2)$$

Como segundo paso, se calcula el número de movimientos por partida:

$$\prod_{COUNT(*) AS Cuenta, fecha, jugB, jugN, ganador} [GROUP BY_{fecha, jugB, jugN, ganador} [(1)]] \quad (2)$$

y, por último, se selecciona la partida con el mínimo número de movimientos y se obtiene el nombre del jugador que ganó:

$$\prod_{nombre_completo} [JUGADOR *_{ganador=federación, país, nº federado} (\sigma_{MIN(Cuenta)} [(2)])] \quad (3)$$

SQL

```
SELECT nombre_completo FROM  
  
  (SELECT MIN(Cuenta) ,ganador FROM  
  
    (SELECT COUNT(*) AS Cuenta,fecha,jugB,jugN  
  
      FROM PARTIDAS JOIN MOVIMIENTOS  
      ON(fecha_ini=fecha AND jugador_B = jugB  
      AND jugador_N=jugN)  
  
      GROUP BY fecha,jugB,jugN,ganador) ),  
  
  JOIN JUGADOR  
  
  ON ganador.federacion=federacion AND  
  
  ganador.pais=pais AND ganador.n°federado=federado;
```

PROBLEMA 2.5: FÁBRICA DE MUEBLES

La empresa de fabricación y venta al por mayor de muebles “El pino de Valsaín” desea confeccionar una base de datos para gestionar sus actividades. La empresa está organizada en cinco departamentos: producción, comercial, pedidos, finanzas, y dirección. De momento, se desea recoger tan solo las necesidades de los tres primeros.

Los requisitos del departamento de producción se centran en registrar todos los muebles que fabrican: su referencia (única para cada producto), su nombre genérico, estilo al que pertenece, nombre de la fábrica donde se realiza, y las materias primas (madera) que son necesarias para su construcción. La materia prima es proporcionada por aserraderos, de los que se precisa conocer el nombre, dirección, CIF, zona o zonas en las que opera (bosques de donde saca la madera), y maderas que provee. De las maderas hay que registrar su procedencia (árbol), su forma (tablero, listón, etc.), cuál es su fresado (opcional, no todas están fresadas), sus dimensiones (largo, ancho, alto), y su coste. La misma madera puede ser provista por varios aserraderos, variando solo el coste (cada proveedor pone el suyo).

Por su parte, el departamento comercial realiza campañas para promocionar algunos de sus artículos (o ninguno de ellos en particular). De éstas se ha de almacenar el nombre de la campaña (único para cada una), fecha de inicio y fin, zona, coste, y los productos que promociona (si los hay). Además, este mismo departamento se ocupa de la representación de la empresa en ferias del mueble. Éstas suelen venir caracterizadas por estar dedicadas a un solo estilo de mueble. Además de esta información, hay que registrar el nombre de la feria, el ordinal de la edición, fechas de celebración, muebles que allí se promocionan, y costes imputados a la feria.

Por otro lado, el departamento de pedidos controla los pedidos y las facturas. En los pedidos propios aparece el nombre del proveedor, maderas que incluye (y la cantidad de cada una), y las fechas de solicitud y de entrega. En los pedidos de clientes hacia la empresa, aparece el nombre del cliente, su CIF, muebles que ha pedido, fecha de solicitud, y fecha de entrega. De las facturas, se corresponde una por pedido, y hay que registrar la fecha de pago y el importe.

Se pide:

1. Diseñar un Esquema Relacional adecuado a estas especificaciones, sin olvidar las asociaciones entre relaciones y las reglas de integridad que apliquen.
2. Instrucciones SQL para garantizar que en ninguna feria se promocione ningún mueble cuyo estilo no esté en consonancia con el estilo de la feria.
3. Describir en Álgebra Relacional y en SQL las siguientes consultas:
 - a) Nombres de las campañas del año 2010.
 - b) Referencias de muebles para los que se necesita una madera que nadie provee.
 - c) Fábricas donde se producen muebles de pino.
 - d) Costes totales de promocionar en ferias el mueble de estilo rústico durante el año 2011.

Discusión del enunciado

El diseño del esquema relacional resultante de la consideración de las especificaciones se muestra en la Figura 2.21.

Supuestos semánticos añadidos por el diseñador:

- Se ha considerado un nuevo atributo en la relación *MADERAS* para poder identificar esta relación, al no presentarse en las especificaciones del problema ningún atributo con valores únicos. Al igual que en las relaciones *PEDIDOS_PROPIOS* y *PEDIDOS*, donde se ha considerado un código de pedido como atributo que identifica cada una de las tuplas.
- En las especificaciones del texto no aparece ninguna opción de borrado o de modificación por lo que se opta por la política de realizar borrados controlados (sin acción) y las modificaciones en cascada. Aunque existen dos excepciones correspondientes a los borrados de las claves ajenas que referencian a *PEDIDOS_PROPIOS* y *PEDIDOS*, que lo normal es que sean en cascada ya que se corresponden con los detalles de los pedidos, que dependen totalmente de estas relaciones.

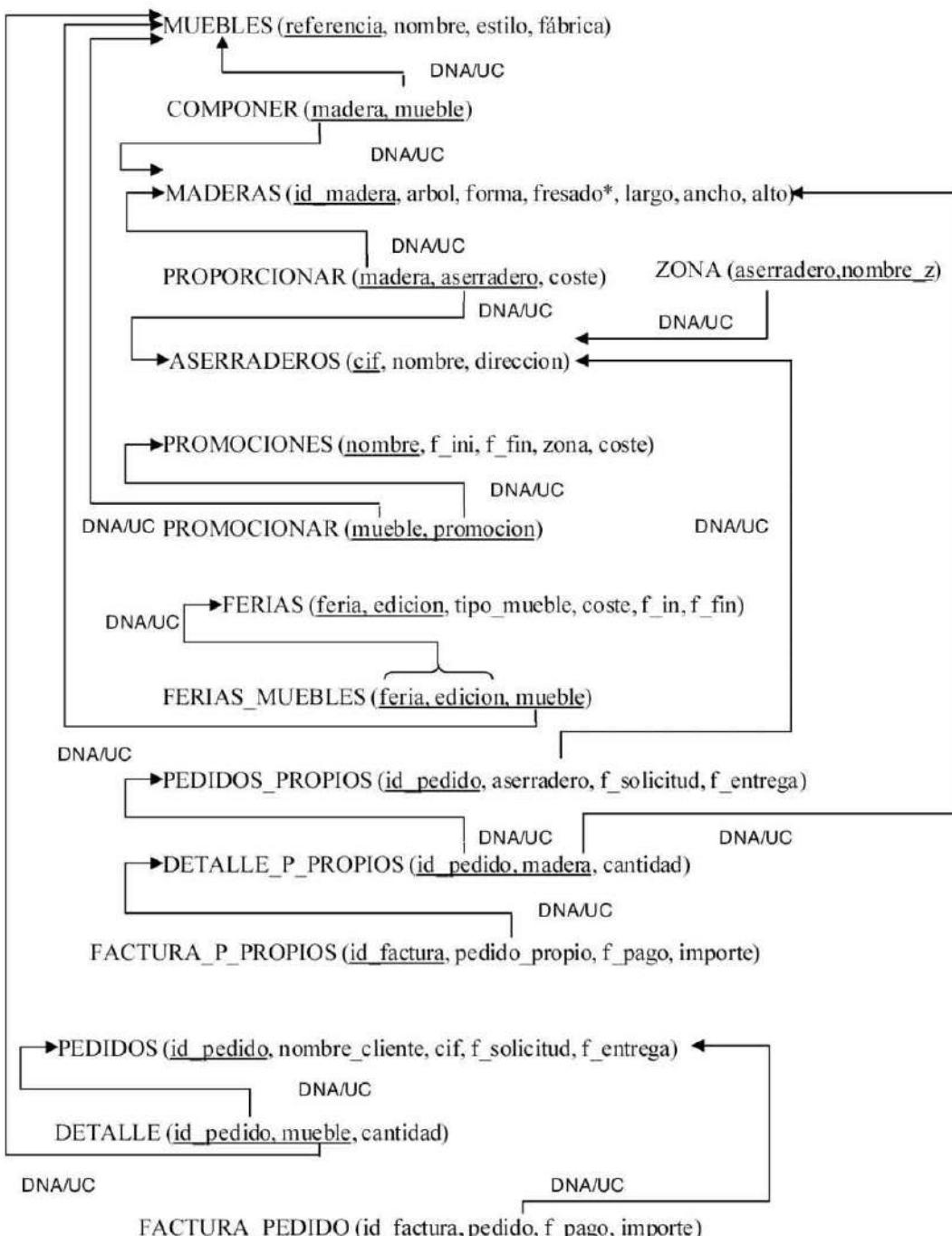


Figura 2.21. Grafo relacional resultante

Los atributos *f_entrega* de las relaciones *PEDIDOS_PROPIOS* y *PEDIDOS* se consideran no obligatorios, para reflejar que pueden existir pedidos no entregados.

- En la relación *ASERRADEROS*, las zonas en las que opera un

aserradero se han modelado con una relación aparte, como requieren los atributos multivaluados. Asegurar que todo aserradero tiene asociada al menos una zona requerirá una aserción adicional que no es posible incluir en el grafo relacional.

Instrucciones SQL para garantizar que en ninguna feria se promocione ningún mueble cuyo estilo no esté en consonancia con el estilo de la feria.

```
CREATE ASSERTION FERIA_MUEBLE  
CHECK NOT EXIST (SELECT 'X' FROM FERIAS A JOIN  
FERIAS_MUEBLES B JOIN MUEBLES C ON (B.mueble=  
C.refencia) WHERE A.tipo_mueble = C.estilo);
```

CONSULTAS

Nombres de las campañas del año 2010.

ÁLGEBRA RELACIONAL

$\Pi_{nombre}(\sigma_{f_ini=2010}(PROMOCIONES))$

SQL

```
SELECT nombre  
FROM PROMOCIONES  
WHERE f_ini = 2010;
```

Referencias de muebles para los que se necesita una madera que nadie provee.

ÁLGEBRA RELACIONAL

$$\prod_{mueble} \left[COMPONER^*_{madera} \left[\prod_{madera} (COMPONER) - \prod_{madera} (PROPORCIONAR) \right] \right]$$

En esta consulta el operador diferencia entre las relaciones *COMPONER* y *PROPORCIONAR* permite obtener las maderas que se necesitan para componer un mueble pero que ningún proveedor está suministrando. El resultado de esta diferencia se combina con la relación *COMPONER* para obtener los muebles que se fabrican con esas maderas.

SQL

```
SELECT mueble FROM COMPONER JOIN (
    SELECT madera
    FROM COMPONER
    EXCEPT
    SELECT madera
    FROM PROPORCIONAR) ;
```

Fábricas donde se producen muebles de pino.

ÁLGEBRA RELACIONAL

$$\prod_{fabrica} [\sigma_{arbol=pino}(MUEBLES *_{madera} COMPONER)]$$

SQL

```
SELECT fabrica  
FROM MUEBLES A JOIN COMPONER B ON (A.referencia = B.mueble)  
WHERE B.arbol = 'pino';
```

Costes totales de promocionar en ferias el mueble de estilo rústico durante el año 2011.

ÁLGEBRA RELACIONAL

$$\prod_{SUM(Costes)} [\sigma_{f_ini='1-1-2011' \text{ AND } f_fin \leq '31-12-2011' \text{ AND } estilo='ristico'}(FERIAS)]$$

SQL

```
SELECT SUM(costes)
FROM FERIAS
WHERE f_in>='1-1-2011' AND f_fin<='31-12-2011'
AND estilo = 'rústico';
```

PROBLEMA 2.6: OBSERVACIÓN DE AVES

Se quiere diseñar una base de datos que contenga información sobre las observaciones realizadas a distintas especies de aves en la Península Ibérica. Para ello se han de considerar las siguientes especificaciones:

Cada especie de ave se identifica por su nombre científico. Se desea conocer su nombre vulgar y una breve descripción de las características más importantes. Es importante también conocer el género al que pertenece cada especie, y junto a éste, la familia y el orden. El género agrupa un conjunto de especies con características estructurales similares y es importante almacenar la información del mismo, así como de la familia y el orden.

Existen asociaciones ornitológicas de las cuales se quiere saber su nombre, la dirección y el teléfono. Cada una de estas asociaciones consta de un grupo de personas, los observadores, que son los que realizan los avistamientos de las distintas especies. De los observadores se requiere su código de observador, el DNI, el nombre completo y la dirección. Además, un observador solo puede estar adscrito a una asociación.

Las zonas de observación tienen un código de zona que es único. También se quiere saber el nombre de la misma, la comunidad autónoma a la que pertenece, la provincia y el tipo (si es una laguna, un río, una zona costera, etc.). En cada zona se encuentra por lo menos un observador, y puede haber varios de distintas asociaciones o de la misma. Cada observador está asignado a una determinada zona.

También se desea guardar información acerca de la fecha en la que el observador divisa una determinada especie, teniendo en cuenta que estos realizan trabajo de campo cada tres días y que además solo anotan el primer avistamiento de una determinada especie por día. Esta información es muy importante para controlar las especies de aves que existen en la Península y, por tanto, tendremos que guardar las observaciones aunque los observadores ya no se encuentren activos.

Se pide:

1. Diseñar el esquema relacional. Indicar en él las claves primarias, claves alternativas y las claves ajenas. De estas últimas especificar los modos de borrado y modificación y los supuestos semánticos que se

han tenido que añadir y aquellos que no se han podido representar.

2. Escribir las sentencias necesarias en SQL para crear el esquema relacional obtenido.
3. Escribir las siguientes consultas en SQL y en álgebra relacional:
 - a) Nombre científico y vulgar de las especies observadas por los observadores pertenecientes a la asociación ornitológica “El Petirrojo”.
 - b) Se desea conocer el nombre y la zona a la que están asignados aquellos observadores activos que no han realizado ninguna observación.

Dicusión del enunciado

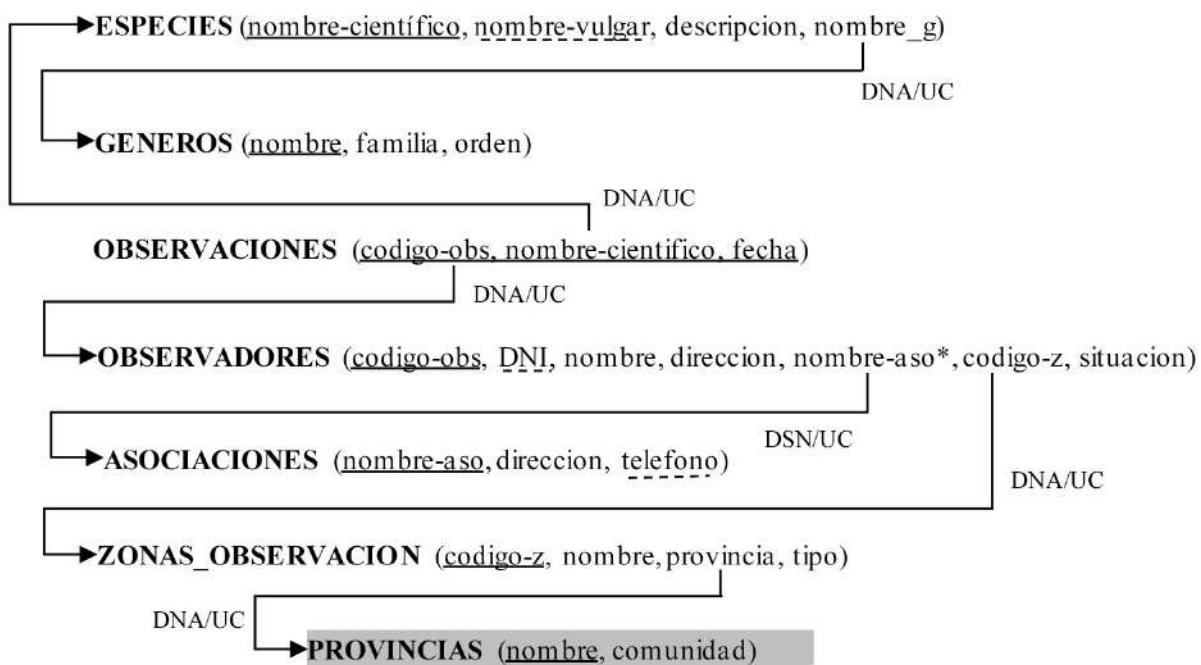


Figura 2.22. Esquema relacional

Los supuestos semánticos que completan el grafo relacional se refieren a las opciones de borrado y modificación de la clave ajena. En el caso de la clave ajena *codigo_g* son sin acción y en cascada, respectivamente (DNA/UC). Al no ser especificadas en los requisitos del dominio se escoge el borrado controlado, de esta manera si se intenta borrar un grupo ornitológico pero este tiene asociadas especies, se debe primero borrar las especies del grupo. Las mismas opciones se han aplicado para el resto de claves ajenas, menos para el *nombre_aso* en la relación *OBSERVADORES*, en la cual se considera el borrado con puestas a nulo.

Existe una restricción que no podemos contemplar en el grafo, que especifica que una zona al menos tiene un observador asignado. En la fase de implementación se podría subsanar definiendo disparadores para la inserción de zonas y observadores de manera conjunta.

SCRIPT EN SQL3

```
CREATE DOMAIN Nombre-Valido  CHAR(60);
CREATE DOMAIN Tipo-Codigo   CHAR(5) NOT NULL;
CREATE DOMAIN Tipo-Situacion CHAR(8)
CHECK (VALUE IN ('Activo', 'Inactivo'));

CREATE TABLE GENEROS (
    nombre      Nombre-Valido,
    familia     Nombre-Valido,
    orden       Nombre-Valido
CONSTRAINT pk-gr-or PRIMARY KEY (nombre);

CREATE TABLE ESPECIES (
    nombre-cientifico  Nombre-Valido,
    nombre-vulgar       Nombre-Valido NOT NULL,
    descripcion        CHAR(100) NOT NULL,
    nombre-g            Tipo-Codigo,
CONSTRAINT pk-especies PRIMARY KEY (nombre-cientifico),
CONSTRAINT uk-especies UNIQUE (nombre-vulgar),
CONSTRAINT fk-especies-generos FOREIGN KEY (nombre-g)
REFERENCES GENEROS ON UPDATE CASCADE);

CREATE TABLE ASOCIACIONES (
    nombre-aso    Nombre-Valido ,
    direccion    CHAR(50) NOT NULL,
    telefono     CHAR(9) NOT NULL,
CONSTRAINT pk-asociaciones PRIMARY KEY (nombre-aso)
CONSTRAINT uk-asociaciones UNIQUE (telefono));
```

```

CREATE TABLE PROVINCIAS (
    nombre      Nombre-Valido,
    comunidad   Nombre-Valido NOT NULL,
CONSTRAINT pk-provincias PRIMARY KEY (nombre)) ;

CREATE TABLE ZONAS_OBSERVACION (
    codigo-z    Tipo-Codigo ,
    nombre       CHAR(15) NOT NULL,
    provincia   CHAR(20) NOT NULL,
    tipo         CHAR(15) NOT NULL,
CONSTRAINT pk-zonas-obs PRIMARY KEY (codigo-z)
CONSTRAINT fk-zonas-observacion-provincias FOREIGN KEY
(provincia) REFERENCES PROVINCIAS ON UPDATE CASCADE) ;

CREATE TABLE OBSERVADORES (
    codigo-obs  Tipo-Codigo
    DNI          CHAR(9) NOT NULL,
    nombre       CHAR(40) NOT NULL,
    direccion   CHAR(50) NOT NULL,
    nombre-aso   Nombre-Valido,
    codigo-z    Tipo-Codigo,
    situacion    Tipo-Situacion NOT NULL,
CONSTRAINT pk-observadores PRIMARY KEY (codigo-obs),
CONSTRAINT uk-observadores UNIQUE (DNI),
CONSTRAINT fk-observadores-asociaciones FOREIGN KEY (nombre-
aso) REFERENCES ASOCIACIONES ON UPDATE CASCADE ON DELETE
SET NULL,
CONSTRAINT fk-observadores-zonas-observacion FOREIGN KEY
(codigo-z) REFERENCES ZONAS_OBSERVACION ON UPDATE CASCADE) ;

CREATE TABLE OBSERVACIONES (
    codigo-obs      Tipo-Codigo,
    nombre-cientifico Nombre-Valido,
    fecha           DATE,
CONSTRAINT pk-observaciones PRIMARY KEY (codigo-obs, nombre-
cientifico, fecha),
CONSTRAINT fk-observaciones-observadores FOREIGN KEY (codigo-
obs) REFERENCES OBSERVADORES ON UPDATE CASCADE,
CONSTRAINT fk-observaciones-especies FOREIGN KEY (nombre-
cientifico) REFERENCES ESPECIES ON UPDATE CASCADE);

```

CONSULTAS

Nombre científico y vulgar de las especies observadas por los observadores pertenecientes a la asociación ornitológica “El Petirrojo”.

ÁLGEBRA RELACIONAL

$$\begin{aligned} & \Pi_{\text{nombre-científico}, \text{nombre-valor}} (\text{ESPECIES}) * \Pi_{\text{nombre-científico}} \\ & \left[(\text{OBSERVACIONES}) \right. \\ & \left. * \Pi_{\text{codigo-obs}} (\sigma_{\text{nombre-aso}='Elpetirrojo'} (\text{OBSERVADORES})) \right] \end{aligned}$$

SQL

```
SELECT DISTINCT nombre_cientifico, nombre_vulgar  
  
FROM    ESPECIES      JOIN    (SELECT    nombre_cientifico    FROM  
OBSERVACIONES JOIN (SELECT codigo-obs FROM OBSERVADORES WHERE  
nombre_aso= 'El Petirrojo'));
```

Se desea conocer el nombre y la zona a la que están asignados aquellos observadores activos que no han realizado ninguna observación.

ÁLGEBRA RELACIONAL

$$\Pi_{\text{codigo-obs}, \text{nombre}, \text{codigo-z}} \left[\begin{array}{l} (\text{OBSERVADORES})^* \\ \left[\begin{array}{l} \Pi_{\text{codigo-obs}} (\sigma_{\text{situacion}=\text{'activo'}} (\text{OBSERVADORES})) - \\ \Pi_{\text{codigo-obs}} (\text{OBSERVACIONES}) \end{array} \right] \end{array} \right]$$

SQL

```
SELECT nombre, codigo-zona  
FROM OBSERVADORES JOIN (SELECT codigo-obs FROM OBSERVADORES  
WHERE situacion= 'activo')  
EXCEPT SELECT codigo-obs FROM OBSERVACIONES) ;
```

PROBLEMA 2.7: GESTIÓN DE EMPRESA DE OCIO

“GESTIONOCIO” es una empresa que pretende estructurar la información de los locales de moda para comer y poder ofrecer cada mes los diez mejores sitios para degustar una buena comida en Madrid.

La información que necesitamos de cada local es la siguiente: nombre que identifica cada local, dirección, zona, teléfono para realizar reservas, estilo de comida, dirección web del local (si posee) y si es posible acceder al local a través del metro, recogiendo para ello la estación o estaciones cercanas al local y sus líneas correspondientes.

También es necesario saber la persona o empresa que es responsable del local y el jefe de cocina o chef del mismo. Para ello, sabemos que es posible que una empresa o persona gestione varios locales en Madrid aunque el jefe de cocina es único en cada local. De los gestores del local guardaremos el nombre, la dirección, CIF o NIF y *e-mail* si tiene. Si los gestores de un local cambian, el nombre del local cambiará y por tanto tendremos un nuevo local.

El chef puede cambiar a lo largo del tiempo e incluso puede que exista un periodo de tiempo en el que este dato se desconozca para un determinado local. De él se guardará su nombre completo, descripción de las características de su cocina, y locales en los que trabajó anteriormente que siguen abiertos en la actualidad.

La forma de acceso a un determinado local en transporte público, concretamente por la línea de metro, se especifica por un número línea, su color y el nombre de la estación. Puede ocurrir que para un local existan varias líneas de metro por las que se pueda acceder a la misma estación o que tengamos distintas estaciones que se encuentren cerca de este local. Por lo que en este caso, tendremos que indicar la cercanía de cada una de ellas a través de una puntuación. Si solo existe una manera de acceder al local pondremos la puntuación máxima (de 5 por defecto).

Para poder sacar cada mes los *top ten*, la empresa GESTIONOCIO contrata a personas especializadas que recorren los locales como si fuesen clientes, degustan los productos del local y realizan un cuestionario acerca del mismo. Cada catador se identifica por su DNI, y se guarda además un teléfono móvil dado por la empresa para poder localizarle en cualquier momento, su nombre

completo y una dirección. Un local podrá recibir varias visitas de distintos catadores. Las opiniones de un catador acerca de un local no pueden ser borradas ni modificadas. Ahora bien si el local desaparece sí es necesario borrar todas las calificaciones asociadas al mismo.

Los cuestionarios que tienen que llenar por cada local visitado se identifican por el identificador del catador y el número de visita a ese local. En él se puntuá el trato del personal, la tardanza en servir la comida y la calidad de la misma. Esta puntuación tiene un rango de 0 a 10, de menor a mayor a calidad.

Se pide:

1. Grafo relacional que describa todas estas necesidades.
2. Describe en Álgebra Relacional y en SQL las siguientes consultas:
 - a) Locales cuya cercanía a una estación de metro tiene una puntuación menor que 3.
 - b) Nombre, dirección y zona de los locales cuya media en la calidad de la comida supero el 5.
 - c) Locales que todavía no han sido visitados por ningún catador.

Discusión del enunciado

Supuestos que no se han podido reflejar en el grafo relacional:

- Los atributos *personal servicios calidad* pertenecientes a la relación *CUESTIONARIOS* solo pueden tomar valores entre 0 y 10.
- ...*Si solo existe una manera de acceder al local pondremos la puntuación máxima (de 5 por defecto)...* Para reflejar este supuesto debemos crear un disparador asociado a la relación *TRANSPORTES*.
- El siguiente supuesto semántico *Las opiniones de un catador acerca de un local no pueden ser borradas ni modificadas* no puede contemplarse en el grafo relacional.

Supuestos añadidos por el diseñador:

- La relación *EXPERIENCIA* tiene como clave primaria los atributos *local, f_ini* al considerar que en un momento determinado en un local solo puede haber un chef asignado. Pero se permite que un mismo chef haya estado en un mismo local en periodos distintos. También se define como clave alternativa *f_ini, chef* puesto que en un periodo de tiempo un chef no puede estar en dos locales simultáneamente.

Para la relación *CATADORES* se incluye como clave alternativa el atributo *tel_movil* de esta forma un catador siempre estará localizable e identificado por este número de teléfono.

Al considerar el atributo *jefe_cocina* no obligatorio no podemos definir que sea clave alternativa y ajena en *LOCALES*, ya que una clave alternativa no puede admitir valores nulos. Este es el motivo de haber considerado una nueva relación *CONTRATOS*.

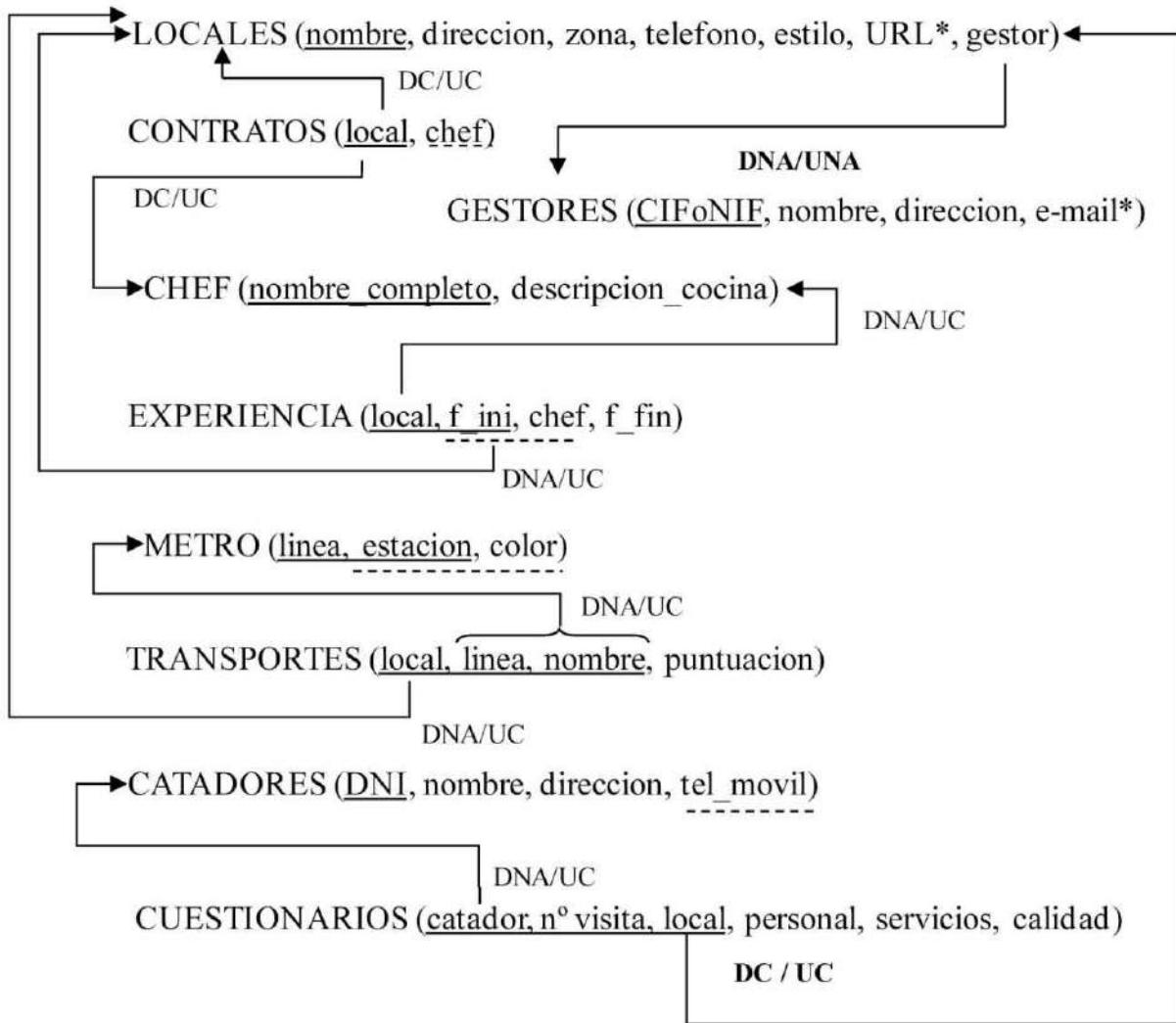


Figura 2.23. Grafo relacional propuesto

CONSULTAS

Locales cuya cercanía a una estación de metro tiene una puntuación menor que 3.

ÁLGEBRA RELACIONAL

$$\Pi_{local, estacion} (\sigma_{puntuacion < 3} (TRANSPORTES))$$

SQL

```
SELECT local, estacion  
FROM TRANSPORTES WHERE puntuacion > 3;
```

Nombre, dirección y zona de los locales cuya media en la calidad de la comida superó el 5.

ALGEBRA RELACIONAL

$$\begin{aligned} & \Pi_{\text{nombre}, \text{direccion}, \text{zona}} \\ & \left[\Pi_{\text{nombre}, \text{direccion}, \text{zona}} (\text{LOCALES})^*_{\text{nombre}=\text{local}} \right. \\ & \left. \Pi_{\text{local}} (\sigma_{\text{AVG(calidad)}>5} (\text{GROUP_BY}_{\text{local}} (\text{CUESTIONARIOS}))) \right] \end{aligned}$$

SQL

```
SELECT nombre, direccion, zona
FROM (SELECT nombre, direccion, zona FROM LOCALES) A
JOIN
    (SELECT local FROM CUESTIONARIOS
     GROUP BY local HAVING AVG (calidad) > 5) B
ON (A.nombre=B.nombre);
```

Locales que todavía no han sido visitados por ningún catador.

ALGEBRA RELACIONAL

$\Pi_{\text{nombre}}(\text{LOCALES}) - \Pi_{\text{local}}(\text{CUESTIONARIOS})$

SQL

```
SELECT nombre FROM LOCALES  
EXCEPT  
SELECT local FROM CUESTIONARIOS;
```

PROBLEMA 2.8: GESTIÓN DE INCENDIOS

La Comunidad de Madrid preocupada por los incendios forestales quiere crear una red informática entre todos los puntos creados para el control de estos incendios de una forma rápida y efectiva. Para ello ha pensado en el diseño de una base de datos relacional que contenga la siguiente información.

La Comunidad de Madrid está dividida en zonas de riesgo de incendios que se distinguen por lo que se denomina punto geodésico (cadena de 6 dígitos que identifica cada zona). También se desea almacenar la frecuencia de radio por la que emite cada zona y la latitud y longitud tomadas por el sistema GSM. La latitud y la longitud definen unas coordenadas que son únicas para cada zona.

Cada zona, dependiendo de su extensión, está compuesta de varios puestos de control. Estos puestos son gestionados por un guarda forestal, el cual es el responsable de que exista una persona 24 horas vigilando el bosque. Cada puesto de control está identificado por un nombre y el punto geodésico de la zona a la que pertenece, pero también se desea conocer otras características como el responsable del puesto, frecuencia de radio en la que se encuentra (si es distinta de la asignada a la zona).

Los guardas forestales están asignados a uno o varios puestos forestales pero únicamente a una zona de riesgo. Eso sí, en un determinado momento solo podrán estar en un puesto de control. De ellos se requiere el nombre, el DNI, el teléfono, la dirección y su antigüedad en años (el primer año de servicio tendrá una antigüedad igual cero).

Por último, para que el sistema funcione, un conjunto de cuerpos de bomberos se encuentran distribuidos por distintos puntos y acudirán si existe una emergencia a cualquiera de las zonas en las que está dividida la comunidad. El radio de actuación de un determinado cuerpo de bomberos puede cubrir varios puestos de control de distintas zonas por lo que se almacenará un atributo de prioridad (1..10) para indicar, de menor a mayor, el orden de actuación si existen dos incendios simultáneos en la misma zona. De cada uno de los cuerpos de bomberos se requiere un código de cuerpo, dos teléfonos de emergencia y los efectivos de los que dispone divididos en: número de hombres, número de camiones cisternas y helicópteros. Si existe alguna unidad de bomberos que desaparezca toda la información asociada a los mismos debe desaparecer de la base de datos.

Tanto si algún puesto forestal como cuerpo de bomberos desaparece se deberá eliminar toda la información referente a ellos.

Se pide:

1. Grafo relacional del esquema que describa los supuestos semánticos anteriormente descritos, especificando qué semántica no se ha podido reflejar y comentarios al diseño.
2. Describir en Álgebra Relacional:
 - a) Números de teléfono y código de los cuerpos de bombero asignados a la zona “123456” ordenados por el atributo prioridad de menor a mayor.
 - b) Nombre, dirección y teléfono de los guardas forestales asignados a la zona “123456” en cada uno de los puestos de control, el día “11-08- 2004”.
 - c) Número total de helicópteros, bomberos y cisternas disponibles en cada una de las zonas en las que se divide la Comunidad de Madrid.

Discusión del enunciado

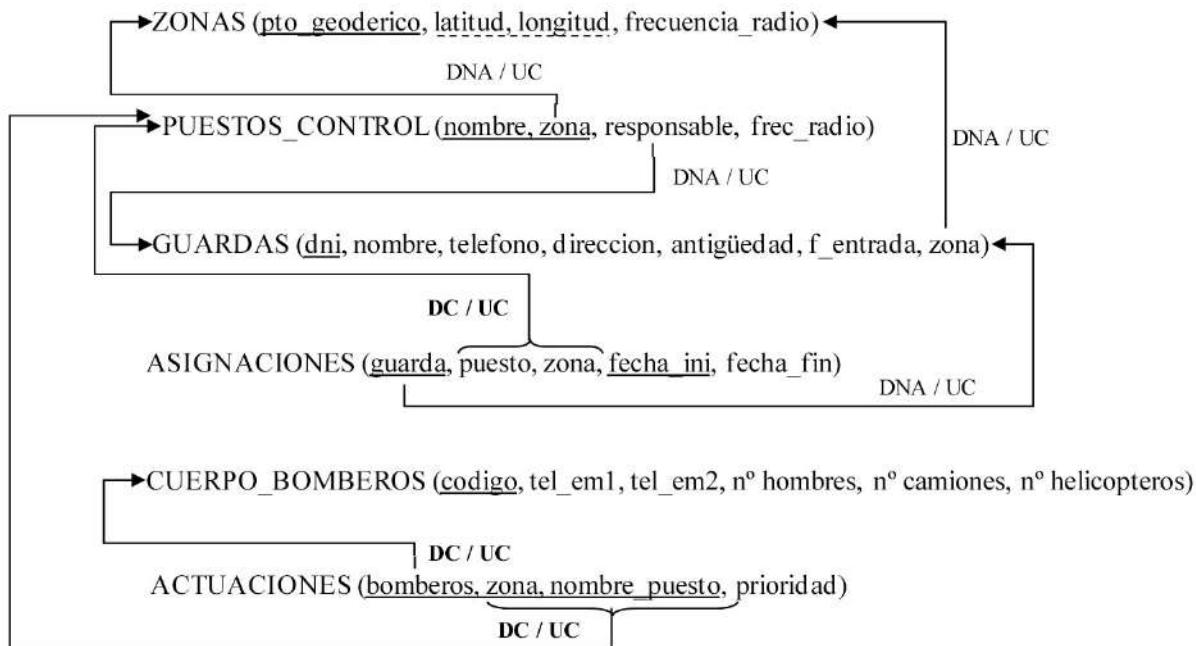


Figura 2.24. Esquema relacional

La semántica indicada en las especificaciones del problema que no se ha podido reflejar se comenta a continuación:

1. La frecuencia de radio de un puesto de control, si no tiene una propia, es la de su zona.
2. La relación **ASIGNACIONES** se crea para conocer las asignaciones de los guardas forestales a cada puesto forestal, pero no se puede contemplar que estos están asignados a una única zona.
3. El atributo *prioridad* en la relación **ACTUACIONES** es un rango entre 1 y 10.

La semántica que se ha supuesto es la siguiente:

1. Se ha añadido un atributo, que no aparece en las especificaciones, en la tabla **GUARDAS** denominado *f Entrada*. Puesto que de esta forma es fácil calcular el atributo *antigüedad* y, por tanto, no será un atributo que se actualice manualmente sino que tendrá que llevar un

mecanismo para ser calculado.

2. Las opciones de borrado y modificación que no aparecen especificadas en el texto se han tenido que suponer. Para el borrado se ha considerado la opción de no acción y para la modificación en cascada.

CONSULTAS

Números de teléfono y código de los cuerpos de bombero asignados a la zona “123456” ordenados por el atributo prioridad de menor a mayor.

$$\Pi_{codigo,tel_em1,tel_em2} \left(\sigma_{zona='123456'} (ACTUACIONES) *_{codigo=bomberos} \Pi_{codigo,tel_em1,tel_em2} (CUERPO_BOMBEROS) \right)$$

Nombre, dirección y teléfono de los guardas forestales asignados a la zona “123456” en cada uno de los puestos de control, el día “11-08-2004”.

$$\Pi_{nombre,direccion,telefono} \left(GUARDAS *_{dni=guarda} \sigma_{zona='123456' and f_ini='11/08/2004'} (ASIGNACIONES) \right)$$

Número total de helicópteros, bomberos y cisternas disponibles en cada una de las zonas en las que se divide la Comunidad de Madrid.

$$\Pi_{codigo,SUM(n^{\circ}hom\ bre),SUM(n^{\circ}camiones),SUM(n^{\circ}helicopteros)} \left[GROUP_BY_{zona} \left(\begin{array}{l} \Pi_{zona,bomberos} (ACTUACIONES) *_{bomberos=codigo} \\ \Pi_{codigo,n^{\circ}hom\ bre,n^{\circ}camiones,n^{\circ}helicopteros} (CUERPO_BOMBEROS) \end{array} \right) \right]$$

PROBLEMA 2.9: GRAN PREMIO DE FÓRMULA 1

La organización del Gran Premio de Fórmula 1 requiere de ayuda de expertos diseñadores para gestionar la información asociada a este deporte que cada vez es más popular. Para ello nos han proporcionado la información básica acerca de los circuitos, los pilotos y las escuderías que están actualmente participando en el mundial 2006 y que ya se han diseñado como relaciones dentro de la base de datos relacional (que solo recogerá información de un año) que dará soporte a la organización. A continuación son mostradas:

CIRCUITOS (nombre, año_fundacion, web, longitud, nº_curvas, velocidad_max)

ESCUDERIAS (nombre, jefe, director, tipo_coche)

PILOTOS (nombre, f_nacimiento, lugar_nacimiento, dorsal, puntos)

A partir de estas relaciones, se pide completar este esquema relacional inicial con las especificaciones que a continuación pasamos a describir y que fueron recogidas en diversas entrevistas con la organización.

Tanto el nombre de las escuderías como de los pilotos y los circuitos son únicos. El resto de atributos especificados en estas relaciones son siempre conocidos y obligatorios.

Las escuderías son generalmente identificadas, además de por su nombre, por su jefe de escudería ya que son personas muy influyentes en el mundo de los negocios y es una manera de publicitarlas. Además, la escudería está formada por pilotos (uno al menos). Lo que nunca se puede dar es que un piloto cambie de escudería durante el campeonato y mucho menos que una escudería cambie o desaparezca durante el mismo. Los pilotos solo pueden pertenecer a una escudería y existen algunos pilotos que tienen compañías que les patrocinan.

Una compañía puede patrocinar a varios pilotos y se quiere saber por cuanto asciende (en euros) su contrato con cada uno de ellos. Las compañías se identifican por un CIF pero también se informa acerca del nombre y el eslogan, si lo tienen, que promociona (independientemente del piloto o pilotos que lo hagan).

Por último, lo más importante del gran premio es reflejar las carreras que se han disputado y los participantes en cada una de ella. Se sabe que una carrera se

realiza en un determinado día y circuito. Es decir, en un circuito solo se corre una carrera del gran premio y en una fecha solo puede haber una carrera. Además, la información más relevante en cada una de ellas son los pilotos que han participado, su posición tanto inicial como final y el tiempo realizado (si terminan la carrera).

Se pide:

1. El grafo relacional resultante, teniendo en cuenta que solo se debe gestionar la información de las carreras realizadas en el Premio de Fórmula 1 del 2006, se muestra en la Figura 2.25.
2. Realizar las siguientes consultas Álgebra Relacional y en SQL.
 - a) Para cada carrera, indicar el nombre del piloto con peor tiempo y la escudería a la que pertenece.
 - b) Nombre de los pilotos que no han participado en la carrera del 26-05-2006.
 - c) La *pole position* es conseguida por el piloto que mejor tiempo ha realizado en los entrenamientos y lo que significa, es que el piloto que la consigue se coloca el primero en la línea de salida. Por cada carrera, queremos conocer qué piloto obtuvo la *pole* y el circuito.

Discusión del enunciado

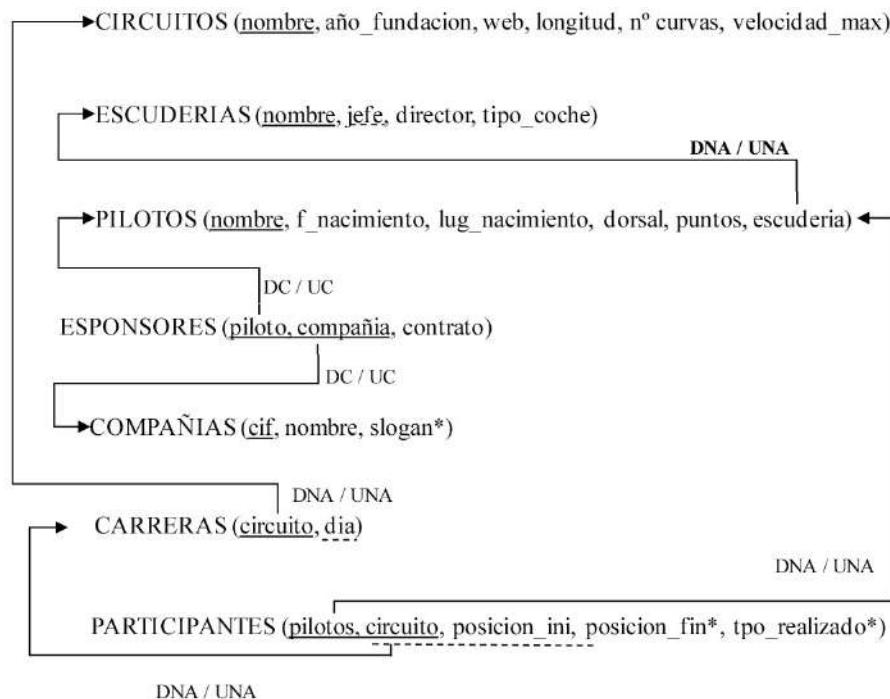


Figura 2.25. Grafo relacional resultante

Supuestos semánticos no recogidos en el grafo:

- ...Además, la escudería está formada por pilotos (uno al menos)... En fase de implementación podría controlarse a través de disparadores de inserción, borrado y modificación en la relación *ESCUADERIAS*.
- ...Lo que nunca se puede dar es que un piloto cambie de escudería durante el campeonato... Esta restricción se podría controlar a través de un disparador de modificación para la tabla *PILOTOS*, el cual rechace la operación de modificación, siempre que se trate de la modificación del atributo *escuderia*.

Supuestos semánticos añadidos por el diseñador:

- Las opciones de borrado y modificación que dependen de las carreras y la participación de los pilotos en las mismas son restringidas (no acción).

- El resto de las opciones de borrado y modificación se han elegido en cascada.

CONSULTAS

Para cada carrera, indicar el nombre del piloto con peor tiempo y la escudería a la que pertenece.

ÁLGEBRA RELACIONAL

$$(\Pi_{circuito, MIN(tpo_realizado)} [GROUP_BY_{circuito} (PARTICIPANTES)]) \rightarrow A$$

A indica por cada carrera el tiempo mínimo realizado.

$$\Pi_{circuito, piloto} \left(A^* \Pi_{pilotos, circuitos, tpo_realizado} (PARTICIPANTES) \right)$$

SQL

```
SELECT piloto, circuito  
FROM PARTICIPANTES A JOIN  
(SELECT circuito, MIN(tpo_realizado) tiempo  
FROM PARTICIPANTES GROUP BY circuito) ON  
(A.circuito=B.circuito AND A.tpo_realizado=B.tiempo);
```

Nombre de los pilotos que no han participado en la carrera del 26-05-2006.

ÁLGEBRA RELACIONAL

$$\Pi_{\text{nombre}}(\text{PILOTOS}) - \\ [\Pi_{\text{pilotos}}(\text{PARTICIPANTES} * \Pi_{\text{circuito}}(\sigma_{\text{dia}=26-05-2006}(\text{CARRERAS})))]$$

SQL

```
SELECT nombre FROM PILOTOS  
EXCEPT  
SELECT piloto FROM PARTICIPANTES JOIN  
(SELECT circuito FROM CARRERA WHERE dia='26-05-2006') ;
```

La *pole position* es conseguida por el piloto que mejor tiempo ha realizado en los entrenamientos y lo que significa, es que el piloto que la consigue se coloca el primero en la línea de salida. Por cada carrera, queremos conocer qué piloto obtuvo la *pole* y el circuito.

ÁLGEBRA RELACIONAL

$$\Pi_{circuito, pilotos} (\sigma_{posicion_ini=1} (PARTICIPAN\ TES))$$

SQL

```
SELECT    circuito,    pilotos      FROM      PARTICIPANTES      WHERE
posicion_ini = 1;
```

PROBLEMA 2.10: FEDERACIÓN DE TAXIS

La Federación de taxis “Fastcab” precisa organizar la información referente a su negocio. En esta federación, existen varios propietarios que poseen al menos una licencia de taxi (asociada a un taxi aunque después se puede modificar). Cada taxi tenía asociado un conductor, aunque esta condición se ha detectado que debe cambiar. El esquema relacional se describe mediante el grafo relacional de la Figura 2.26.

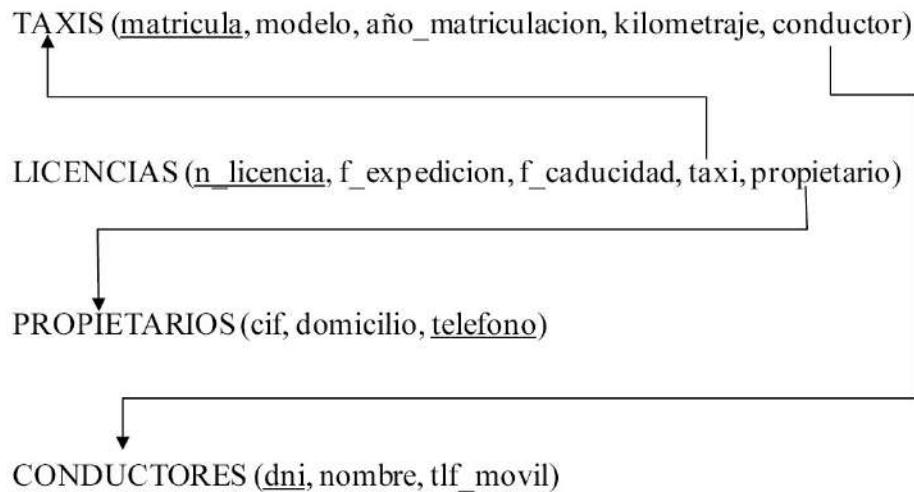


Figura 2.26. Grafo relacional inicial para el problema de la federación de taxis

Se pide actualizar este grafo para reflejar las nuevas necesidades, en caso de que esto sea posible.

- Un taxi no tiene asociado un único conductor, sino que cualquier conductor puede conducir cualquier taxi (no existen esas asociaciones, ni es necesario información del conductor).
- Cada taxi se caracteriza por un modelo. Un modelo de coche (o taxi) se identifica por la marca y el modelo, y posee las siguientes propiedades: cilindrada, potencia, consumo y tipo de combustible.
- El tipo de combustible es un atributo que solo puede tomar los valores: *gasoil*, *95*, *98* y *súper*.
- El combustible lo proporcionan proveedores de los que se quiere almacenar el nombre de la empresa, dirección y tipos de combustibles

que suministran.

- e) Una empresa de combustible puede tener varias direcciones (dos proveedores pueden pertenecer a la misma empresa de combustible pero siempre estarán en distintas direcciones).
- f) Cada taxi solo puede realizar un repostaje diario. Los repostajes los realizan taxis en una fecha en un proveedor.
- g) El repostaje debe ser del tipo de combustible que admite el taxi.
- h) No pueden desaparecer modelos de los que se tenga algún taxi.
- i) Los repostajes deben permanecer almacenados, incluso si ya no está almacenado el taxi que lo efectuó.
- j) Si cambia la dirección del proveedor en un repostaje, se debe actualizar esta información en el registro del proveedor.

Discusión del enunciado

Las especificaciones que no se han podido especificar son: c) que se podrá recoger con un *CHECK* en la fase de implementación, g) para poder comprobarla en la fase de implementación sí es necesario almacenar el tipo de combustible que se le ha suministrado al taxi durante su repostaje, i) al ser el atributo *taxis* no solo clave ajena sino clave primaria de la relación *REPOSTAJES* las únicas opciones de borrado que se pueden elegir son sin acción o en cascada y por último, j) para recogerla en la fase de implementación se podría asociar un disparador para el evento modificación en el atributo *dirección* que reflejara este cambio en la tabla *PROVEEDORES*.

Para finalizar se quiere hacer notar que el grafo que aparece en la Figura 2.27 es el resultado de las modificaciones sufridas para recoger las especificaciones que nos pide el problema pero para completarlo se debe incluir las opciones de borrado y modificación que no están explícitas en el enunciado (y, por lo tanto, ausentes en el grafo 2.27). Se adoptará el modo menos restrictivo y más seguro, esto es los borrados no acción y las modificaciones en cascada.

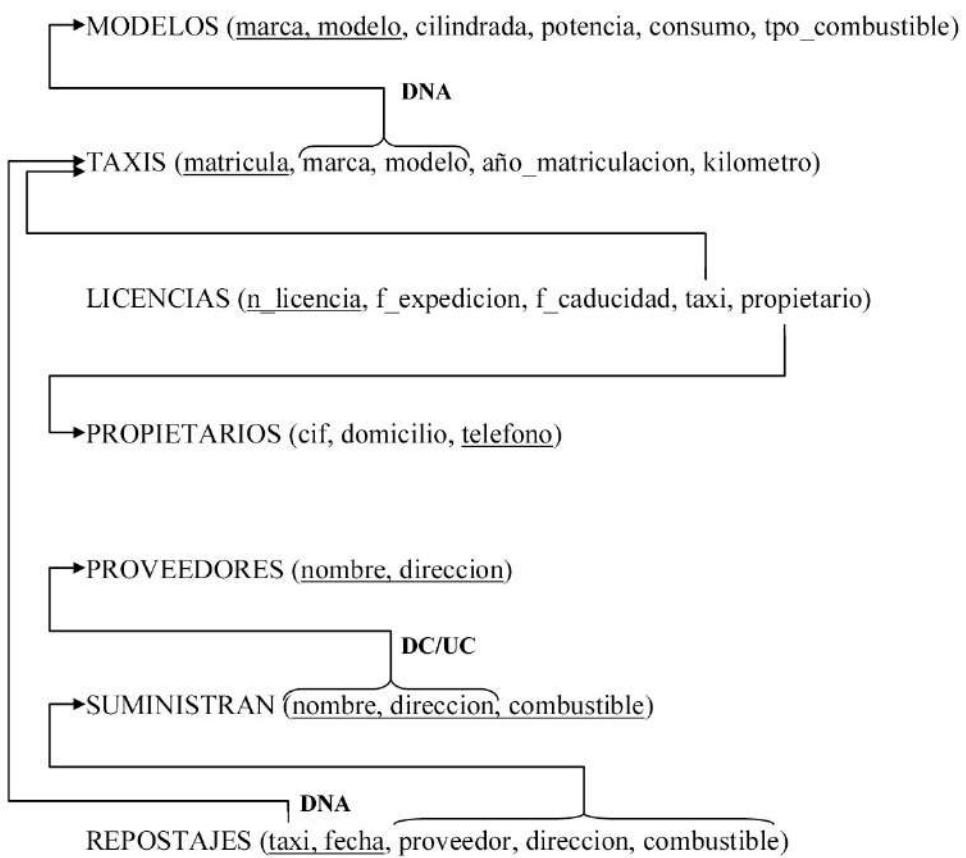


Figura 2.27. Grafo relacional resultante

PROBLEMA 2.11: VIAJES DE INVESTIGACIÓN

Dado el esquema conceptual formado por el diagrama E/R y los supuestos correspondientes a la solución del Problema 1.6 del Capítulo 1 donde se almacena la información relativa a los gastos de los viajes de investigación de los profesores de la universidad, se pide transformar el esquema E/R en su esquema lógico (grafo relacional), representando toda la semántica que no se pueda recoger en el esquema lógico en forma de *checks*, aserciones y disparadores.

DISCUSIÓN DEL ENUNCIADO

Cada entidad del diagrama E/R se transforma en una relación; los atributos univaluados y simples se transforman en columnas de la relación; los atributos compuestos se descomponen y se transforman en distintas columnas de la relación; cada Atributo Identificador Principal (AIP) de cada entidad se convertirá en la clave primaria de la relación; cada Atributo Identificador Alternativo (AIA) se transforma en clave alternativa en la relación.

En cuanto a otro tipo de constructores del modelo E/R (atributos multivaluados, interrelaciones, jerarquías, etc.), su transformación se detalla a continuación, especificando la semántica que se pierde en la transformación al grafo relacional mediante *checks*, aserciones y disparadores.

PARTE 1

En primer lugar, se transforma el esquema E/R asociado al diagrama mostrado en la Figura 2.28:

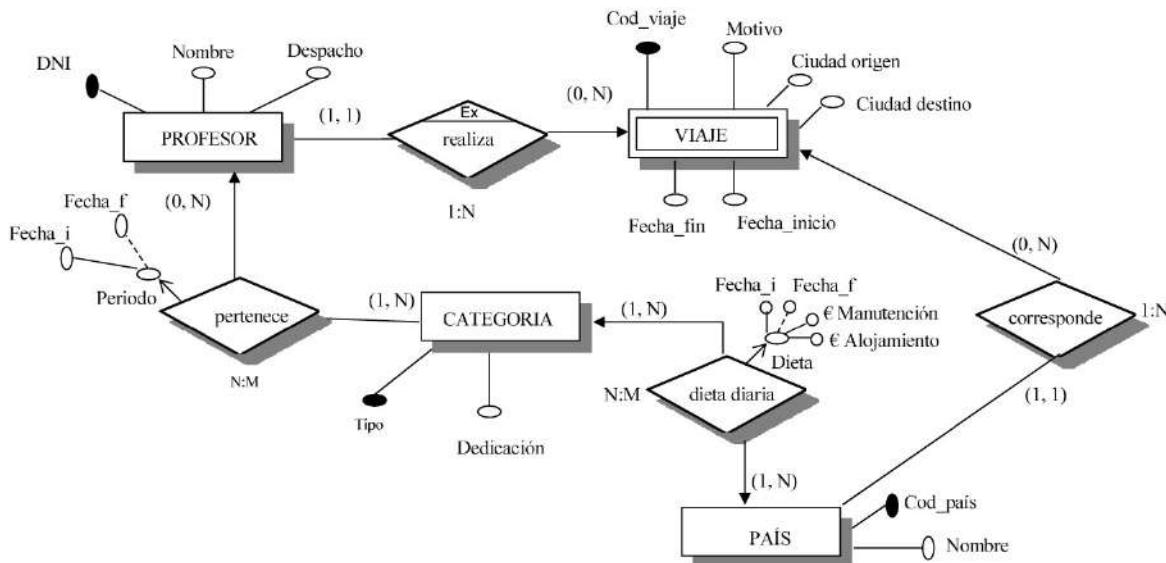


Figura 2.28. Esquema E/R inicial sobre gestión de viajes en la universidad

La interrelación **Realiza** es una interrelación 1:N, por lo que se transforma mediante una propagación de clave: la clave primaria de la relación *PROFESORES* pasará a formar parte de la relación *VIAJES* como columna, tratándose como una clave ajena que referencia a la relación *PROFESORES*.

Como la interrelación es de dependencia en existencia, se tendrá en cuenta en el grafo relacional que si se borrara una ocurrencia de la relación *PROFESORES*, todas las ocurrencias de la relación *VIAJES* relativas al profesor borrado, no tendrían sentido en nuestra base de datos. Por ello, las restricciones de integridad referencial en este caso se tratarán como borrado y modificación de la clave ajena en cascada (DC/UC). Al transformar esta interrelación no se produce pérdida semántica debida a las cardinalidades.

La interrelación **Corresponde** también es una interrelación 1:N y, de la misma forma, se realiza una propagación de clave. La clave primaria de la relación *PAISES* se propaga a la relación *VIAJES* como clave ajena. Las restricciones de integridad referencial de esta clave ajena se consideran como borrado sin acción y modificación en cascada (DNA/UC). En este caso tampoco se pierde semántica en la transformación de la interrelación.

La interrelación **Pertenece** también es una interrelación 1:N pero, debido a la existencia de un atributo en la interrelación, nos planteamos la posibilidad de transformar la interrelación en una nueva relación del grafo relacional. Esta relación estará formada por la clave primaria de las relaciones que une (tratadas como claves ajenas en esta relación), así como por los atributos *Fecha_inicio* y *Fecha_fin*, este último opcional. La clave primaria de la nueva relación *PERTENECEN* estará formada por el DNI del profesor (denominado en el grafo relacional como *Profesor*) y la *Fecha_inicio* del contrato en esa categoría, pues con ello quedan caracterizadas, de forma única y minimal, todas las tuplas de la relación. También se podría tener en cuenta como posible clave alternativa la unión de los atributos *Profesor* y *Fecha_fin* que, aunque *Fecha_fin* es un atributo opcional, se tendrá en cuenta que un profesor no puede tener a nulo dos fechas de finalización de contrato (se supone que solo será nula la fecha de finalización del contrato actual), por lo que la *Fecha_inicio* y la *Categoría* del profesor pueden ser implicados por esta conjunción de atributos. Las restricciones de integridad referencial para todas las claves ajenas de esta relación se consideran como borrado sin acción y modificación en cascada (DNA/UC). En la transformación de esta relación tampoco se ha producido pérdida de semántica.

NOTA: la *Categoría* del profesor no formará parte de la clave primaria de la relación *PERTENECEN*, ya que un profesor con una única fecha de inicio (o fecha de finalización) del contrato solo puede estar asociado a una única categoría contractual.

Para finalizar con el esquema E/R de la Figura 2.28, la interrelación N:M **Dieta_Diaria** se transforma como una nueva relación *DIETAS_DIARIAS*

formada por las claves primarias de las relaciones que une y los atributos del atributo compuesto *Dieta*. La clave primaria de la relación estará formada por la *categoría* de contrato (clave ajena respecto a la relación *CATEGORÍA*), el *país* destino del viaje (clave ajena respecto a la relación *PAÍS*) y la *fecha_inicio* de las dietas, ya que existirá una única *fecha_fin*, *€_manutención* y *€_alojamiento* por cada ocurrencia de estos tres valores. De igual forma que en la relación *PERTENESEN*, las columnas *categoría*, *país* y *fecha_fin* conforman una posible clave alternativa para la relación. La columna *fecha_fin* de esta relación también será un campo opcional.

En cuanto a la semántica perdida en la transformación del esquema E/R al esquema relacional, habrá que tener en cuenta que siempre existirá al menos información sobre *dietas_diarias* de cada país y de cada categoría. Es decir, en la relación *DIETAS_DIARIAS* será necesario comprobar que hay información de todos los países y de todas las categorías. Para ello, será necesaria una aserción que compruebe que se cumple esta semántica en todo momento.

La Figura 2.29 muestra el grafo relacional inicial de la semántica comentada anteriormente:

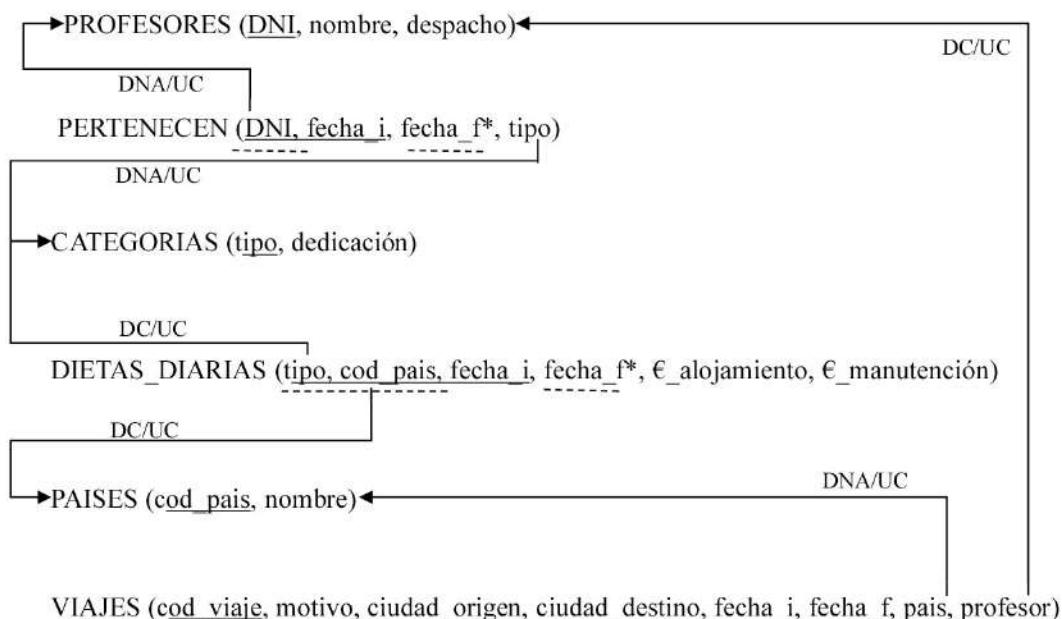


Figura 2.29. Grafo relacional inicial sobre gestión de viajes en la universidad

PARTE 2

La Figura 2.30 muestra el diagrama E/R de los supuestos semánticos

relativos a los gastos de los viajes.

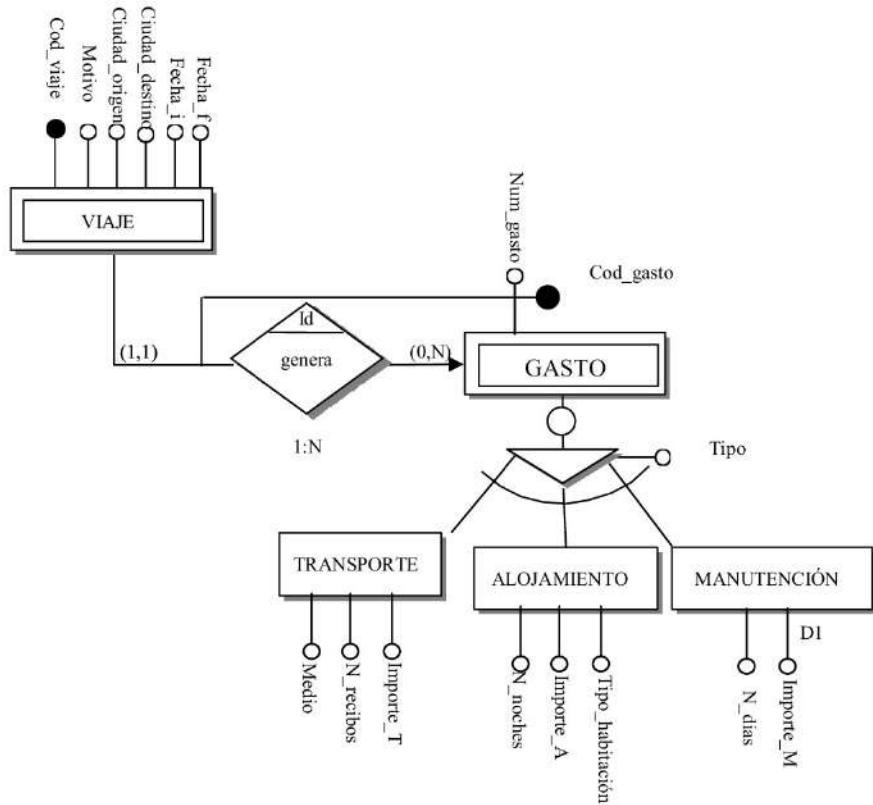


Figura 2.30. Diagrama E/R sobre gastos de viaje

La jerarquía **GASTO** da lugar a cuatro nuevas relaciones, una por cada entidad de la jerarquía. La relación **GASTOS** se identifica por la clave primaria de la relación **VIAJES** unida a un número de secuencia de gastos, según se muestra en la Figura 2.30, dado que existe una dependencia en identificación entre las dos entidades. Las opciones de borrado y modificación de la clave ajena de **GASTOS** se consideran en cascada (DC/UC), debido a la fuerte dependencia que existe respecto a la relación **VIAJES** (no tiene sentido hablar de gastos si no están relacionados con algún viaje).

Por otro lado, no hay que olvidar que el atributo discriminante de la jerarquía pasa a formar parte como columna en la relación proveniente del supertipo de la misma, es decir, el atributo *Tipo* de la jerarquía pasará a formar parte de la relación **GASTOS** como columna. Como se trata de una jerarquía total, esta columna no podrá tomar valores nulos (no será opcional). Además, como se trata de una jerarquía exclusiva, la columna *tipo* solo puede tomar valores simples de los subtipos de la jerarquía (por ejemplo, los valores *transporte*, *alojamiento* o *manutencion*), que habrá que comprobarlo mediante un *check* debido a que esta

semántica será imposible de almacenar en el grafo relacional. Por último, como la totalidad implica que la unión de todas las tuplas de los subtipos tienen que estar en el supertipo, se necesitaría una aserción para su comprobación.

Las relaciones *TRANSPORTES*, *ALOJAMIENTOS* y *MANUTENCIONES*, comparten la misma clave primaria, consideradas como claves ajenas referenciando a la relación *GASTOS*, como muestra la Figura 4. En este caso, las opciones de borrado y modificación de las claves ajenas se consideran en cascada (DC/UC), debido a su dependencia en existencia respecto a las ocurrencias en la relación *GASTOS*. Los demás campos de la relaciones *TRANSPORTES*, *ALOJAMIENTOS* y *MANUTENCIONES* provienen directamente de los atributos de las entidades de las que se originan las relaciones. Solamente comentar que es imposible representar en el grafo relacional la forma de calcular los atributos derivados, por lo que será necesario incluir un disparador asociado al grafo donde se indica que la columna *importe_manutención* se calcula multiplicando el valor en euros de las dietas diarias por manutención de los profesores dependiendo de su categoría y país destino del viaje por el número de días que dura el viaje de investigación.

Por otro lado, también será necesario comprobar la exclusividad de la jerarquía mediante una aserción, ya que no es posible recoger esta semántica en el grafo relacional. Será necesario comprobar que si existe una ocurrencia en la relación *GASTOS* con valor en la columna *tipo* igual a *Transportista*, ha de existir una ocurrencia en la relación *TRANSPORTES* con la misma clave primaria y no puede existir una ocurrencia en la relación *ALOJAMIENTOS* y *MANUTENCIONES* con la misma clave primaria. Esta comprobación también sería necesaria hacerla con los valores de la columna *tipo* igual a *Alojamiento* y *Manutencion*, teniendo en cuenta las ocurrencias en las correspondientes relaciones. Por ejemplo, el pseudocódigo de la aserción que compruebe la exclusividad en la jerarquía sería la siguiente:

```
CREATE ASSERTION "Comprobar exclusividad"
Check ((Select (cod_viaje, num_gasto)
          From GASTOS
         Where tipo= 'Transportista' IN
               (Select (cod_viaje, num_gasto)
          From TRASPORTISTAS
         Where (cod_viaje, num_gasto) NOT IN
               (Select (cod_viaje, num_gasto)
          From ALOJAMIENTOS) AND
               (Select (cod_viaje, num_gasto)
          From MANUTENCIONES)))
```

AND

```
(Select (cod_viaje, num_gasto)
       From GASTOS
      Where tipo= 'Alojamiento' IN
            (Select (cod_viaje, num_gasto)
               From ALOJAMIENTOS
              Where (cod_viaje, num_gasto) NOT IN
                    (Select (cod_viaje, num_gasto)
                      From TRANSPORTISTAS) AND
                    (Select (cod_viaje, num_gasto)
                      From MANUTENCIONES)))
```

AND

```
(Select (cod_viaje, num_gasto)
       From GASTOS
      Where tipo= 'Manutencion' IN
            (Select (cod_viaje, num_gasto)
               From MANUTENCION
              Where (cod_viaje, num_gasto) NOT IN
                    (Select (cod_viaje, num_gasto)
                      From TRANSPORTISTAS) AND
                    (Select (cod_viaje, num_gasto)
                      From ALOJAMIENTOS))));
```

La Figura 2.31 muestra el grafo relacional de la transformación realizada en este apartado.

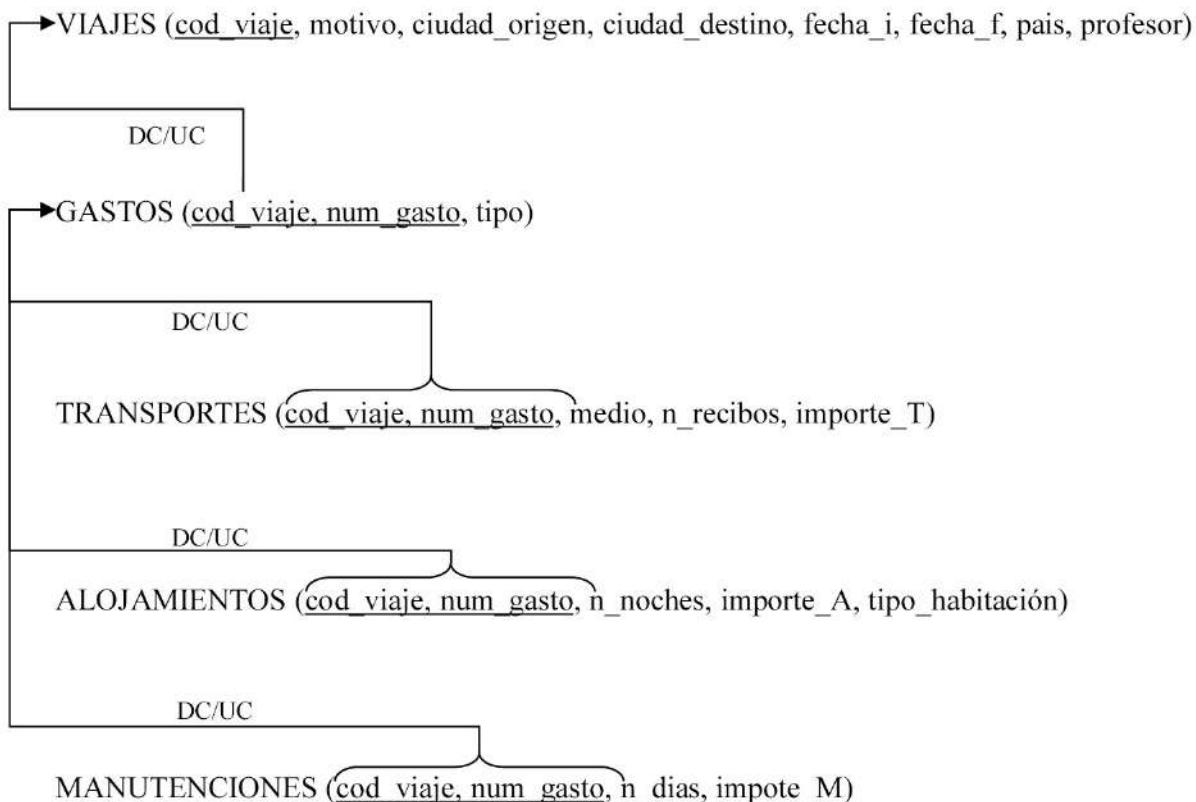


Figura 2.31. Grafo relacional sobre gastos de viajes

PARTE 3

Por último, en la Figura 2.32 se muestra el diagrama E/R de los supuestos semánticos relativos a quién financia los viajes, los proyectos de investigación en los que se encuentra trabajando un profesor o el departamento del profesor.

En este diagrama surgirá una nueva relación por cada entidad, donde su clave primaria se derivará directamente del atributo identificador principal de cada entidad. En cuanto a las interrelaciones que existen entre entidades, se detalla su transformación a continuación:

Tanto la interrelación **Financia** como la interrelación **Sufraga** son interrelaciones 1:N, por lo que se propaga la clave primaria de la relación *DEPARTAMENTOS* y la relación *PROYECTOS_INVESTIGACIÓN* a la relación *VIAJES*. Las nuevas columnas de la relación *VIAJES*, *cod_proy* y *nombre_dpto*, serán opcionales, ya que un viaje puede estar sufragado por cero o un departamento y, a su vez, un viaje puede estar financiado por cero o un proyecto de investigación. La semántica que es imposible incorporar en el grafo relacional

en este caso es que o el viaje lo financia un proyecto de investigación o lo sufraga un departamento, pero nunca se pueden dar ambos casos al mismo tiempo. Es decir, nunca pueden encontrarse estos dos campos vacíos o con valores al mismo tiempo. Esta semántica se puede comprobar mediante un *Check* en la tabla VIAJES, que tendría el siguiente pseudocódigo:

```
Check ((cod_proy<>NULL AND nombre_dpto=NULL)
      OR
      (cod_proy=NULL AND nombre_dpto<>NULL))
```

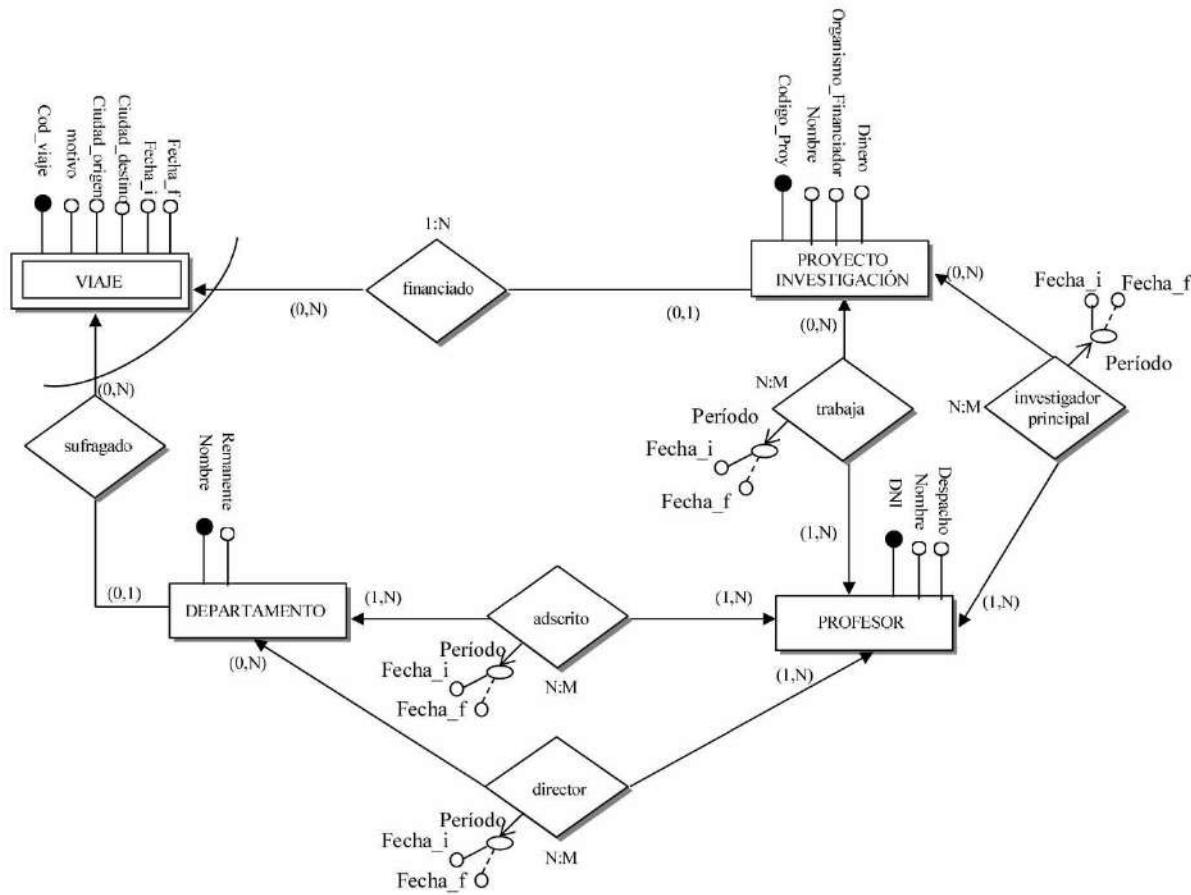


Figura 2.32. Diagrama E/R sobre financiación de viajes

Por otro lado, las interrelaciones N:M **Trabaja**, **Investigador_principal**, **Adscrito** y **Dirige** se transformarán en nuevas relaciones, donde sus atributos multivaluados compuestos serán nuevos campos de las relaciones.

En la relación **TRABAJAN** la clave primaria estará formada por las claves primarias de las relaciones implicadas en la interrelación y el atributo *Fecha_ini*,

ya que se supone que un profesor puede trabajar en varios proyectos de investigación al mismo tiempo y en un proyecto de investigación pueden trabajar varios profesores al mismo tiempo. En esta relación también se detecta una clave alternativa formada por los campos *profesor*, *proyecto* y *fecha_fin*, ya que la *fecha_inicio* puede venir determinada por los valores de estos campos. El campo *fecha_fin* permite nulos, pero esta característica no impide que forme parte de la clave alternativa comentada anteriormente. Si suponemos que en todo proyecto de investigación al menos ha de trabajar un profesor (al menos el investigador principal), sería necesario comprobar esta semántica mediante una aserción.

En cuanto a la clave primaria de la relación *INVESTIGADORES_PRINCIPALES*, es necesario tener en cuenta que en un momento dado un proyecto solo puede tener un investigador principal, aunque suponemos que un profesor puede ser investigador principal de varios proyectos en el mismo momento. Por ello, la clave primaria de la relación estará formada por los atributos *Cod_proyecto* y *Fecha_inicio*. También se puede detectar una clave alternativa en esta relación formada por los atributos *Cod_proyecto* y *Fecha_fin*, este último campo opcional. En el grafo relacional de la Figura 2.33 ha sido imposible representar la semántica de que todo proyecto de investigación ha de tener asociado un profesor investigador, y que éste ha de trabajar en el proyecto de investigación. Esta semántica se puede comprobar mediante una aserción.

En la relación *ADSCRITO* suponemos que un profesor no puede estar adscrito en dos departamentos al mismo tiempo, pero en un departamento puede haber varios profesores adscritos, por lo que podríamos tener dos claves candidatas: $\{\text{profesor}, \text{fecha_ini}\}$, $\{\text{profesor}, \text{fecha_fin}\}$. Nosotros hemos elegido la primera de ellas como clave primaria y la segunda como clave alternativa. En el caso de que supusiéramos que en todo departamento al menos tendría que haber un profesor trabajando (aquel que dirige el departamento), sería necesario comprobar mediante una aserción que en todo departamento al menos existe un profesor trabajando.

Por último, en la relación *DIRIGEN* se pueden encontrar cuatro claves candidatas: $\{\text{profesor}, \text{fecha_inicio}\}$, $\{\text{departamento}, \text{fecha_inicio}\}$, $\{\text{profesor}, \text{fecha_fin}\}$ y $\{\text{departamento}, \text{fecha_fin}\}$, ya que se supone que un profesor no puede pertenecer a varios departamentos al mismo tiempo y un departamento no puede ser dirigido por varios profesores al mismo tiempo. En el grafo relacional de la Figura 6 se ha tomado como clave primaria $\{\text{profesor}, \text{fecha_inicio}\}$, tomándose el resto de claves candidatas como claves alternativas. La semántica que no se ha podido incorporar en el grafo relacional es que todo departamento

ha de tener un director del mismo (no puede haber departamentos sin directores), y además el director del departamento ha de estar adscrito al mismo. Esta semántica se podrá comprobar mediante una aserción.

La Figura 2.33 muestra el grafo relacional de la transformación de la Figura 2.32.

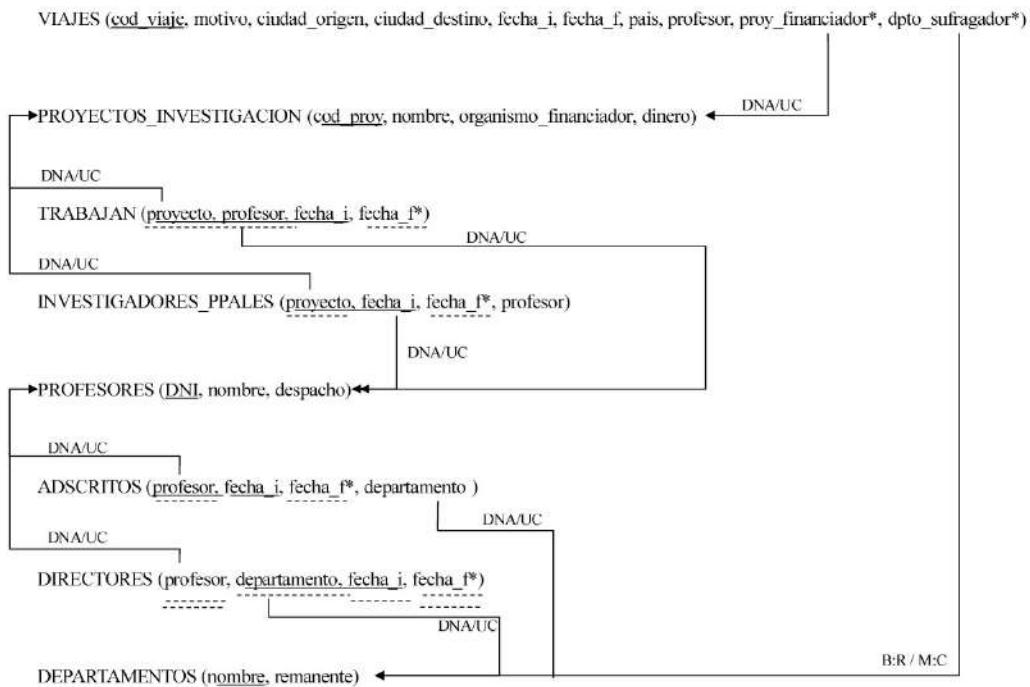


Figura 2.33. Grafo relacional sobre financiación de viajes

TRANSFORMACIÓN DE SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

La transformación de los supuestos semánticos que no se pudieron recoger en el diagrama E/R del enunciado se resume en la siguiente tabla:

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
La fecha de inicio en la que un profesor pertenece a una determinada categoría siempre ha de ser anterior o igual a la fecha de finalización (en el caso de que sea un dato conocido).	<p><i>CHECK</i> en tabla <i>PERTENECEN</i>.</p> <p>Check ((fecha_f<=fecha_i AND fecha_f<>NULL) OR (fecha_f=NULL))</p>

<p>La fecha de inicio en la que un profesor pertenece a una determinada categoría implica una única fecha de fin. También ocurre que si la fecha de fin es conocida, ésta implicará una única fecha de inicio de pertenencia a la categoría.</p>	<p>Se implementa mediante la definición de claves primarias y alternativas.</p>
<p>La fecha de inicio de las dietas diarias siempre ha de ser anterior o igual a la fecha de finalización de las mismas.</p>	<p><i>CHECK</i> en tabla <i>DIETAS_DIARIAS</i>.</p>
<p>La fecha de inicio del periodo en el que se fijan las dietas diarias en un país implica un único valor de los atributos <i>Fecha de finalización</i>, <i>Euros para manutención</i> y <i>Euros para alojamiento</i>. La fecha de finalización de un periodo de dietas diarias en un país también implica una única fecha de inicio, euros para manutención y euros para alojamiento.</p>	<p>Definición de claves primarias y alternativas.</p>
<p>Un profesor ha de pertenecer a alguna categoría en el momento en el que está realizando el viaje.</p>	<p><i>ASERCIÓN</i> en el ámbito de las relaciones <i>PROFESORES</i>, <i>VIAJES</i> y <i>PERTENECEN</i>.</p>
<p>El atributo <i>Tipo</i> de la entidad <i>GASTO</i> toma valores en el dominio <i>TIPO_GASTO</i> = {manutención, transporte, alojamiento}.</p>	<p><i>CHECK</i> en tabla <i>GASTOS</i>. Check (tipo IN { "Manutención", "Transporte", "Alojamiento" })</p>
<p>Existe una única ocurrencia en gasto de transporte por cada medio de transporte.</p>	<p><i>CHECK</i> en tabla <i>GASTOS</i>.</p>
<p>El atributo <i>Tipo_habitación</i> de la entidad <i>ALOJAMIENTO</i> toma valores en el dominio <i>TIPO_HABITACIÓN</i> = {simple, doble, triple, etc.}.</p>	<p><i>CHECK</i> en tabla <i>ALOJAMIENTOS</i>.</p>

El importe de alojamiento nunca podrá superar la dieta diaria establecida para cada categoría de profesor y país destino del viaje en las fechas en las que se ha realizado el mismo.	ASERCIÓN sobre las tablas <i>ALOJAMIENTOS</i> , <i>VIAJES</i> , <i>DIETAS_DIARIAS</i> y <i>PERTENECEN</i> .
El atributo derivado <i>Importe_M</i> de la entidad <i>MANUTENCIÓN</i> se calcula multiplicando el número de días del gasto de manutención por la dieta diaria de manutención en las fechas que se realizó el viaje.	<p><i>DISPARADOR</i> sobre la tabla <i>MANUTENCIONES</i> antes de insertar un nuevo registro.</p> <p>Consulta las tablas <i>DIETAS_DIARIAS</i>, <i>VIAJES</i> y <i>PERTENECEN</i> para poder realizar el cálculo del valor asociado al atributo derivado.</p>
El valor del atributo <i>Número_días</i> tenido en cuenta en los gastos de manutención ha de ser igual o menor al número de días del viaje (restando el atributo <i>Fecha_fin</i> al atributo <i>Fecha_inicio</i> de la entidad <i>VIAJE</i>).	ASERCIÓN en el ámbito de las tablas <i>MANUTENCIONES</i> y <i>VIAJES</i> .
El valor de <i>Fecha_fin</i> (en el caso de que sea conocido) en la interrelación Adscrito es siempre igual o posterior al valor de la <i>Fecha_inicio</i> .	<i>CHECK</i> sobre la tabla <i>ADSCRITOS</i> .
Un profesor no puede estar adscrito a dos departamentos al mismo tiempo.	Definición de claves primarias y ajenas.
<p>La fecha de inicio en la que un profesor trabaja en un determinado departamento implica una única fecha de finalización.</p> <p>De igual forma ocurre con la fecha de fin (en el caso de que sea conocida), implicaría una única fecha de inicio de adscripción a un departamento.</p>	Definición de claves primarias y alternativas.

La fecha de inicio de un viaje ha de coincidir con las fechas en las que el profesor estaba adscrito al departamento que sufraga el viaje.	<i>ASERCIÓN</i> sobre las tablas <i>VIAJES</i> y <i>ADSCRITOS</i> .
Durante el periodo en el que el profesor se encuentra adscrito a un departamento, éste pertenece a alguna categoría de profesorado.	<i>ASERCIÓN</i> sobre las tablas <i>ADSCRITOS</i> , <i>PERTENECEN</i> .
La fecha de inicio de un viaje ha de coincidir con las fechas en las que el profesor ha estado trabajando en el proyecto de investigación que financia el viaje.	<i>ASERCIÓN</i> sobre las tablas <i>VIAJES</i> y <i>TRABAJAN</i> .
Durante el periodo de trabajo en un determinado proyecto de investigación, el profesor pertenece a alguna categoría de profesorado.	<i>ASERCIÓN</i> sobre las tablas <i>TRABAJAN</i> y <i>PERTENECEN</i> (no sería necesario comprobarlo tras comprobar la asercción del supuesto semántico siguiente).
Durante el periodo de trabajo en el proyecto el profesor está adscrito a algún departamento de la universidad.	<i>ASERCIÓN</i> sobre las tablas <i>TRABAJAN</i> y <i>ADSCRITOS</i> .
En un momento dado solo un profesor puede ser investigador principal de un proyecto de investigación.	Definición de claves primarias y alternativas.
La fecha de inicio en la que un profesor es investigador principal de un proyecto siempre ha de ser anterior o igual a la fecha de fin.	<i>CHECK</i> sobre la tabla <i>INVESTIGADORES_PPALES</i> .
Además, ésta implica una única fecha de fin y, si ésta fuese conocida, implicaría una única fecha de inicio.	Definición de claves primarias y alternativas.
En las fechas en las que el profesor ha sido investigador principal de un proyecto, éste se encontraba trabajando en el mismo.	<i>ASERCIÓN</i> en el ámbito de las tablas <i>INVESTIGADORES_PPALES</i> y <i>TRABAJAN</i> .

En un momento dado solo un profesor puede ser director de un departamento.	Definición de claves primarias y alternativas.
La fecha de inicio en la que un profesor es director de un departamento ha de ser anterior o igual a la fecha de fin.	<i>CHECK</i> sobre la tabla <i>DIRECTORES</i> .
Además, ésta implica una única fecha de fin y, si ésta fuese conocida, implicaría una única fecha de inicio.	Definición de claves primarias y alternativas.
En las fechas en las que el profesor ha sido director de un departamento, éste estaba adscrito al mismo.	<i>ASERCIÓN</i> sobre las tablas <i>DIRECTORES</i> y <i>ADSCRITOS</i> .
El total de gastos asociados a los viajes que financian los proyectos de investigación o que sufragan los departamentos nunca será superior al dinero de que disponen respectivamente.	<i>ASERCIÓN</i> sobre las tablas <i>PROYECTOS_INVESTIGACION</i> o <i>DEPARTAMENTOS</i> (respectivamente) y las tablas <i>MANUTENCIONES</i> , <i>ALOJAMIENTOS</i> , <i>TRANSPORTES</i> y <i>VIAJES</i> .
Los periodos de tiempo en los que un profesor trabaja en departamentos no pueden solaparse, y solo una de las fechas de finalización puede ser <i>NULL</i> (la actual).	<i>CHECK</i> en la tabla <i>TRABAJAN</i> .
De la misma forma, un proyecto de investigación no puede tener dos investigadores principales de manera solapada y solo una fecha de finalización a <i>NULL</i> (la actual).	<i>CHECK</i> en la tabla <i>INVESTIGADORES_PPALES</i> .
Totalidad de la jerarquía <i>GASTOS</i> .	<i>ASERCIÓN</i> sobre los subtipos y el supertipo.

PROBLEMA 2.12: GESTIÓN DE ALQUILERES

En este problema se parte del esquema E/R del Problema 1.1 del Capítulo 1, y se pide realizar la transformación del esquema E/R a un grafo relacional, recogiendo como *checks*, aserciones y disparadores todos aquellos supuestos semánticos que haya sido imposible incorporar en el grafo.

El ejercicio de transformación del grafo relacional se va a desarrollar en tres partes con el objetivo de facilitar la comprensión del estudiante. Posteriormente se comenta la transformación de todos los supuestos semánticos no incluidos en el diagrama E/R.

DISCUSIÓN DEL ENUNCIADO

PARTE 1

La Figura 2.34 corresponde al diagrama E/R sin redundancias del diseño inicial de una BD que almacena información sobre alquileres de viviendas a particulares.

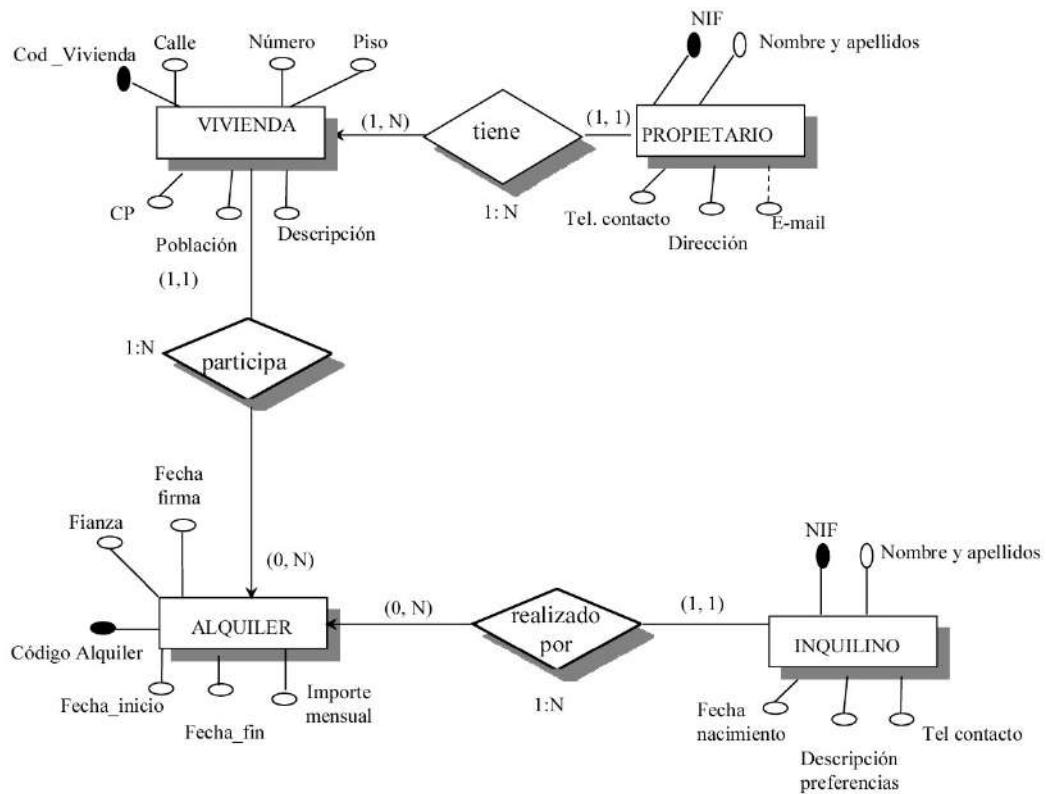


Figura 2.34. Diagrama inicial E/R sobre gestión de alquileres

En la Figura 2.35 se muestra el grafo relacional que surge a partir de la transformación del esquema E/R de la Figura 2.34.

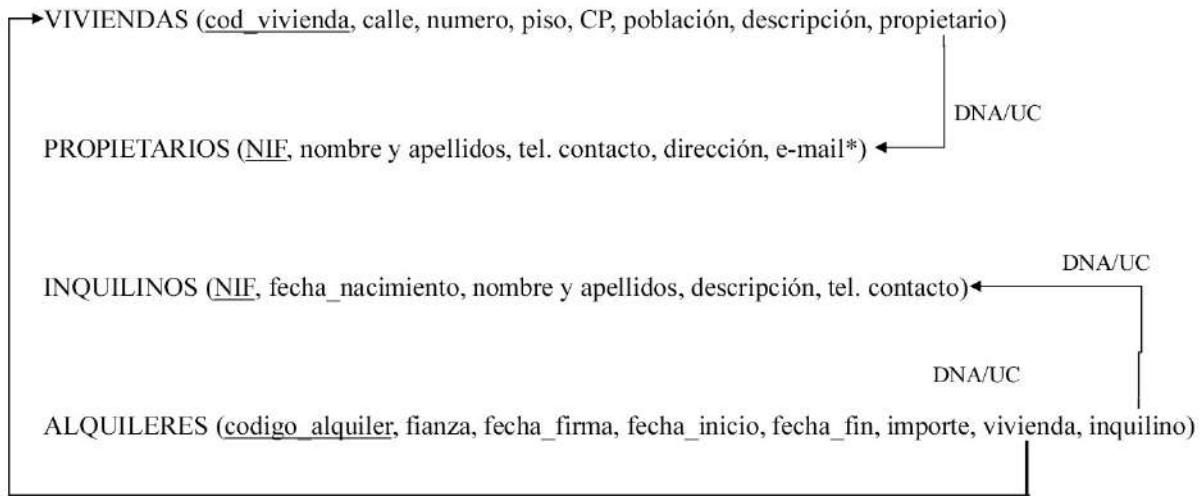


Figura 2.35. Esquema Relacional inicial sobre gestión de alquileres

Solo comentar que los atributos opcionales en el diagrama E/R se han marcado como campos que permiten nulos en el grafo relacional (como por ejemplo el campo *e-mail* de la tabla *PROPIETARIOS*). Los atributos identificadores primarios de las entidades se han transformado en las claves primarias de las relaciones en el grafo relacional. Por último, se ha considerado el borrado sin acción y la modificación restringida como opciones de las restricciones de integridad para las claves ajenas en todas las relaciones, ya que son las opciones menos restrictivas.

Durante la transformación del diagrama E/R al grafo relacional se ha perdido semántica que será necesario comprobar, en este caso, mediante aserciones. Se ha de comprobar que todo propietario de nuestra base de datos ha de tener al menos una vivienda (aserción que consulta las tablas *PROPIETARIOS* y *VIVIENDAS*).

PARTE 2

En la Figura 2.36 se muestra el diagrama E/R que representa conceptualmente información sobre la renovación de los alquileres.

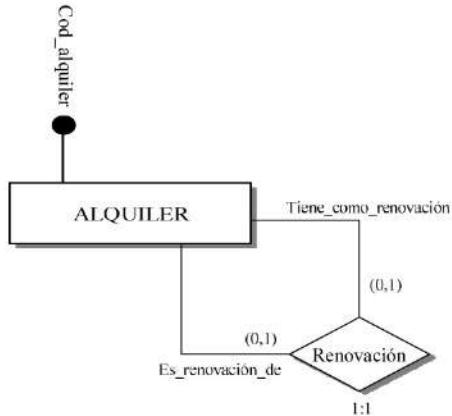


Figura 2.36. Interrelación reflexiva “renovación”

La transformación de una interrelación 1:1 se puede llevar a cabo de diferentes maneras, fijando como objetivo principal en todo momento incorporar el máximo de semántica al grafo relacional y evitar tuplas en las relaciones con valores nulos.

Si se realizara una propagación de claves para transformar la interrelación (suponiendo que se trata de un caso especial de una interrelación 1:N), el grafo relacional sería el que aparece en la Figura 4. En este grafo se ha añadido un campo opcional *alquiler_antiguo*, considerando de esta manera la cardinalidad mínima de cero y máxima de uno del rol *tiene_como_renovación* en la interrelación. Para hallar los códigos de los alquileres que son renovación de los alquileres antiguos sería necesario buscar en la tabla por el *cod_alquiler* que posee el *alquiler_antiguo* en la relación *ALQUILERES*.

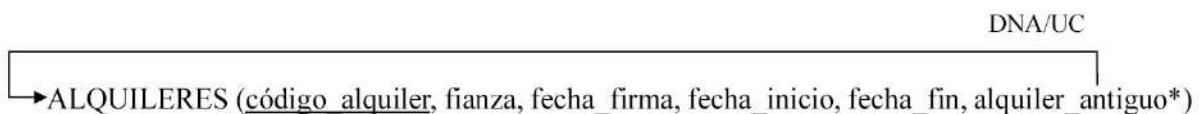


Figura 2.37. Transformación propagando claves de la interrelación reflexiva “renovación”

El problema de transformar la interrelación de esta forma es que podría ocurrir que hubiera muchos alquileres nuevos (que no fueran renovación de otros alquileres antiguos), por lo que surgirían muchos valores nulos en las tuplas de esta tabla. Además, no ha sido posible incorporar la información de que un alquiler solo puede aparecer como *alquiler_antiguo* en una única tupla.

Otra forma de transformar esta interrelación es considerando que es un caso especial de una interrelación N:M y transformar la interrelación en una nueva

tabla. De esta forma, en la relación *RENOVACIONES* únicamente se tendrá información sobre qué alquileres han sido renovados y por qué alquileres (no aparecerá información en esta tabla sobre alquileres que no han sido renovados). De esta forma se evitan valores nulos en las tuplas.

En la Figura 2.38 se muestra cómo en este caso existen dos posibles claves candidatas: el atributo *alquiler_nuevo* o el atributo *alquiler_antiguo*. Se considerará como clave primaria aquélla por la que sea más probable realizar búsquedas en la base de datos y necesitemos relacionarla con otras ocurrencias de otras tablas, siendo clave alternativa la otra clave candidata. Como en el enunciado no se incluye información al respecto, nosotros hemos considerado que las búsquedas y relaciones con otras tablas se realizarán a través del atributo *alquiler_nuevo*.

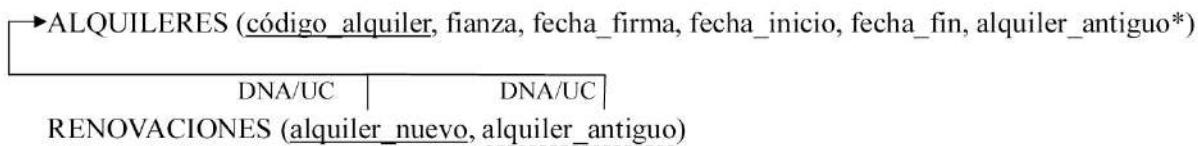


Figura 2.38. Transformación en una nueva tabla de la interrelación reflexiva “renovación”

Otra ventaja de transformar de esta manera la interrelación reflexiva es que de esta forma se ha podido incorporar toda la semántica al grafo relacional.

PARTE 3

El diagrama E/R de la Figura 2.39 añade información sobre las agencias de alquiler que ofertan las viviendas. En el grafo relacional de la Figura 2.40 se ha transformado al esquema relacional el diagrama E/R.

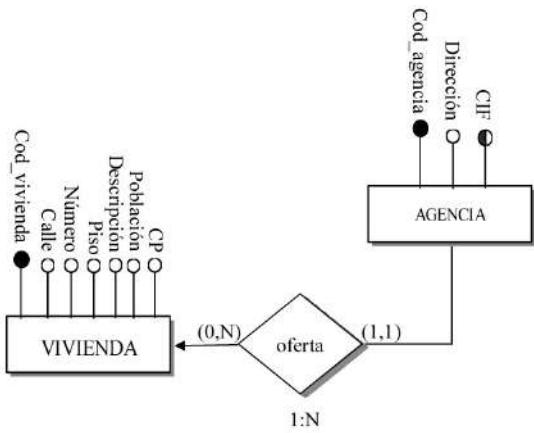


Figura 2.39. Diagrama E/R sobre agencias que ofertan viviendas en alquiler

AGENCIAS (<u>cod_agencia</u> , dirección, CIF)	←	DNA/UC
VIVIENDAS (<u>cod_vivienda</u> , calle, numero, piso, CP, población, descripción, agencia)	→	

Figura 2.40. Grafo relacional sobre agencias que ofertan viviendas en alquiler

La transformación de esta interrelación 1:N se ha realizado mediante propagación de clave. El atributo identificador alternativo *CIF* de la entidad *AGENCIA* se ha transformado en la clave alternativa de la relación *AGENCIAS*. No se ha producido pérdida de semántica en la transformación del diagrama E/R al grafo relacional.

PROPUESTA DE SOLUCIÓN

La Figura 2.41 muestra el grafo relacional completo.

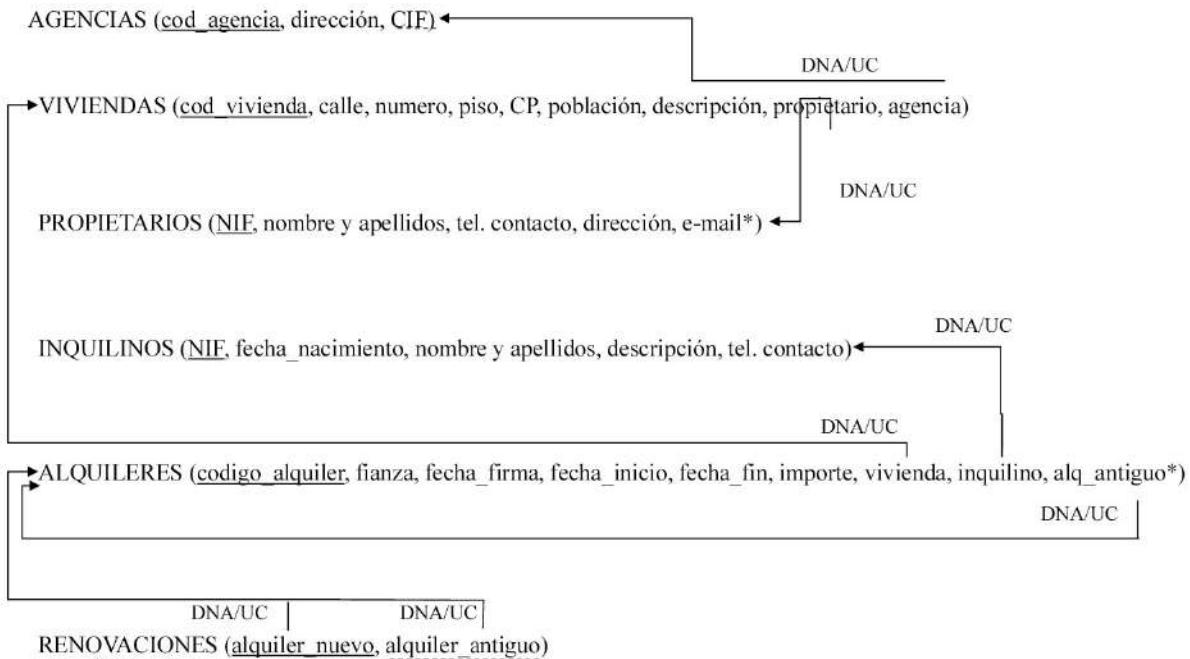


Figura 2.41. Grafo relacional completo

TRANSFORMACIÓN DE LOS SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

Los supuestos semánticos que no pudieron ser incluidos en el diagrama E/R se han transformado al grafo relacional de la siguiente forma:

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
<p>La fecha de firma (<i>Fecha_firma</i>) de un determinado alquiler ha de ser anterior o igual en el tiempo que la fecha de inicio (<i>Fecha_inicio</i>) del alquiler. Al mismo tiempo, la fecha de fin (<i>Fecha_fin</i>) del alquiler ha de ser igual o posterior en el tiempo a la fecha de inicio del alquiler.</p>	<p><i>CHECK</i> sobre la tabla <i>ALQUILERES</i>.</p>
<p>La fecha de nacimiento (<i>Fecha_nacimiento</i>) de un inquilino nunca podrá ser posterior en el tiempo a la fecha de firma (<i>Fecha_firma</i>) de la vivienda que está alquilando. Si se tuviera en cuenta que un inquilino no podrá alquilar una vivienda si es menor de edad, habrá que comprobar que</p> <p><i>INQUILINO.Fecha_nacimiento+18años <= ALQUILER.Fecha_firma</i> para todas las viviendas alquiladas por el mismo inquilino.</p>	<p><i>ASERCIÓN</i> sobre las tablas <i>INQUILINO</i> y <i>ALQUILERES</i>.</p>
<p>No pueden existir dos contratos de alquiler (ocurrencias en la entidad <i>ALQUILER</i>) sobre la misma vivienda cuyos períodos de renta se solapen.</p>	<p><i>CHECK</i> en la tabla <i>ALQUILERES</i>.</p>
<p>Un alquiler no puede nunca ser renovación de sí mismo ni de otros alquileres que han sido a su vez (en uno o varios grados) renovación del mismo.</p>	<p><i>CHECK</i> en la tabla <i>RENOVACIONES</i>.</p>
<p>Un contrato de alquiler sobre una determinada vivienda no puede ser renovado en una fecha anterior a la finalización del contrato anterior.</p>	<p><i>ASERCIÓN</i> sobre las tablas <i>RENOVACIONES</i> y <i>ALQUILERES</i>.</p>

<p>La <i>Fianza</i> y el <i>Importe Mensual</i> de la entidad <i>ALQUILER</i> han de ser números enteros mayores o iguales a cero.</p>	<p><i>CHECK</i> sobre la tabla <i>ALQUILERES</i>.</p>
<p>Los valores de los atributos <i>NIF</i> y <i>Tel_contacto</i> de la entidad <i>INQUILINO</i> han de cumplir un determinado formato. Por ejemplo, el <i>NIF</i> ha de ser una cadena de caracteres formada por un número de 8 dígitos, seguido por un guión y por una letra (ejemplo 02348756-K). También han de comprobarse los formatos de los atributos <i>NIF</i>, <i>E-mail</i>, <i>Tel_contacto</i> de la entidad <i>PROPIETARIO</i> y el código postal (CP), el atributo <i>Piso</i> y el atributo <i>Número</i> de la entidad <i>VIVIENDA</i>.</p>	<p><i>CHECK</i> sobre la tabla <i>INQUILINOS</i>, <i>PROPIETARIOS</i> o <i>VIVIENDAS</i>, respectivamente.</p>

Se plantea al usuario el ejercicio de explicar en pseudocódigo los *checks* y aserciones que se definen en este problema.

PROBLEMA 2.13: PROYECTOS DE INVESTIGACIÓN

Dado el esquema E/R resultado del Problema 1.4 del Capítulo 1.

Se pide:

Transformar el esquema E/R al Modelo Relacional (grafo relacional) indicando toda la semántica que no se ha podido incorporar en el grafo en forma de *checks*, aserciones y disparadores.

DISCUSIÓN DEL ENUNCIADO

La transformación de los diagramas E/R se va a discutir en varias partes, dependiendo de la semántica asociada a los subdiagramas E/R. De esta forma se pretende facilitar la comprensión del ejercicio al lector. Por último, se plantea la transformación de los supuestos semánticos que no se pudieron incorporar al diagrama E/R.

PARTE 1

En la Figura 2.42 se muestra el diagrama E/R que representa información sobre la participación en proyectos de investigación de profesores doctores y no doctores, así como información sobre la supervisión a profesores no doctores.

El grafo relacional que surge a partir de la transformación de esta figura se muestra en la Figura 2.43.

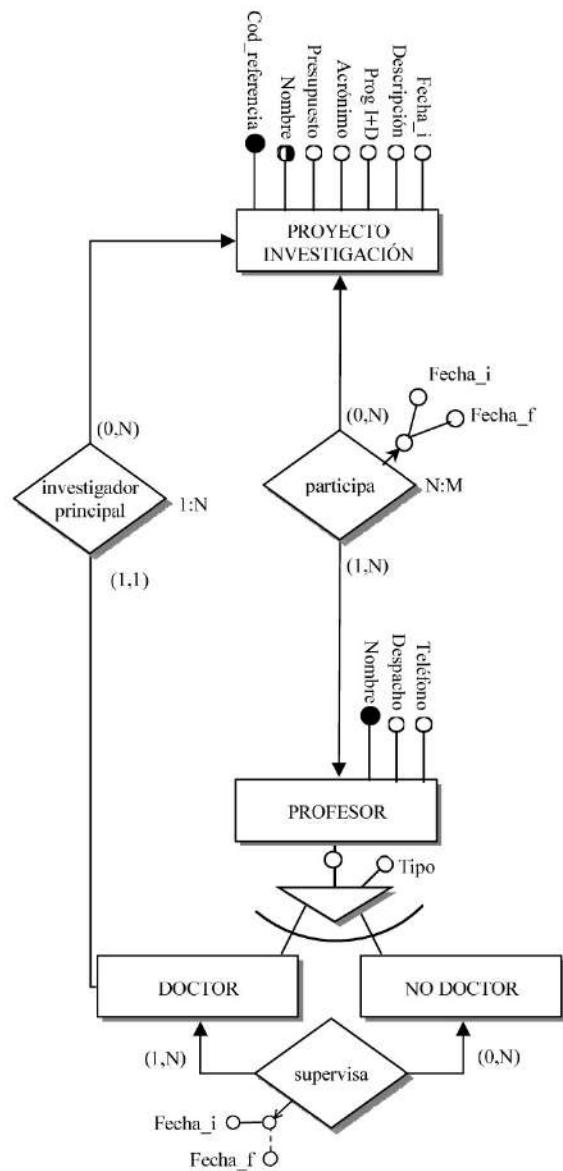


Figura 2.42. Diagrama E/R sobre participación de profesores en proyectos de investigación

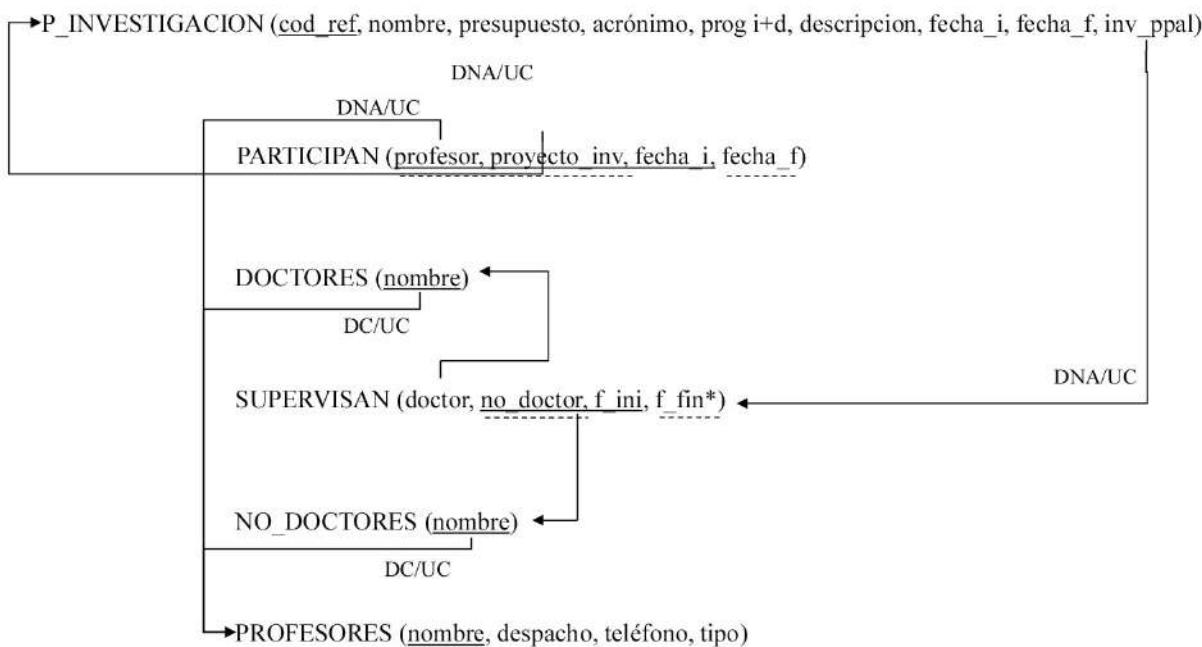


Figura 2.43. Grafo Relacional sobre participación de profesores en proyectos de investigación

Se ha transformado la jerarquía *PROFESOR* creando tres relaciones diferentes, una para el supertipo de la jerarquía (*PROFESORES*) y una por cada suptipo de la misma (*DOCTORES* y *NO DOCTORES*). Las restricciones de integridad sobre las claves ajena que referencian a los profesores de esta generalización se definen como muy restrictivas: tanto el borrado como la modificación de las claves ajena se define en cascada.

La interrelación 1:N **Investigador_principal** se ha transformado mediante propagación de clave a la relación *P_INVESTIGACION*, mientras que la interrelación N:M **Participan** se ha transformado en una nueva relación *PARTICIPAN*. Las restricciones de integridad de las claves ajena de estas transformaciones se han tomado como las menos restrictivas, ya que no se indica nada al respecto en el enunciado: borrado sin acción y modificación en cascada.

En la relación *PARTICIPAN* se han definido dos claves candidatas $\{profesor, proyecto_inv, fecha_i\}$ y $\{profesor, proyecto_inv, fecha_f\}$ dado que un profesor que participa en un proyecto en una fecha de inicio determinado únicamente puede tener asociada una fecha de fin para ese periodo. De igual forma, la fecha de fin en la que un profesor participa en un proyecto, determina la fecha en la que comenzó su participación. En el grafo relacional de la Figura 2.43 se ha elegido $\{profesor, proyecto_inv, fecha_i\}$ como clave primaria de la relación y $\{profesor, proyecto_inv, fecha_f\}$ como clave alternativa.

Al transformar esta interrelación, se ha producido una pérdida de semántica que hay que controlar mediante una aserción: en todo proyecto de investigación al menos ha de participar un profesor.

Por último, la interrelación N:M **Supervisa** que almacena información histórica de qué profesores doctores han supervisado a qué profesores no doctores, se transforma en una nueva relación SUPERVISAN. Se detectan en esta nueva relación dos posibles claves candidatas: $\{no_doctor, f_ini\}$ y $\{no_doctor, f_fin\}$, definiendo la primera como clave primaria y la segunda como clave alternativa en el grafo de la Figura 2.43.

NOTA: la clave primaria de esta relación no incluye en ningún caso el campo *doctor*, ya que en un momento dado un profesor no doctor solo puede ser supervisado por un profesor doctor.

NOTA: aunque la columna *f_fin* puede permitir valores nulos, ésta puede formar parte de claves alternativas en la relación.

La transformación de la interrelación **Supervisa** también produce pérdidas semánticas. En el grafo relacional ha sido imposible comprobar que todo profesor no doctor ha de estar supervisado en todo momento por un profesor doctor.

PARTE 2

En la Figura 2.44 se muestra el diagrama E/R de los supuestos semánticos relativos a las publicaciones que se realizan en los proyectos de investigación. La transformación de este diagrama E/R se muestra en la Figura 2.45.

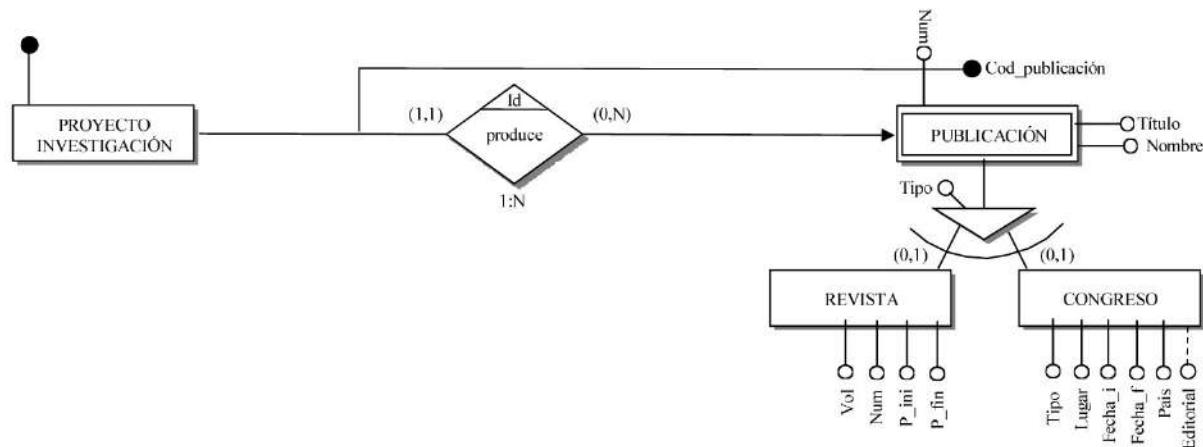


Figura 2.44. Diagrama E/R sobre publicaciones que se derivan de proyectos de investigación

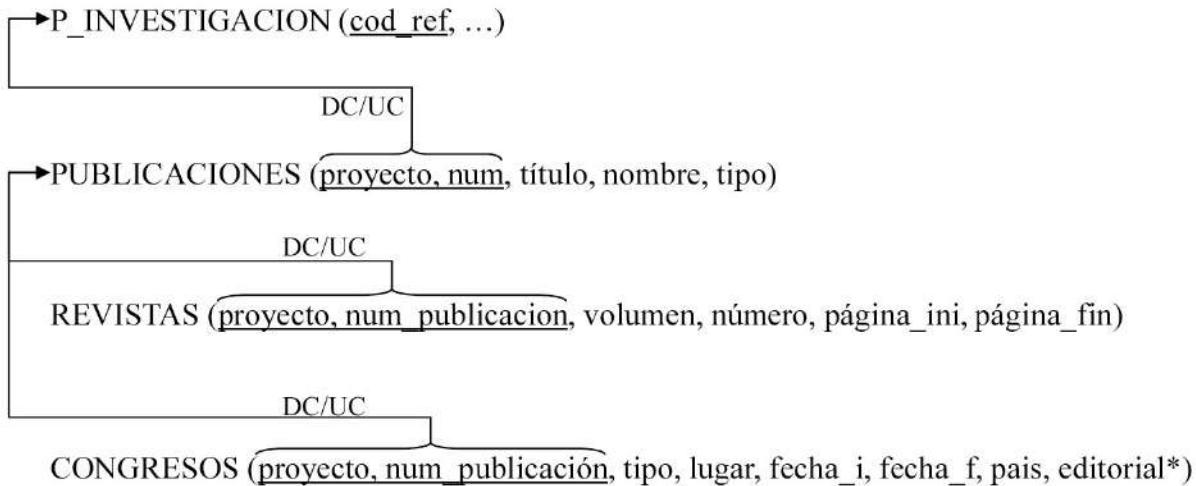


Figura 2.45. Grafo relacional sobre publicaciones que se derivan de proyectos de investigación

La jerarquía *PUBLICACIÓN* se ha transformado en tres nuevas relaciones, una relación que referencia al supertipo de la jerarquía (*PUBLICACIONES*) y una relación referenciando a cada subtipo de la misma (*REVISTAS* y *CONGRESOS*). Se han definido las restricciones de integridad de las claves ajena en este caso como borrados y modificaciones en cascada, ya que las ocurrencias de los subtipos de una jerarquía no tienen sentido en la base de datos a no ser que exista su respectiva ocurrencia en el supertipo de la misma.

Por otro lado, la interrelación en identificación **Produce** se ha transformado mediante una propagación de clave, donde los borrados y modificaciones también se han definido en cascada. La clave primaria de la relación *PUBLICACIONES* se forma con la clave primaria de la relación *P_INVESTIGACION* y un número propio de la publicación.

PARTE 3

Para finalizar, en la Figura 2.46 se muestra información sobre las líneas de investigación incluidas en cada proyecto de investigación, las líneas de investigación de cada profesor y el número de orden en el que aparecen los profesores en las distintas publicaciones según la línea de investigación.

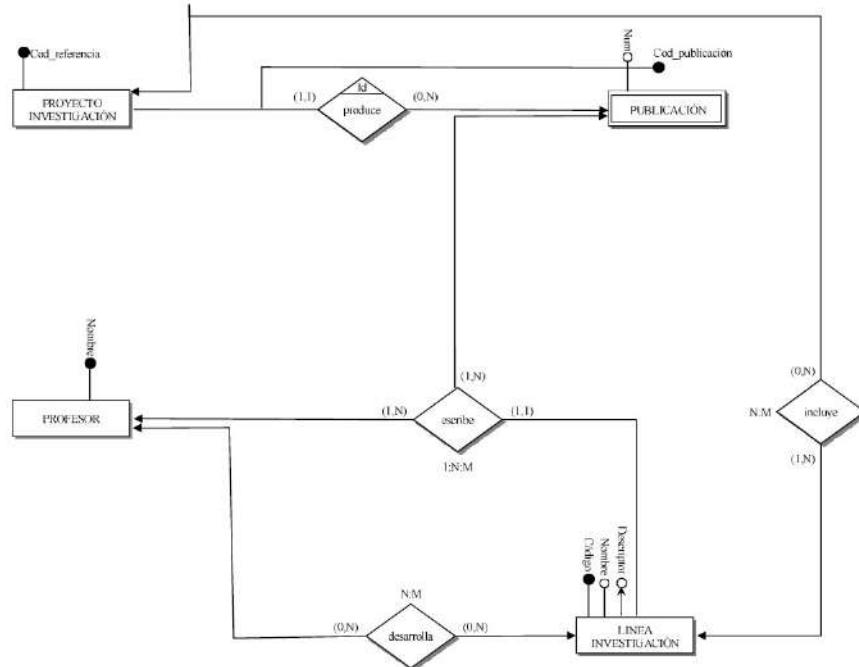


Figura 2.46. Diagrama E/R sobre líneas de investigación incluidas en proyectos

La Figura 2.47 muestra el grafo relacional resultado de transformar el diagrama E/R de la Figura 2.46.

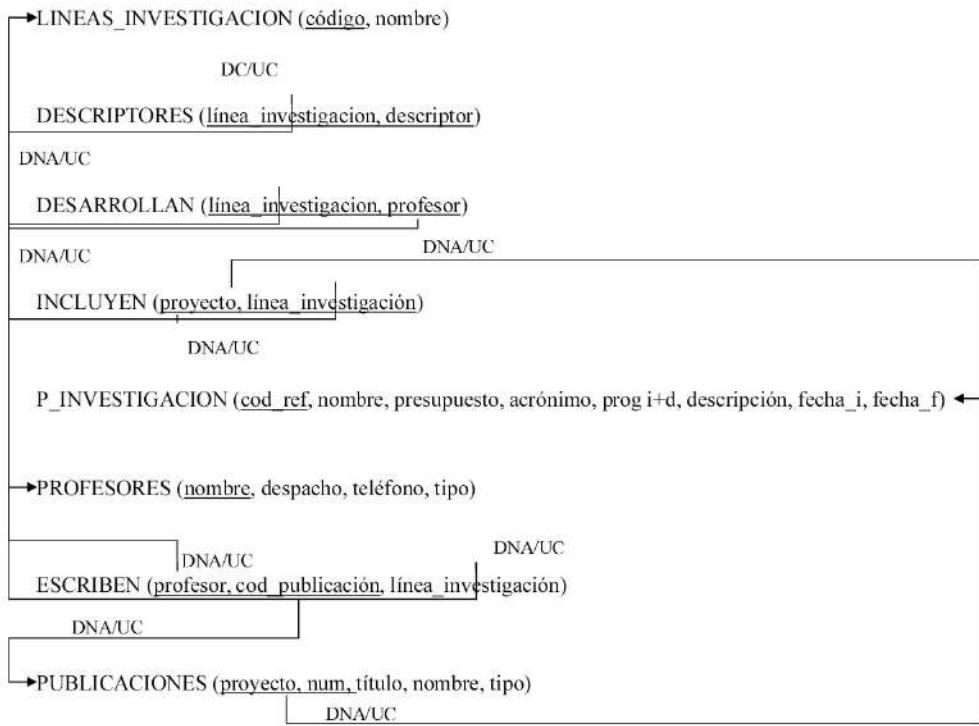


Figura 2.47. Grafo relacional correspondiente al esquema E/R de la Figura 2.46

Para transformar el atributo multivaluado *Descriptores* de la entidad *LINEA_INVESTIGACION* ha sido necesario incorporar al grafo relacional una nueva relación que se ha denominado *DESCRIPTORES*. La clave primaria de esta nueva relación está formada por la clave primaria de la relación *LINEAS_INVESTIGACION* y la columna *descriptor*. De esta forma se hace posible almacenar en una tabla varios descriptores para una misma línea de investigación. Las opciones de borrado y modificación de la clave ajena en la relación *DESCRIPTORES* son en cascada, debido a que un descriptor depende en existencia de las líneas de investigación que describe.

Por otro lado, las interrelaciones N:M **Incluye** y **Desarrolla** se han transformado como dos nuevas relaciones: *INCLUYEN* y *DESARROLLAN*, respectivamente. La clave primaria de ambas se forma uniendo las claves primarias de las relaciones que relaciona. Los borrados y modificaciones de las claves ajenas de las nuevas relaciones se han definido como poco restrictivas: borrados sin acción y modificación en cascada. La transformación de la interrelación **Desarrolla** no produce pérdida de semántica, pero la transformación de la interrelación **Incluye** sí, ya que será necesario controlar mediante una asercción que todo proyecto de investigación ha de incluir al menos una línea de investigación.

Por último, la interrelación ternaria **Escribe** se transforma en una nueva relación. Debido a que la cardinalidad de la interrelación que referencia a las líneas de investigación es (1,1), la columna *linea_investigacion* no será parte de la clave primaria. Es decir, se puede recoger en el esquema relacional la semántica de un profesor cuando escribe sobre una publicación únicamente puede escribir sobre una única línea de investigación. La transformación de esta interrelación no incluye pérdida de semántica.

TRANSFORMACIÓN DE LOS SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

La transformación de los supuestos semánticos que no se pudieron incorporar en el diagrama E/R final se detalla en la siguiente tabla:

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
No se pueden superponer los periodos de tiempo en los que un profesor participa en un proyecto dado.	<i>CHECK</i> en la relación <i>PARTICIPAN</i> .

La fecha de inicio de participación en un proyecto siempre ha de ser anterior a la fecha de finalización.	<i>CHECK</i> en la relación <i>PARTICIPAN</i> .
La fecha de inicio de participación de un profesor en un proyecto de investigación implica una única fecha de finalización de la participación. De igual manera, si la fecha de finalización fuera conocida, ésta implicaría una única fecha de inicio de participación en el proyecto para cada profesor.	Se controla mediante la definición de las claves primarias y alternativas en la relación <i>PARTICIPAN</i> .
Un profesor no doctor nunca podrá tener a varios profesores doctores como supervisores de forma simultánea.	<i>CHECK</i> en la relación <i>SUPERVISAN</i> .
La fecha de inicio de supervisión implica una única fecha de fin y, en el caso de que ésta fuera conocida, una fecha de finalización implicaría una única fecha de inicio de supervisión de un profesor a otro.	Se controla mediante la definición de claves primarias y alternativas en la relación <i>SUPERVISAN</i> .
La fecha de inicio de supervisión de un profesor doctor a un profesor no doctor siempre ha de ser anterior a la fecha de fin de supervisión.	<i>CHECK</i> en la relación <i>SUPERVISAN</i> .
Los profesores solo pueden ser investigadores principales de los proyectos de investigación en los que participan.	<i>ASERCIÓN</i> sobre las tablas <i>P_INVESTIGACION</i> y <i>PARTICIPAN</i> .
Un profesor solo puede escribir publicaciones sobre líneas de investigación que desarrolla.	<i>ASERCIÓN</i> sobre las tablas <i>ESCRIBEN</i> y <i>DESARROLLAN</i> .
Un profesor no puede escribir en una publicación de un proyecto de investigación si éste no trabaja en el mismo.	<i>ASERCIÓN</i> sobre las tablas <i>ESCRIBEN</i> y <i>TRABAJAN</i> .

Un proyecto de investigación solo puede incluir líneas de investigación de los profesores que trabajan en el proyecto.	<i>ASERCIÓN</i> sobre las tablas <i>INCLUYEN</i> , <i>TRABAJAN</i> y <i>DESARROLLAN</i> .
Las líneas de investigación de una determinada publicación han de encontrarse como ocurrencias en la interrelación. Incluye entre las entidades <i>LÍNEA_INVESTIGACIÓN</i> y <i>PROYECTO_INVESTIGACIÓN</i> .	<i>ASERCIÓN</i> sobre las tablas <i>ESCRIBEN</i> e <i>INCLUYEN</i> .
La página de inicio de una publicación en una revista siempre ha de ser anterior a la página de fin de la misma.	<i>CHECK</i> sobre la tabla <i>REVISTAS</i> .
La fecha de inicio de un congreso siempre ha de ser anterior o igual a la fecha de fin del mismo.	<i>CHECK</i> sobre la tabla <i>CONGRESOS</i> .
La fecha de inicio del proyecto de investigación siempre ha de ser anterior o igual a la fecha de inicio de la publicación en un congreso.	<i>ASERCIÓN</i> sobre las tablas <i>P_INVESTIGACIÓN</i> , <i>PUBLICACIONES</i> y <i>CONGRESOS</i> .
El atributo discriminante <i>Tipo</i> de la jerarquía de <i>PUBLICACIÓN</i> toma valores en el dominio <i>TIPO_PUBLICACIÓN</i> = <i>{congreso, revista}</i> .	<i>CHECK</i> sobre la tabla <i>PUBLICACIONES</i> .
El atributo <i>Tipo</i> de la entidad <i>CONGRESO</i> toma valores en el dominio <i>TIPO_CONGRESO</i> = <i>{nacional, internacional}</i> .	<i>CHECK</i> sobre la tabla <i>CONGRESOS</i> .
El atributo discriminante <i>Tipo</i> de la jerarquía <i>PROFESOR</i> toma valores en el dominio <i>TIPO_PROFESOR</i> = <i>{doctor, no_doctor}</i> .	<i>CHECK</i> sobre la tabla <i>PROFESORES</i> .

La fecha de inicio de un proyecto de investigación siempre ha de ser anterior o igual a la fecha de fin del mismo.	<i>CHECK</i> sobre la tabla <i>PROYECTOS</i> .
Exclusividad de la jerarquía <i>PUBLICACIÓN</i> .	<i>ASERCIÓN</i> sobre los subtipos.
Totalidad de la jerarquía <i>PUBLICACIÓN</i> .	<i>ASERCIÓN</i> sobre el supertipo y los subtipos.
Exclusividad de la jerarquía <i>PROFESOR</i> .	<i>ASERCIÓN</i> sobre los subtipos.
Totalidad de la jerarquía <i>PROFESOR</i> .	<i>ASERCIÓN</i> sobre el supertipo y los subtipos.

PROBLEMA 2.14: GESTIÓN DE PROYECTOS INFORMÁTICOS

Dado el esquema E/R resultado del Problema 1.9 del Capítulo 1.

Se pide:

Transformar el esquema E/R al Modelo Relacional (grafo relacional) indicando toda la semántica que no se ha podido incorporar en el grafo en forma de *checks*, aserciones y disparadores.

DISCUSIÓN DEL ENUNCIADO

La transformación de los diagramas E/R se va a discutir en varias partes, dependiendo de la semántica asociada a los subdiagramas E/R. De esta forma se pretende facilitar la comprensión del ejercicio al lector. Por último, se plantea la transformación de los supuestos semánticos que no se pudieron incorporar al diagrama E/R.

PARTE 1

En la Figura 2.48 se muestra el diagrama E/R que representa información sobre la participación de los empleados de la empresa, así como las fases de las que consta cada proyecto.

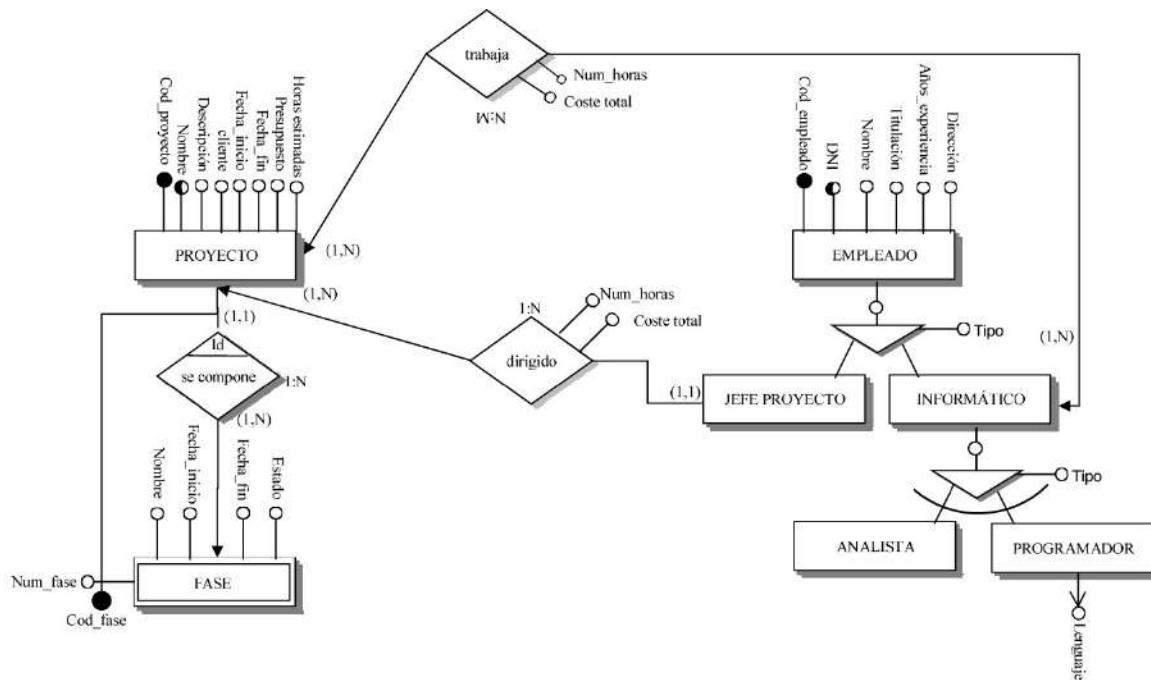


Figura 2.48. Diagrama E/R sobre la participación de los empleados en proyectos informáticos

El grafo relacional que surge a partir de la transformación de esta figura se muestra en la Figura 2.49.

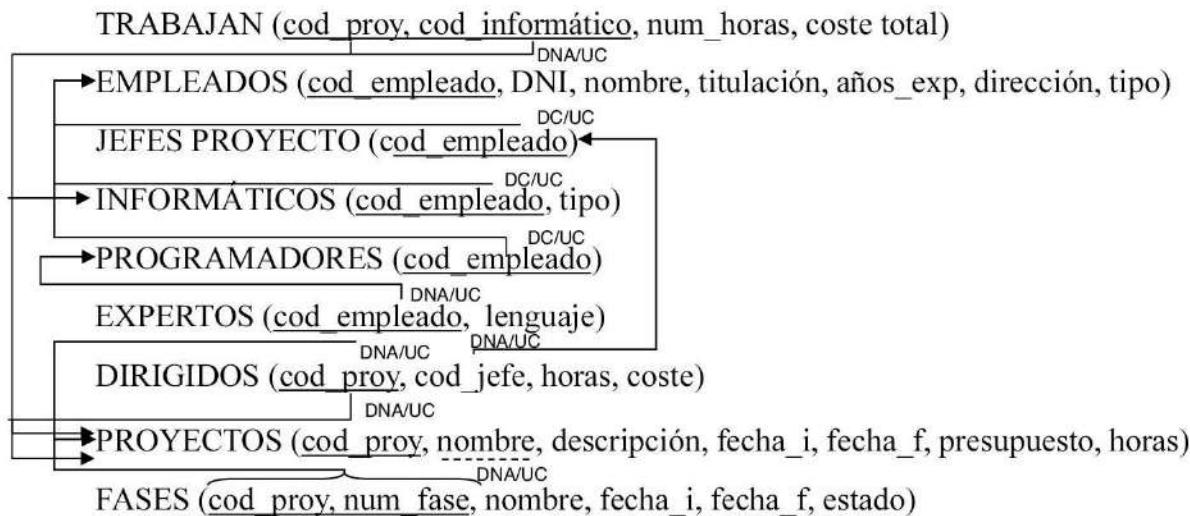


Figura 2.49. Grafo relacional relativo a la participación de empleados en proyectos informáticos

La jerarquía a dos niveles *EMPLEADO* se ha transformado considerando una relación por cada supertipo y otra por cada subtipo excepto para el caso de la entidad *ANALISTA* pues, en principio, ésta ni tiene atributos propios ni se interrelaciona con ninguna otra entidad. Además, se ha incluido una nueva relación *EXPERTOS* para recoger la semántica asociada al atributo multivaluado *Lenguajes* que recoge los lenguajes en los que son expertos los programadores. En cuanto a la semántica que no se puede incorporar en el grafo relacional:

- Solapamiento y totalidad de la jerarquía *EMPLEADO*: es necesario confirmar mediante una aserción que si el código de empleado aparece solo en la relación *JEFES_PROYECTO*, el atributo *Tipo* de la relación *EMPLEADOS* ha de tener un valor referente a jefes de proyecto. Si, por el contrario, el código de empleado aparece solo en la relación *INFORMÁTICOS*, el atributo *Tipo* de la relación *EMPLEADOS* ha de tener un valor referente a empleados de tipo informáticos. Finalmente, si el código de empleado aparece en las dos tablas (permitido al existir solapamiento), el atributo *Tipo* debe referirse a los empleados que son a la vez jefes de proyecto e informáticos. No puede darse el caso de que un código de empleado solo aparezca en la relación *EMPLEADOS*, debido a la restricción de totalidad de la jerarquía.
- Exclusividad y totalidad de la jerarquía *INFORMÁTICO*: una aserción permitirá confirmar la exclusividad de la jerarquía *INFORMÁTICO*, donde se compruebe que si un código de empleado se encuentra en la relación *PROGRAMADORES*, el atributo *Tipo* de *INFORMÁTICOS* ha

de tener el valor adecuado. Así mismo, si no se encuentra en la relación *PROGRAMADORES*, el atributo *Tipo* ha de tener el valor referente a un empleado de tipo analista. Al igual que en el caso anterior, debido a la restricción de totalidad de la jerarquía, no se permite que el código de empleado solo aparezca en la relación correspondiente al supertipo de la jerarquía.

La dependencia en identificación **Se compone** se transforma mediante el mecanismo de propagación de clave y habría que comprobar con una aserción que todo proyecto tiene al menos una fase, es decir, que todo código de proyecto de la relación *PROYECTOS* se refiera en alguna de las ocurrencias de la relación *FASES*.

La interrelación N:M **Trabaja**, se ha transformado en una nueva relación *TRABAJAN* cuya clave primaria es la concatenación de los AIP de las entidades que intervienen en la interrelación. De la misma forma, la interrelación **Dirigido** se ha convertido en otra relación, *DIRIGIDOS*, pues, a pesar de ser 1:N, tiene atributos propios. Esta relación tiene como clave primaria únicamente el código de proyecto, pues cada proyecto solo puede estar dirigido por un jefe de proyecto.

Habría que comprobar que todo jefe de proyecto lo es de al menos un proyecto (aserción que compruebe que todo código de empleado en la relación *JEFES_PROYECTO* existe al menos una vez en la relación *DIRIGIDOS*); que todo informático trabaja al menos en un proyecto (aserción que compruebe que todo código de empleado en la relación *INFORMÁTICOS* existe al menos una vez en la relación *TRABAJAN*) y que todo proyecto tiene asignado al menos un informático (aserción que compruebe que todo código de proyecto en la relación *PROYECTOS* al menos existe una vez en la relación *TRABAJAN*). Además, tal y como aparece representado en el diagrama E/R, también habría que comprobar, para evitar la pérdida de semántica, que todo programador es experto al menos en un lenguaje, pues el atributo *Lenguajes* no es opcional. Esto debe implementarse mediante una aserción que compruebe que todo código de empleado en la relación *PROGRAMADORES* existe al menos una vez en la relación *EXPERTOS*.

En cuanto a los borrados y las modificaciones, se ha considerado como norma y dado que no se dice nada en contra en el enunciado, que el borrado sea sin acción para evitar borrados en cascada en tuplas que puedan ser críticas para la gestión de la empresa, en todos los casos excepto en las transformaciones de

la interrelación en identificación y en las jerarquías, que se ha considerado el mecanismo en cascada como el más coherente.

PARTE 2

En la Figura 2.50 se muestra el diagrama E/R que representa información sobre los productos generados en cada fase de un proyecto y los informáticos involucrados en su elaboración.

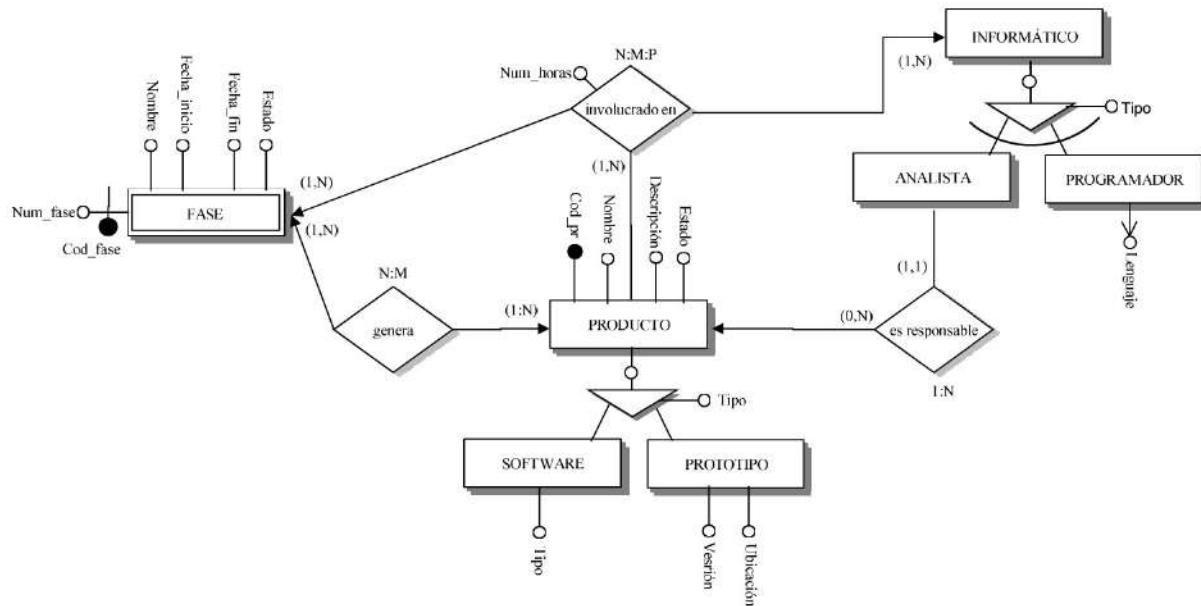


Figura 2.50. Diagrama E/R sobre los productos generados en cada fase de un proyecto

La transformación de este diagrama se muestra en la Figura 2.51.

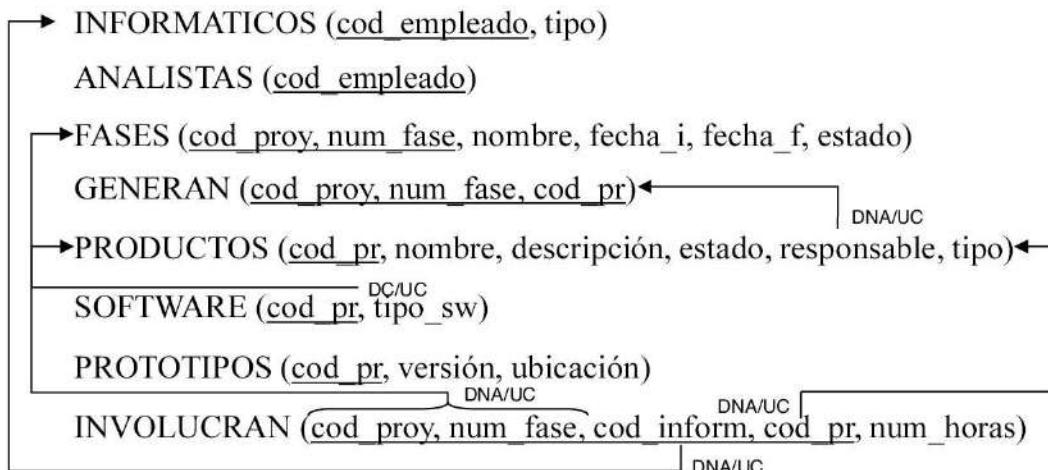


Figura 2.51. Diagrama E/R relativo a los productos generados en cada fase de un proyecto

De nuevo se ha procedido a transformar la jerarquía *PRODUCTO* en relaciones tanto para el supertipo como para los subtipos, debido a que el primero se interrelaciona con varias entidades y los segundos tienen atributos propios. Además en esta parte, al estar la entidad *ANALISTA* interrelacionada con el supertipo *PRODUCTO*, surge la necesidad de incluir una nueva relación para este subtipo. De manera análoga a lo que sucedía en el caso anterior, es necesario comprobar con una aserción la totalidad de la jerarquía *PRODUCTO* teniendo en cuenta la presencia del código de producto en las tablas referentes a los subtipos de la jerarquía.

La interrelación N:M **Genera** entre *FASE* y *PRODUCTO* se ha transformado en una relación *GENERAN*, en la que habría que comprobar que toda fase genera al menos un producto y que todo producto es generado al menos en una fase, debido a las cardinalidades mínimas uno entre ambas entidades. Esto se realiza mediante dos aserciones.

La interrelación 1:N **Es responsable**, entre las entidades *PRODUCTO* y *ANALISTA*, se transforma mediante el mecanismo de propagación de clave, incluyendo un campo *responsable*, dentro de la relación *PRODUCTOS*. Esta solución implica que todo producto tiene como responsable uno y solo un analista, como se indican en las cardinalidades del diagrama E/R.

La interrelación ternaria N:M:P **Involucrado_en**, se debe transformar en una nueva relación *INVOLUCRAN* cuya clave primaria debe ser la concatenación de las claves primarias de las relaciones que provienen de las entidades asociadas con la interrelación, debido al grado de correspondencia. En este caso se debería comprobar que toda fase (ya que toda fase genera al menos un producto) involucre a un informático y que el producto generado en dicha fase sea el mismo que aparece en la relación *INVOLUCRAN*. Que todo informático debe ser involucrado al menos en la generación de un producto y en una fase. El atributo de la interrelación ternaria se convierte en un atributo no opcional de la nueva relación *INVOLUCRAN*.

PARTE 3

En la Figura 2.52 se muestra el diagrama E/R que representa información sobre los gastos generados por los empleados asociados a cada proyecto y los recursos asignados a cada fase de un proyecto.

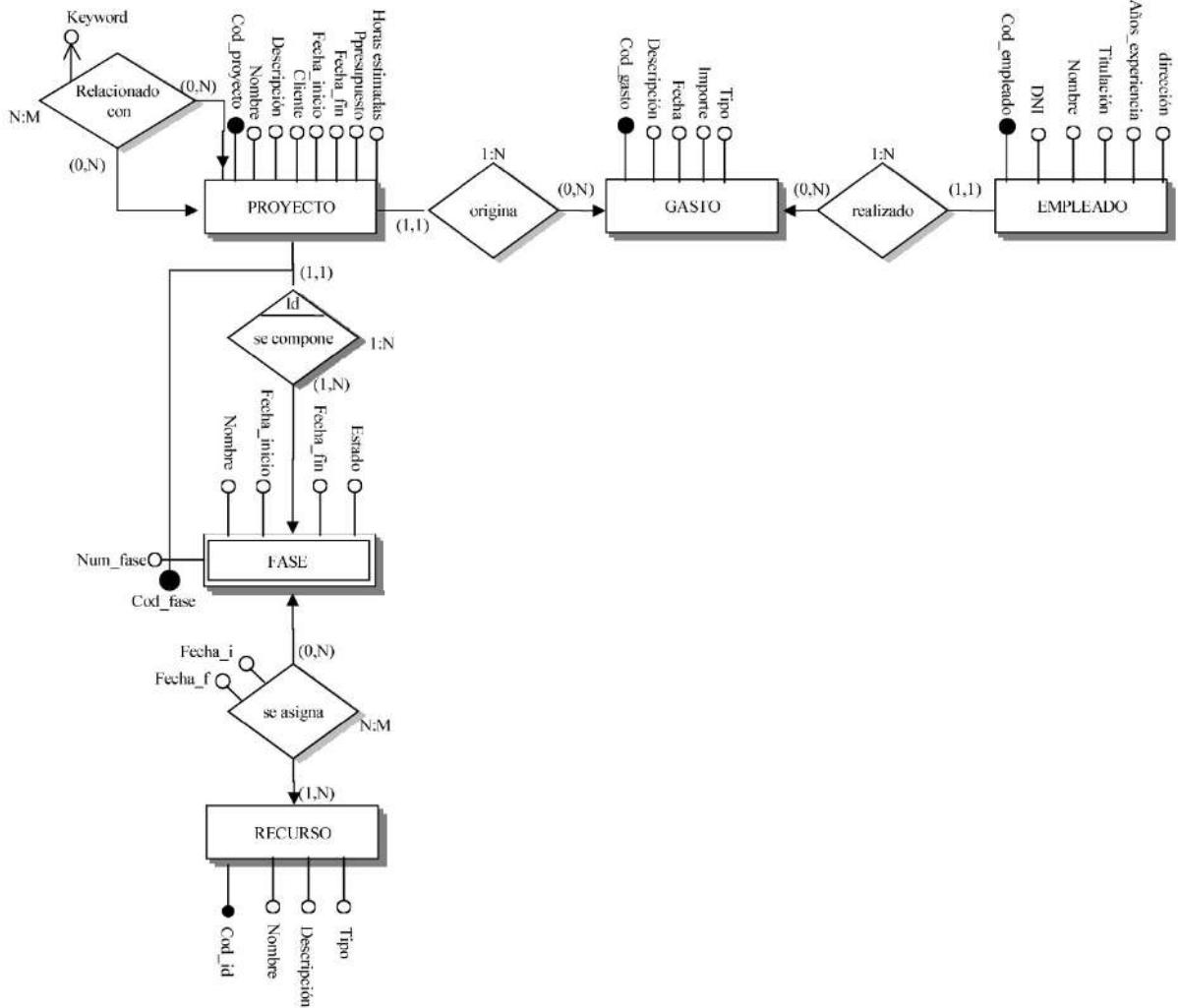


Figura 2.52. Diagrama E/R sobre gastos imputables a proyectos

La transformación al Modelo Relacional de este diagrama, aparece representado en la Figura 2.53.

En el grafo se puede ver cómo todo lo relativo a los gastos originados por los empleados en los proyectos se ha transformado por el mecanismo de propagación de clave, pues las interrelaciones eran 1:N. Se debería comprobar además que todo gasto adscrito a un determinado proyecto, sea originado por un empleado que trabaje en dicho proyecto y no en otro (aserción).

La interrelación reflexiva **Relacionado_con**, se ha transformado en una nueva relación **RELACIONADOS**, cuya clave primaria está compuesta por los códigos de proyecto y código de proyecto relacionado, además de por el campo *keyword* debido a que, al ser un atributo multivaluado, éste ha de incluirse como

parte de la clave primaria para evitar tuplas repetidas por cada par de proyectos relacionados que tengan una palabra clave distinta.

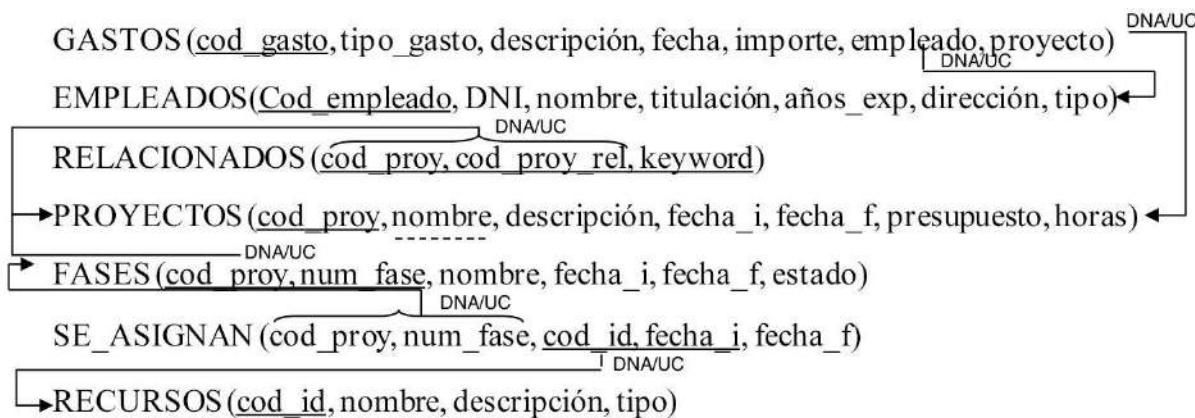


Figura 2.53. Grafo relacional relativo a gastos imputables a proyectos

La interrelación **Se_asigna**, pasa a convertirse en la relación **SE_ASIGNAN**, cuya clave primaria es el identificador del recurso y la fecha de asignación, pues con ello queda plenamente identificada la asignación de un recurso, teniendo en cuenta el supuesto semántico 17 del esquema E/R donde se indica que cada recurso solo puede estar asignado a una fase de un proyecto en un momento dado. Por otro lado, también se podría indicar como clave alternativa la combinación de los atributos *Cod_id* relativo al recurso más la fecha de fin de asignación del recurso. Habría que comprobar que al menos hay un recurso asignado a cada fase de un proyecto (aserción).

Por último, y como no se ha explicitado nada en el enunciado, los borrados se consideran sin acción y las modificaciones en cascada.

TRANSFORMACIÓN DE LOS SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

En cuanto a los supuestos semánticos que no se pudieron incorporar en el diagrama E/R final, su transformación se detalla en la siguiente tabla:

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
FASE.Estado = {en curso, finalizada}.	<i>CHECK</i> sobre la tabla <i>FASES</i> .
PRODUCTO.Estado = {si, no}.	<i>CHECK</i> sobre la tabla <i>PRODUCTOS</i> .
PRODUCTO.Tipo = {software, prototipo, informe técnico}.	<i>CHECK</i> sobre la tabla <i>PRODUCTOS</i> .
EMPLEADO.Tipo = {jefe proyecto, informático, null}.	<i>CHECK</i> sobre la tabla <i>EMPLEADOS</i> .
INFORMÁTICO.Tipo = {analista, programador}.	<i>CHECK</i> sobre la tabla <i>INFORMÁTICOS</i> .
RECURSO.Tipo = {hw, sw}.	<i>CHECK</i> sobre la tabla <i>RECURSOS</i> .
PROYECTO.Fecha_fin >= PROYECTO.Fecha_inicio.	<i>CHECK</i> sobre la tabla <i>PROYECTOS</i> .
PROYECTO.Horas estimadas <= PROYECTO.Fecha_fin - PROYECTO.Fecha_inicio.	<i>CHECK</i> sobre la tabla <i>PROYECTOS</i> .
FASE.Fecha_fin >= FASE.Fecha_inicio.	<i>CHECK</i> sobre la tabla <i>FASES</i> .
Comprobar que los intervalos de fechas en la entidad <i>FASE</i> estén incluidos en el intervalo compuesto por la fecha de inicio y la fecha de fin del proyecto.	<i>ASERCIÓN</i> sobre las tablas <i>PROYECTOS</i> y <i>FASES</i> .

Comprobar que la suma de todas las fechas correspondientes al conjunto de fases de un determinado proyecto, no exceda la fecha de finalización del mismo.	<i>ASERCIÓN</i> sobre las tablas <i>PROYECTOS</i> y <i>FASES</i> .
Dirigido.Num_horas = PROYECTO.Fecha_fin - PROYECTO.Fecha_inicio.	<i>ASERCIÓN</i> sobre las tablas <i>PROYECTOS</i> y <i>DIRIGIDOS</i> .
Trabaja.Num_horas = PROYECTO.Fecha_fin - PROYECTO.Fecha_inicio.	<i>ASERCIÓN</i> sobre las tablas <i>PROYECTOS</i> y <i>TRABAJAN</i> .
Involucrado.Num_horas =< FASE.Fecha_fin - FASE.Fecha_inicio.	<i>ASERCIÓN</i> sobre las tablas <i>INVOLUCRAN</i> y <i>FASES</i> .
Se_asigna.Fecha_f >= Se- _asigna.Fecha_i	<i>CHECK</i> sobre la tabla <i>SE_ASIGNAN</i> .
Comprobar que el intervalo de tiempo de la interrelación Se asigna está contenido en el compuesto por las fechas de inicio y de fin de la fase correspondiente.	<i>ASERCIÓN</i> sobre las tablas <i>SE_ASIGNAN</i> y <i>FASES</i> .
Los periodos de tiempo de la interrelación Se asigna no pueden solaparse para el mismo recurso.	<i>CHECK</i> sobre la tabla <i>SE_ASIGNAN</i> .
Un informático ha de aparecer en la interrelación Trabaja para todos los proyectos en los que aparece como Involucrado en alguna de sus fases.	<i>ASERCIÓN</i> sobre las tablas <i>TRABAJAN</i> e <i>INVOLUCRAN</i> .
Un analista ha de aparecer en la interrelación Involucrado para todos los productos en los que aparece como responsable del producto.	<i>ASERCIÓN</i> sobre las tablas <i>INVOLUCRAN</i> y <i>PRODUCTOS</i> .

PROYECTO.Fecha_inicio =<GASTO.Fecha =< PROYECTO.Fecha_fin.	<i>ASERCIÓN</i> sobre las tablas <i>PROYECTOS</i> y <i>GASTOS</i> .
Controlar que el empleado que realiza el gasto esté trabajando en el proyecto al que se carga dicho gasto.	<i>ASERCIÓN</i> sobre las tablas <i>TRABAJAN</i> y <i>GASTOS</i> .
El presupuesto de un proyecto siempre ha de ser mayor o igual a la suma de todos los costes de sus empleados más los gastos que generan.	<i>ASERCIÓN</i> sobre las tablas <i>PROYECTOS</i> , <i>TRABAJAN</i> y <i>GASTOS</i> .
Un proyecto no está relacionado consigo mismo.	<i>CHECK</i> sobre la tabla <i>RELACIONADOS</i> .

PROBLEMA 2.15: MEDICAMENTOS

Dado el esquema E/R propuesto como solución en el Problema 1.4 del Capítulo 1 que versa sobre la gestión de los medicamentos en un hospital.

Se pide:

Transformar el esquema E/R propuesto como solución en el capítulo al grafo relacional teniendo en cuenta los supuestos semánticos que no se pudieron incorporar al diagrama E/R. Todos los supuestos semánticos que no sea posible incluir en el grafo relacional se deberán definir como *checks*, aserciones o disparadores.

DISCUSIÓN DEL ENUNCIADO

La transformación del esquema E/R se ha dividido en varias partes con el objetivo de facilitar la comprensión del grafo relacional al lector.

PARTE 1

En la Figura 2.54 se muestra el diagrama E/R que trata los ingresos de los pacientes en los diferentes servicios del hospital. En la Figura 2.55 se muestra el grafo relacional que surge de la transformación del diagrama E/R.

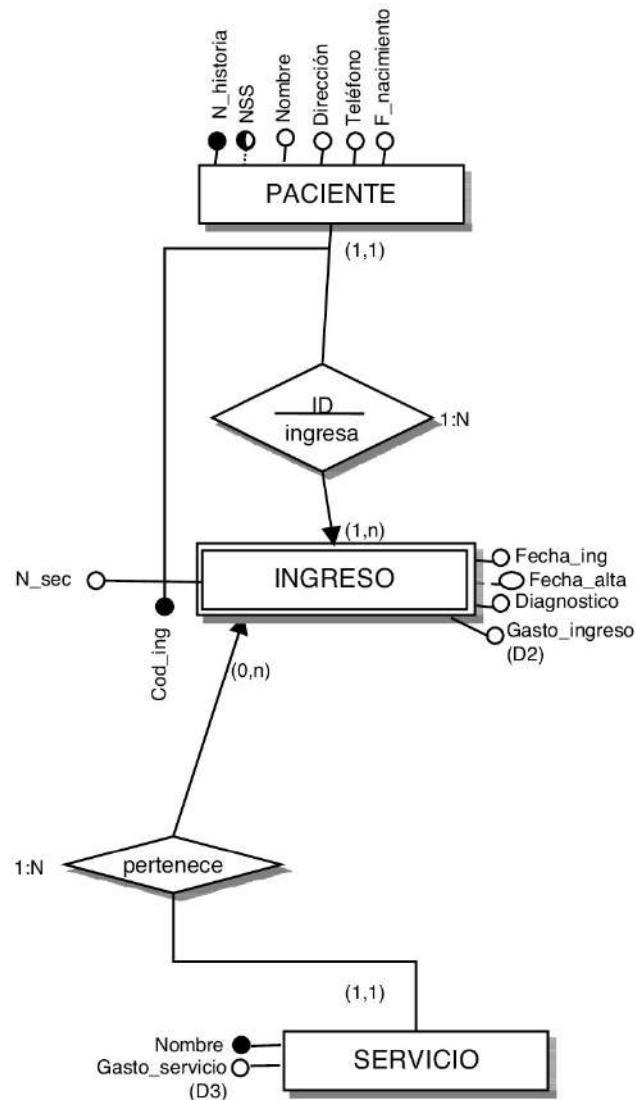


Figura 2.54. Diagrama E/R sobre ingresos de pacientes en los servicios del hospital

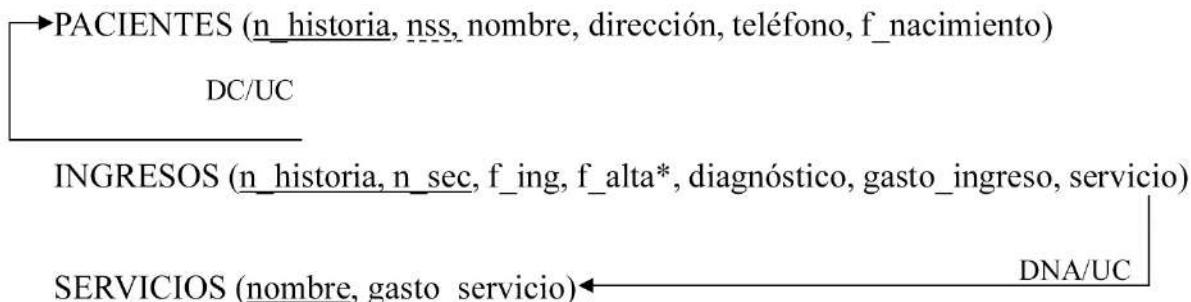


Figura 2.55. Grafo relacional relativo a ingresos de pacientes en los servicios del hospital

La interrelación en identificación Ingresa se ha transformado mediante una propagación de clave, donde la clave primaria de la relación *PACIENTES* forma parte de la clave primaria de la relación *INGRESOS*. El borrado y la modificación de esta clave ajena en la relación *INGRESOS* se han definido en cascada, ya que una ocurrencia en la relación *INGRESOS* no tiene sentido si no existe previamente una ocurrencia de *PACIENTES* de ese ingreso. El campo *fecha_alta* de la relación *INGRESOS* se ha marcado como un campo opcional y el campo *nss* de la relación *PACIENTES* se ha marcado como clave alternativa en la relación. En la transformación al grafo relacional de esta interrelación se ha perdido la semántica de que todo paciente en nuestra base de datos ha de haber estado ingresado al menos una vez. Esta semántica será necesario comprobarla mediante una asercción en el ámbito de las tablas *PACIENTES* e *INGRESOS*. Además no se ha podido recoger información sobre el cálculo de los atributos derivados de las entidades *SERVICIO* e *INGRESO*, por lo que será necesario crear disparadores que calculen automáticamente el valor de los campos *gasto_servicio* y *gasto_ingreso* respectivamente.

Por otro lado, la interrelación 1:N **Pertenece** también se ha transformado mediante propagación de clave, pero esta vez el atributo propagado no formará parte de la clave primaria de la relación *INGRESOS*. Para la clave ajena *servicio* que referencia a la relación *SERVICIOS* se han definido los borrados sin acción y las modificaciones en cascada (se han tomado las opciones menos restrictivas). En esta transformación no se ha producido pérdida semántica.

PARTE 2

El diagrama E/R correspondiente a los fármacos que se consumen de forma general en los servicios se muestra en la Figura 2.56. La transformación de este diagrama se presenta en la Figura 2.57

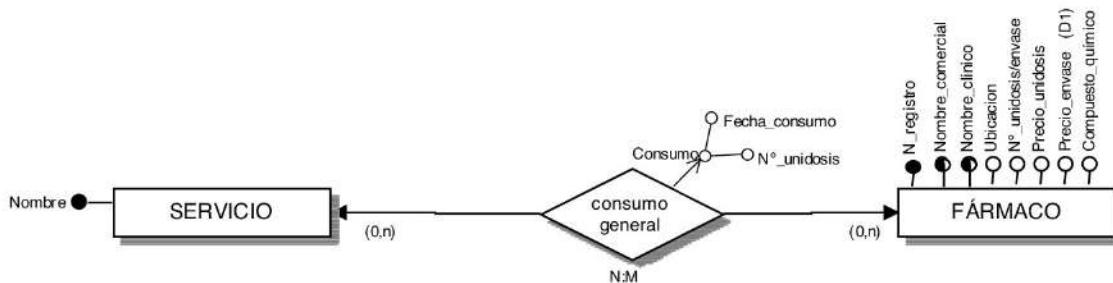


Figura 2.56. Diagrama E/R sobre los fármacos consumidos por servicio

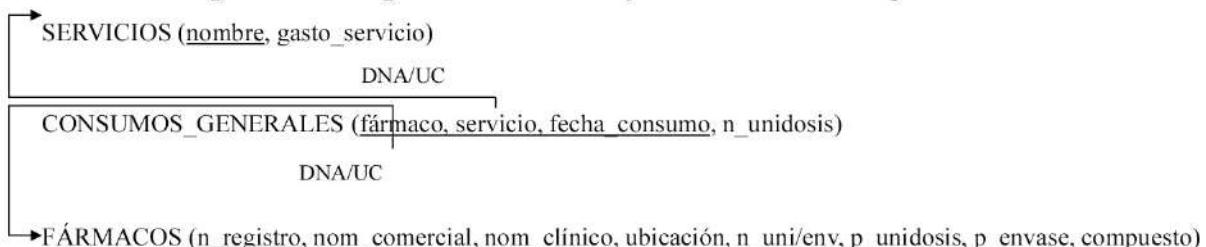


Figura 2.57. Grafo relacional relativo a la transformación del esquema de la Figura 2.56

La entidad **FÁRMACO** se ha transformado en la relación **FÁRMACOS** identificado por la columna *n_registro*. Las columnas *nom_comercial* y *nom_clínico* se han declarado claves alternativas de la relación. En el grafo no se ha podido incorporar la información del cálculo del atributo derivado *Precio_envase*, será necesario añadir un disparador que se encargue automáticamente de realizar este cálculo.

La interrelación N:M **Consumo_general** se transforma en la relación **CONSUMOS_GENERALES** donde la clave primaria está formada por las claves primarias de las relaciones que une más la columna *fecha_consumo*. No se produce pérdida de semántica en la transformación de esta interrelación.

PARTE 3

La Figura 2.58 muestra el diagrama E/R correspondiente a los facultativos que trabajan en los diferentes servicios de la universidad y emiten informes sobre ingresos. El grafo relacional derivado de la transformación de este diagrama se muestra en la Figura 2.59.

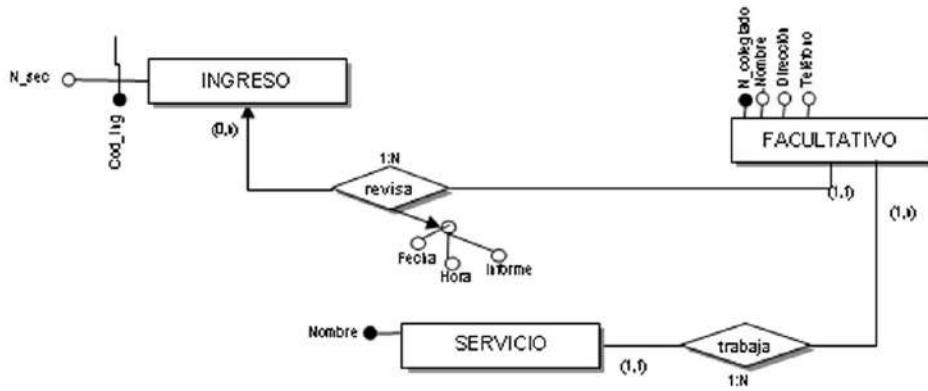


Figura 2.58. Diagrama E/R sobre facultativos que trabajan en los servicios del hospital

→**INGRESOS** (n_historia, n_sec, f_ing, f_alta*, diagnóstico, gasto_ingreso, servicio)

DNA/UC

REVISIONES (n_historia, n_sec, fecha, hora, informe, **facultativo**)

DNA/UC

SERVICIOS (nombre, gasto_servicio) ←

DNA/UC

FACULTATIVOS (n_colegiado, Nombre, Dirección, Teléfono, **servicio**) ←

Figura 2.59. Grafo relacional relativo a los facultativos que trabajan en los servicios del hospital

Al transformar la interrelación 1:N **Trabaja** mediante propagación de clave, se hace necesario añadir una nueva columna a la tabla **FACULTATIVOS**, la clave ajena **servicio**. Los borrados y modificaciones de esta clave ajena se definen poco restrictivos (borrado sin acción y modificación en cascada). Se produce pérdida semántica en la transformación, ya que no se puede indicar en el grafo relacional que todo servicio ha de tener al menos un facultativo asociado. Esta pérdida semántica se puede comprobar mediante una aserción.

La transformación de la interrelación 1:N **Revisa**, sin embargo, se transformará en una nueva relación debido a la posesión del atributo derivado *Revisión*. La clave primaria de la nueva relación **REVISAN** se compone de la fecha y hora de la revisión y del número de historia clínica. No es en este caso necesario incluir información en la clave primaria sobre el facultativo que revisa el ingreso, ya que siempre es el mismo facultativo para el ingreso. De igual

forma, la columna *informe* tampoco forma parte de la clave primaria de la relación. Las restricciones de integridad de las claves ajenas en esta relación se han definido poco restrictivas (borrados sin acción y modificaciones en cascada). No se produce pérdida semántica al transformar esta interrelación.

PARTE 4

Para finalizar, el diagrama E/R de la Figura 2.60 muestra información sobre el consumo de fármacos por cada ingreso recetado por los facultativos. El grafo relacional que surge a partir de este diagrama E/R se muestra en la Figura 2.61.

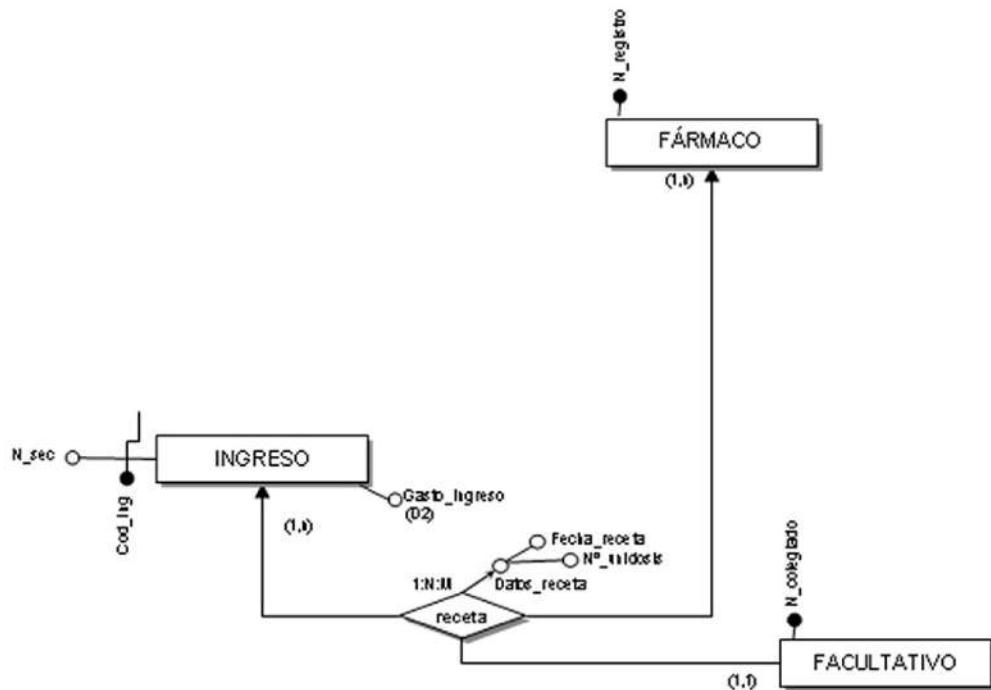


Figura 2.60. Diagrama E/R sobre los fármacos recetados por los facultativos a los pacientes

→INGRESOS (n_historia, n_sec, ...)

FACULTATIVOS (n_colegiado, ...)

→FÁRMACOS (n_registro, nom_comercial, nom_clínico, ubicación, n_uni/env, p_unidosis, p_envase, compuesto)

DNA/UC

DNA/UC

RECETAS (ingreso, fármaco, facultativo, fecha, n_unidosis)

Figura 2.61. Grafo relacional relativo a los fármacos recetados por los facultativos a los pacientes

La interrelación ternaria **Receta** se transforma en una nueva relación **RECETAN** cuya clave primaria está formada por la unión de los atributos *ingreso, fármaco y fecha*. La columna *facultativo* no forma parte de la clave primaria ya que para un fármaco recetado a un ingreso en una fecha determinada solo ha podido ser prescrito por un facultativo, tal y como indica la interrelación **Receta**.

TRANSFORMACIÓN DE SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

La transformación de los supuestos semánticos que no se pudieron recoger en el diagrama E/R del enunciado se resume en la siguiente tabla:

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
En la entidad <i>INGRESO</i> siempre se ha de cumplir que el valor del atributo <i>Fecha_ing</i> es anterior al valor del atributo <i>Fecha_alta</i> en el caso de que sea conocido.	<i>CHECK</i> en la tabla <i>INGRESOS</i> .
La fecha de nacimiento de un paciente siempre ha de ser anterior o igual a la fecha de ingreso de ese paciente en el hospital.	<i>ASERCIÓN</i> sobre las tablas <i>PACIENTES</i> e <i>INGRESOS</i> .
En la interrelación Revisa se ha de cumplir que la fecha de la revisión ha de encontrarse entre las fechas de inicio y alta de un ingreso. En el caso de que la fecha de alta del ingreso no se conozca, se supondrá que la fecha de la revisión ha de ser anterior a la actual y posterior a la fecha de inicio del ingreso.	<i>ASERCIÓN</i> en el ámbito de las tablas <i>REVISAN</i> e <i>INGRESOS</i> .
En la interrelación Receta se ha de cumplir que la fecha de la receta, al igual que ocurría con la fecha de la revisión, ha de encontrarse entre las fechas de ingreso del paciente.	<i>ASERCIÓN</i> en el ámbito de las tablas <i>RECETAN</i> e <i>INGRESOS</i> .
Cada vez que un facultativo revisa a un determinado paciente (en una fecha y hora) se crea un único informe.	Definición de clave primaria en la relación <i>REVISAN</i> .
Existe un único valor para el atributo <i>Nº_unidosis</i> para cada vez (fecha/hora/minuto) que un facultativo le receta un fármaco a un paciente ingresado.	Definición de clave primaria en la relación <i>RECETAN</i> .

<p>Lo mismo ocurre para la interrelación Consumo_general. La fecha de consumo de un fármaco en un servicio implica el número de unidosis consumidas.</p>	<p>Definición de clave primaria en la relación <i>CONSUMOS_GENERALES</i>.</p>
<p>En la entidad <i>FÁRMACO</i> el atributo <i>Precio_envase</i> se calculará según la fórmula: $Precio_{envase} = Precio_{unidosis} * N^o_{unidosis/ envase}$.</p>	<p><i>DISPARADOR</i> antes de insertar en la tabla <i>FÁRMACOS</i>, consulta la tabla <i>FÁRMACOS</i>.</p>
<p>En la entidad <i>INGRESO</i> el atributo <i>Gasto_ingreso</i> se calculará a partir de las ocurrencias de la interrelación Receta para ese ingreso. Se sumará el gasto de cada receta, calculando este gasto multiplicando el número de unidosis consumida por el valor del atributo <i>Precio_unidosis</i> de la entidad <i>FÁRMACO</i>.</p>	<p><i>DISPARADOR</i> antes de insertar en la tabla <i>INGRESOS</i>, consulta las tablas <i>RECETAN</i> y <i>FÁRMACOS</i>.</p>
<p>En la entidad <i>SERVICIO</i> el atributo <i>Gasto_servicio</i> se calculará sumando los gastos derivados de los ingresos (sumando el valor del atributo <i>Gasto_ingreso</i> de todos los ingresos pertenecientes al servicio) a los gastos derivados del consumo general del servicio en el hospital (derivados de la interrelación Consumo_general).</p>	<p><i>DISPARADOR</i> antes de insertar en la tabla <i>SERVICIOS</i>, consulta las tablas <i>INGRESOS</i>, <i>CONSUMOS_GENERALES</i> y <i>FÁRMACOS</i>.</p>

Se plantea como ejercicio al lector la generación del pseudocódigo asociado a los *checks*, aserciones y disparadores de este problema basándose en los ejemplos propuestos en el Problema 2.13 de este mismo capítulo.

PROBLEMA 2.16: PROYECTOS I+D

Dado el diagrama E/R de la Figura 2.62 correspondiente al diseño de una BD del Ministerio de Educación y Ciencia sobre los proyectos de investigación del Plan Nacional de I+D.

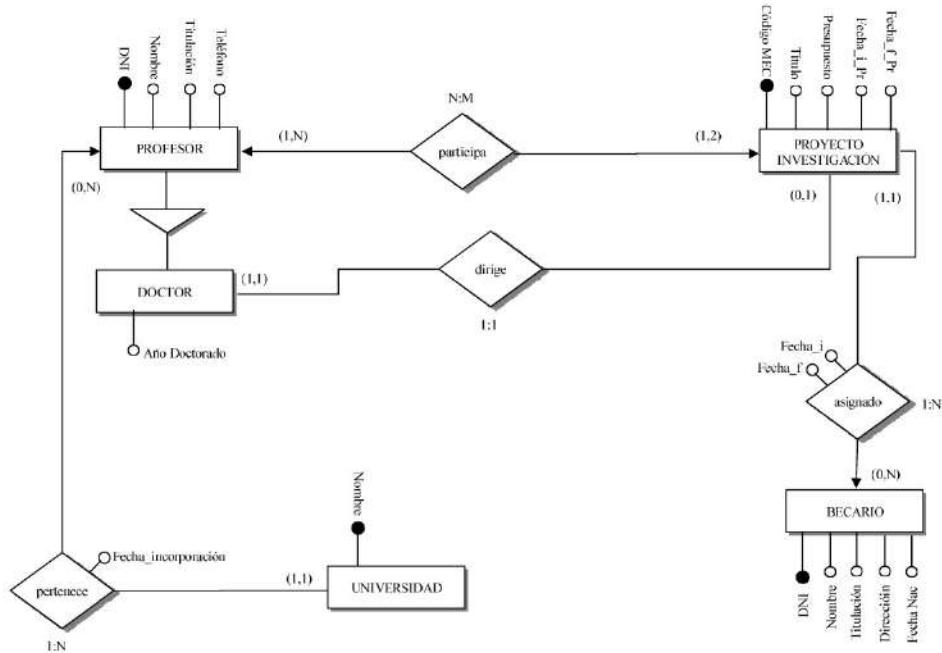


Figura 2.62. Esquema inicial E/R sobre proyectos I+D

Se pide:

Transformar el esquema E/R al Modelo Relacional e indicar todas las restricciones que no haya sido posible representar directamente en el Modelo Relacional.

PROPUESTA DE SOLUCIÓN

La Figura 2.63 muestra el grafo relacional completo.

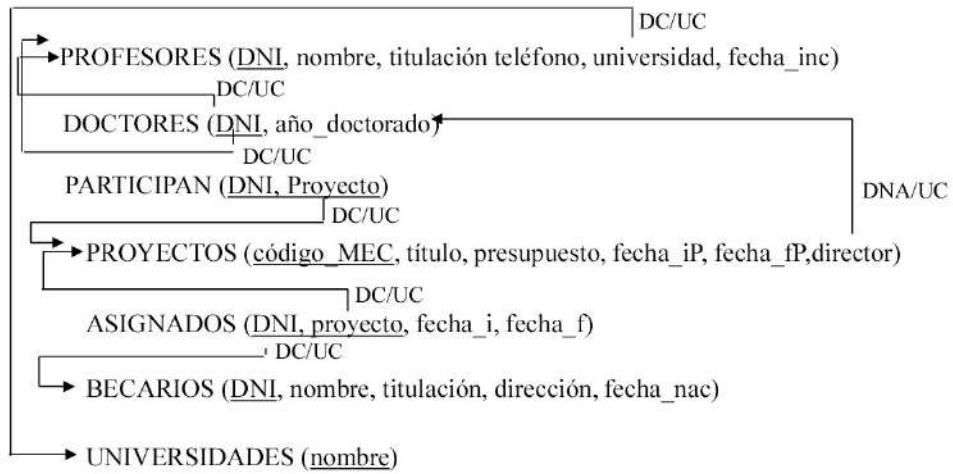


Figura 2.63. Grafo relacional completo

TRANSFORMACIÓN DE LOS SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

A pesar de no venir explicitados en el enunciado, los supuestos semánticos que no pudieron ser incluidos en el diagrama E/R se transformarían al grafo relacional de la siguiente forma:

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
La fecha inicio de un proyecto ha de ser anterior o igual en el tiempo que la fecha de finalización del mismo.	<i>CHECK</i> sobre la tabla <i>PROYECTOS</i> .
La fecha de nacimiento (<i>Fecha_nac</i>) de un becario nunca podrá ser posterior en el tiempo a la fecha de firma (<i>Fecha_i</i>) de la beca.	<i>ASERCIÓN</i> sobre las tablas <i>BECARIOS</i> y <i>ASIGNADOS</i> .
La fecha de inicio de una beca no puede ser posterior a la fecha de finalización de la misma.	<i>CHECK</i> sobre la tabla <i>ASIGNADOS</i> .
El intervalo que dura cada beca asignada a un proyecto debe estar incluido en el intervalo formado por las fechas de inicio y fin del proyecto.	<i>ASERCIÓN</i> sobre las tablas <i>PROYECTOS</i> y <i>ASIGNADOS</i> .
La fecha de incorporación de un profesor asignado a un proyecto nunca debe ser posterior a la fecha de concesión de un proyecto en el que participe.	<i>ASERCIÓN</i> sobre las tablas <i>PROFESORES</i> , <i>PROYECTOS</i> y <i>PARTICIPAN</i> .
La fecha de incorporación de un profesor doctor que dirige un proyecto nunca debe ser posterior a la fecha de concesión de un proyecto en el que participe.	<i>ASERCIÓN</i> sobre las tablas <i>DOCTORES</i> y <i>PROYECTOS</i> .

PROBLEMA 2.17: VUELTA CICLISTA

Dado el esquema E/R que trata del almacenamiento de información histórica relativa a distintas ediciones de la Vuelta Ciclista a España, representado en la Figura 2.64.

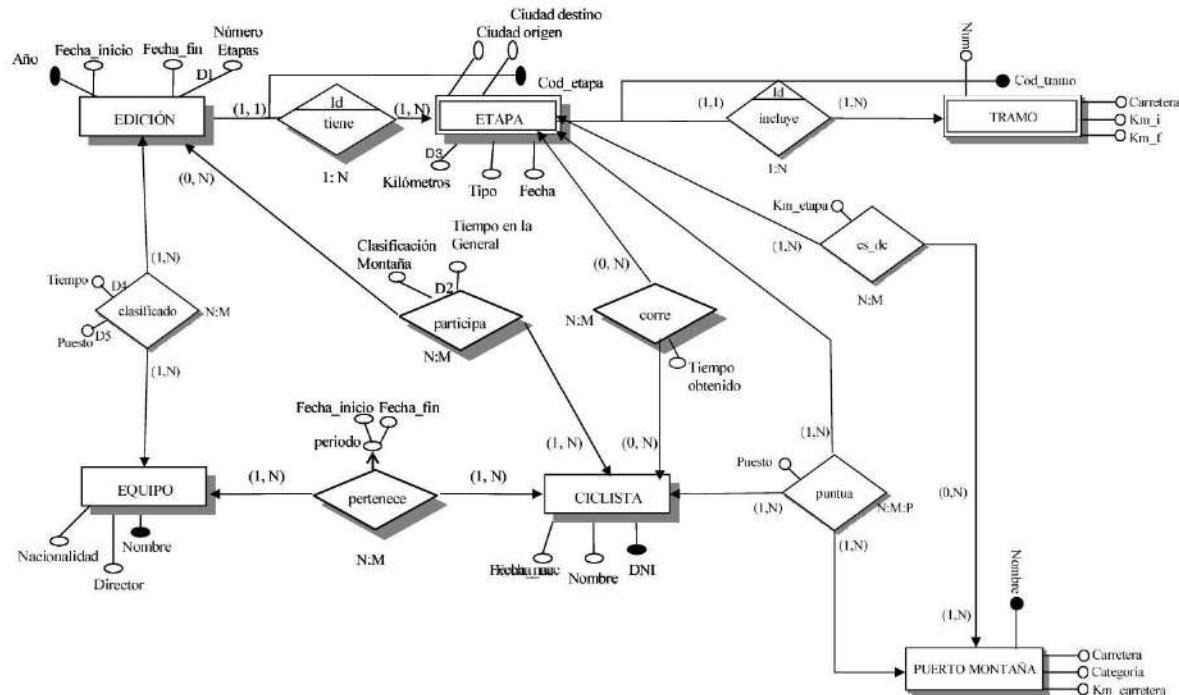


Figura 2.64. Diagrama E/R inicial sobre la vuelta ciclista

Se pide:

Transformar el esquema E/R al Modelo Relacional indicando todas las restricciones que no haya sido posible representar directamente en el modelo relacional. Para estas restricciones es necesario indicar cómo podrían controlarse mediante *checks*, aserciones o disparadores.

PROPUESTA DE SOLUCIÓN

La Figura 2.65 muestra el grafo relacional completo.

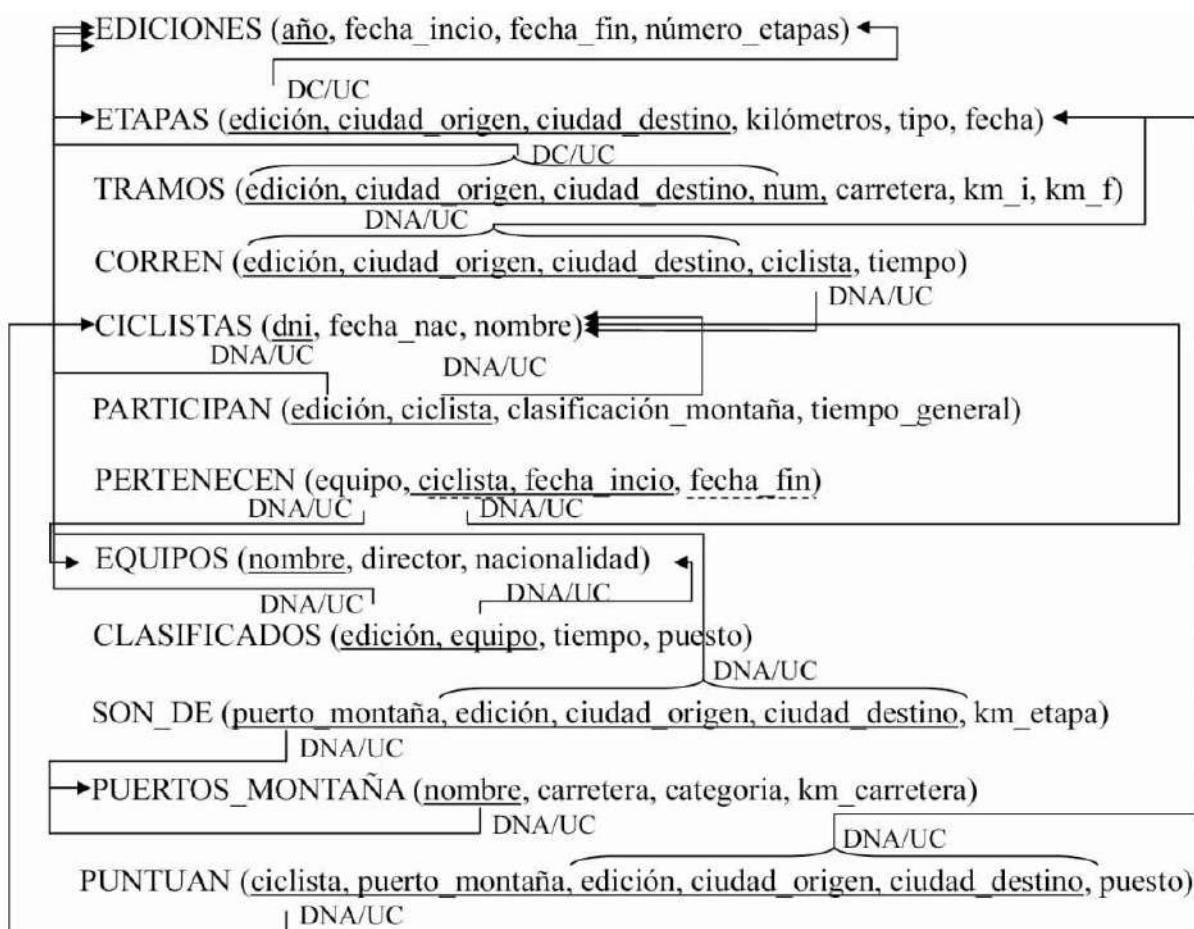


Figura 2.65. Grafo relacional completo

TRANSFORMACIÓN DE LOS SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

En cuanto a los supuestos semánticos que no se pudieron incorporar en el diagrama E/R final, su transformación se detalla en la siguiente tabla:

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
El atributo derivado D1 (<i>Número_etapas</i> de la entidad <i>EDICIÓN</i>) se calcula sumando todas las etapas de la edición de la vuelta ciclista (D1: Número Etapas = \sum etapas de la edición).	<i>DISPARADOR</i> antes de insertar en la tabla <i>EDICIONES</i> , consulta información de la tabla <i>ETAPAS</i> .

<p>El atributo derivado D2 (<i>Tiempo_en_la_general</i> de la interrelación PARTICIPA) se calcula sumando el tiempo obtenido por cada ciclista que corre en cada etapa (D2: Tiempo en la General = \sum Tiempo obtenido en cada etapa).</p>	<p><i>DISPARADOR</i> antes de insertar en la tabla <i>PARTICIPAN</i>, consulta datos de las tablas <i>CORREN</i> y <i>EDICIONES</i>.</p>
<p>El atributo derivado D3 (<i>Kilómetros</i> de la entidad <i>ETAPA</i>) se calcula sumando los kilómetros de cada tramo de la etapa de la vuelta ciclista.</p>	<p><i>DISPARADOR</i> antes de insertar en la tabla <i>ETAPAS</i>, consulta datos de la tabla <i>TRAMOS</i>.</p>
<p>El atributo derivado D4 (<i>Puesto</i> de la interrelación Clasificado) se calcula a partir del atributo derivado <i>Tiempo</i> de la misma interrelación, que almacena información sobre la suma de los tiempos de los tres primeros corredores del equipo en esa edición.</p>	<p><i>DISPARADOR</i> antes de insertar en la tabla <i>CLASIFICADOS</i>, consulta el nuevo dato derivado D5 de la misma tabla. Este disparador y el siguiente se implementarían en uno solo.</p>
<p>El atributo derivado D5 (<i>Tiempo</i> de la interrelación Clasificado) se calcula a partir del atributo <i>Tiempo_obtenido</i> de la interrelación Corre, sumando los tiempos de los tres primeros corredores del equipo en cada etapa de la edición.</p>	<p><i>DISPARADOR</i> antes de insertar en la tabla <i>CLASIFICADOS</i>, consulta datos de las tablas <i>PUNTUAN</i>, <i>CORREN</i> y <i>PERTENECEN</i>.</p>
<p>Para una misma etapa de la vuelta ciclista, todos los tramos que pertenecen a la misma etapa deberían seguir un número de secuencia consecutivo y que empiece por el número uno.</p>	<p><i>CHECK</i> sobre la tabla <i>TRAMOS</i>.</p>
<p>El atributo <i>Kilómetro_etapa</i> en la interrelación Es_de siempre tendrá que tener un valor menor a los kilómetros que se recorren en una determinada etapa.</p>	<p><i>ASERCIÓN</i> en el ámbito de las tablas <i>SON_DE</i> y <i>ETAPAS</i>.</p>

Los valores del atributo <i>Categoría</i> de la entidad <i>PUERTO_MONTAÑA</i> pertenecen al dominio <i>CATEGORÍA_PUERTO_MONTAÑA</i> con los valores {especial, 1 ^a , 2 ^a , 3 ^a }.	<i>CHECK</i> sobre la tabla <i>PUERTOS_MONTAÑA</i> .
Siempre se ha de cumplir que existe el mismo número de ciclistas que puntuán en el mismo puesto según la categoría del puerto de montaña para toda la vuelta ciclista.	<i>CHECK</i> sobre la tabla <i>PUNTÚAN</i> .
El valor del atributo <i>Fecha_fin</i> en la entidad <i>EDICIÓN</i> siempre ha de ser posterior o igual al valor del atributo <i>Fecha_inicio</i> de la misma entidad.	<i>CHECK</i> sobre la tabla <i>EDICIONES</i> .
El valor del atributo <i>Fecha_fin</i> en la interrelación Pertenece siempre ha de ser posterior o igual al valor del atributo <i>Fecha_inicio</i> de la misma entidad.	<i>CHECK</i> sobre la tabla <i>PERTENECEN</i> .
El valor del atributo <i>Fecha_nac</i> en la entidad <i>CICLISTA</i> siempre ha de ser posterior al valor del atributo <i>Fecha_inicio</i> de las ediciones de la vuelta ciclista donde participa y al valor del atributo <i>Fecha_inicio</i> de la pertenencia a equipos ciclistas.	<i>ASERCIÓN</i> sobre las tablas <i>CICLISTA</i> , <i>EDICIÓN</i> y <i>PARTICIPAN</i> . <i>ASERCIÓN</i> sobre las tablas <i>CICLISTA</i> y <i>PERTENECEN</i> .
Todo ciclista que corre en una etapa de una edición también ha de estar incluido en la interrelación Participa para la misma edición.	<i>ASERCIÓN</i> sobre las tablas <i>PARTICIPAN</i> , <i>CORREN</i> y <i>ETAPA</i> .
En una edición de la vuelta ciclista un ciclista solo puede pertenecer a un equipo.	<i>CHECK</i> sobre la tabla <i>PERTENECEN</i> .
Todo ciclista que corre en una etapa al menos ha de puntuar en cada puerto de montaña.	<i>ASERCIÓN</i> sobre la tabla <i>CORREN</i> , <i>SON_DE</i> y <i>PUNTÚAN</i> .

PROBLEMA 2.18: TARJETAS DESCUENTO

El esquema E/R de la Figura 2.66 corresponde a una BD que almacena la información actual sobre las tarjetas de descuento que ofrecen diferentes entidades (por ejemplo, Tarjetas de Estudiante de las Universidades, la Tarjeta Joven <26 de la Comunidad de Madrid, etc.) y qué comercios/lugares de ocio (tiendas, cines, etc.) ofrecen estos descuentos. También almacena información sobre las personas que poseen este tipo de tarjetas (NIF, sexo y fecha de nacimiento).

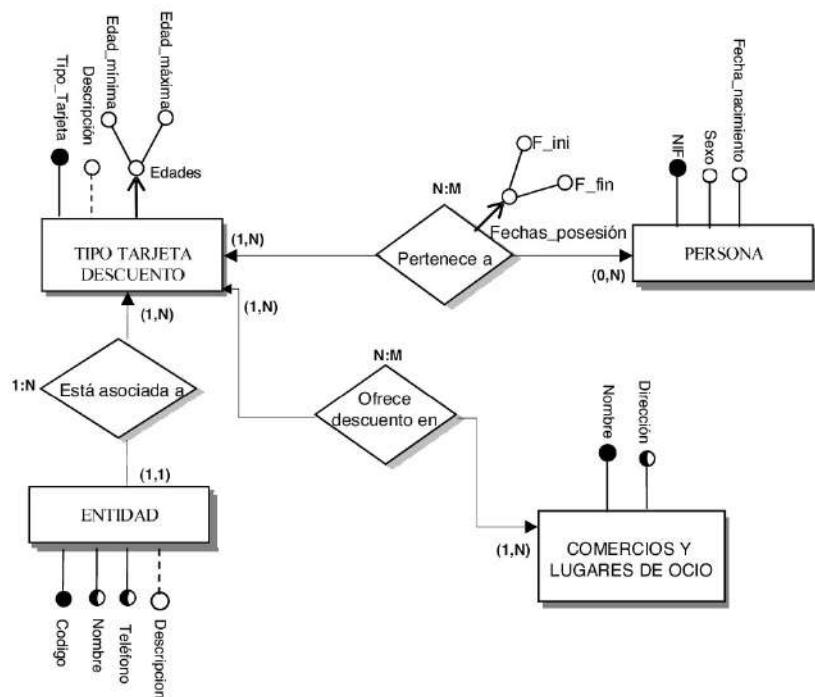


Figura 2.66. Diagrama E/R sobre tarjetas descuento

Se pide:

Transformar el esquema E/R al Modelo Relacional indicando todas las restricciones que no haya sido posible representar directamente en el modelo relacional. Para estas restricciones es necesario indicar cómo podrían controlarse.

PROPUESTA DE SOLUCIÓN

La Figura 2.67 muestra el grafo relacional completo. Las opciones de borrado y modificación se han elegido sin acción y en cascada respectivamente por ser las opciones que permiten controlar los borrados al no decirse nada explícito en el enunciado.

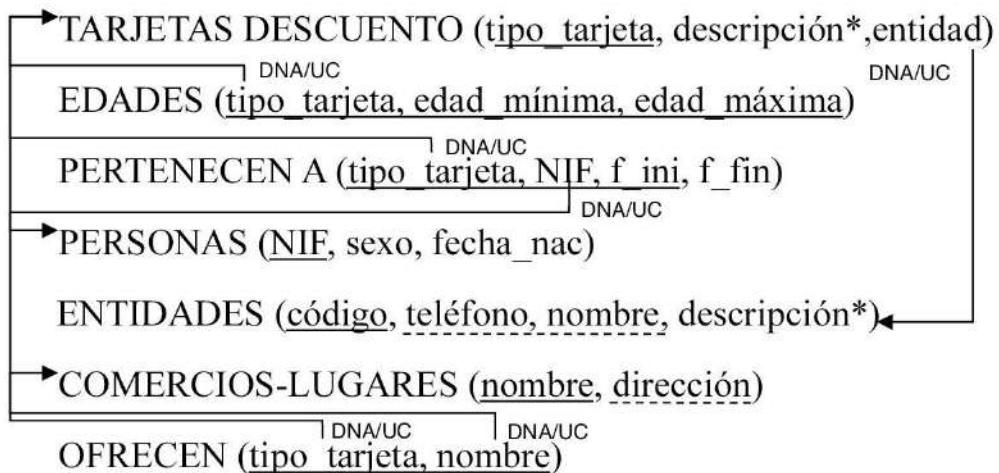


Figura 2.67. Esquema relacional correspondiente a las tarjetas descuento

TRANSFORMACIÓN DE LOS SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
La edad de una persona (Fecha actual-Fecha de nacimiento) debe estar incluida en el rango de edades de todas las tarjetas descuento que posee.	<i>ASERCIÓN</i> en el ámbito de las tablas <i>PERSONAS</i> , <i>PERTENECEN</i> , <i>EDADES</i> y <i>TARJETAS DESCUENTO</i> .
La fecha de nacimiento de una persona debe ser anterior a la fecha de inicio de pertenencia de cada una de sus tarjetas.	<i>ASERCIÓN</i> en el ámbito de las tablas <i>PERSONAS</i> y <i>PERTENECEN</i> .
La fecha de inicio de pertenencia de una tarjeta debe ser menor o igual que la fecha de fin.	<i>CHECK</i> en la tabla <i>PERTENECEN</i> .
La edad mínima del atributo multivaluado <i>Edades</i> debe ser inferior a la edad máxima.	<i>CHECK</i> en la tabla <i>EDADES</i> .

PROBLEMA 2.19: AGENCIA DE CASTINGS

Dado el esquema E/R representado en la Figura 2.68.

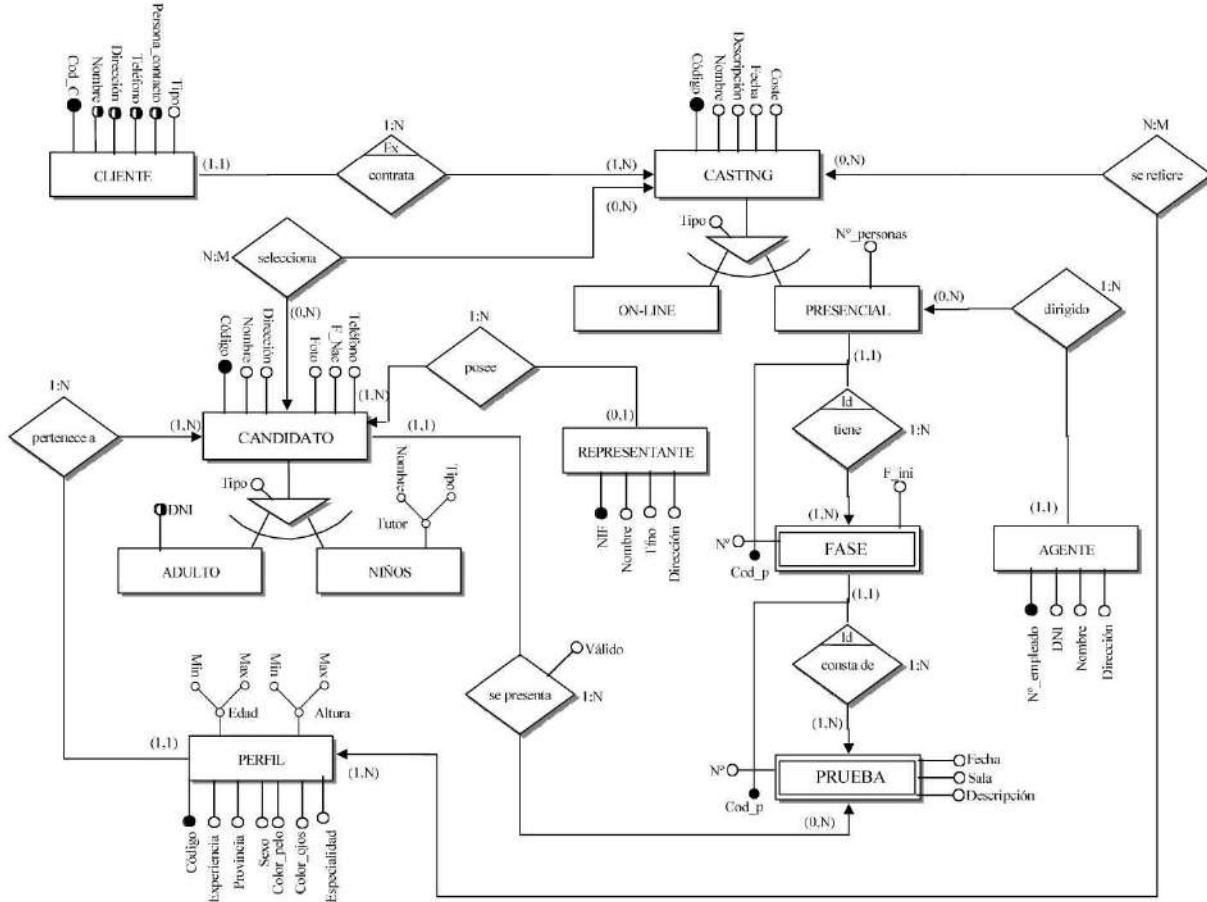


Figura 2.68. Diagrama E/R sobre agencia de castings

Se pide:

Transformar el esquema E/R al Modelo Relacional (grafo relacional) indicando toda la semántica que no se ha podido incorporar en el grafo en forma de checks, aserciones y disparadores.

PROPUESTA DE SOLUCIÓN

La Figura 2.69 muestra el grafo relacional completo. Las opciones de borrado y modificación se han elegido sin acción y en cascada respectivamente por ser las opciones que permiten controlar los borrados al no decirse nada explícito en el enunciado.

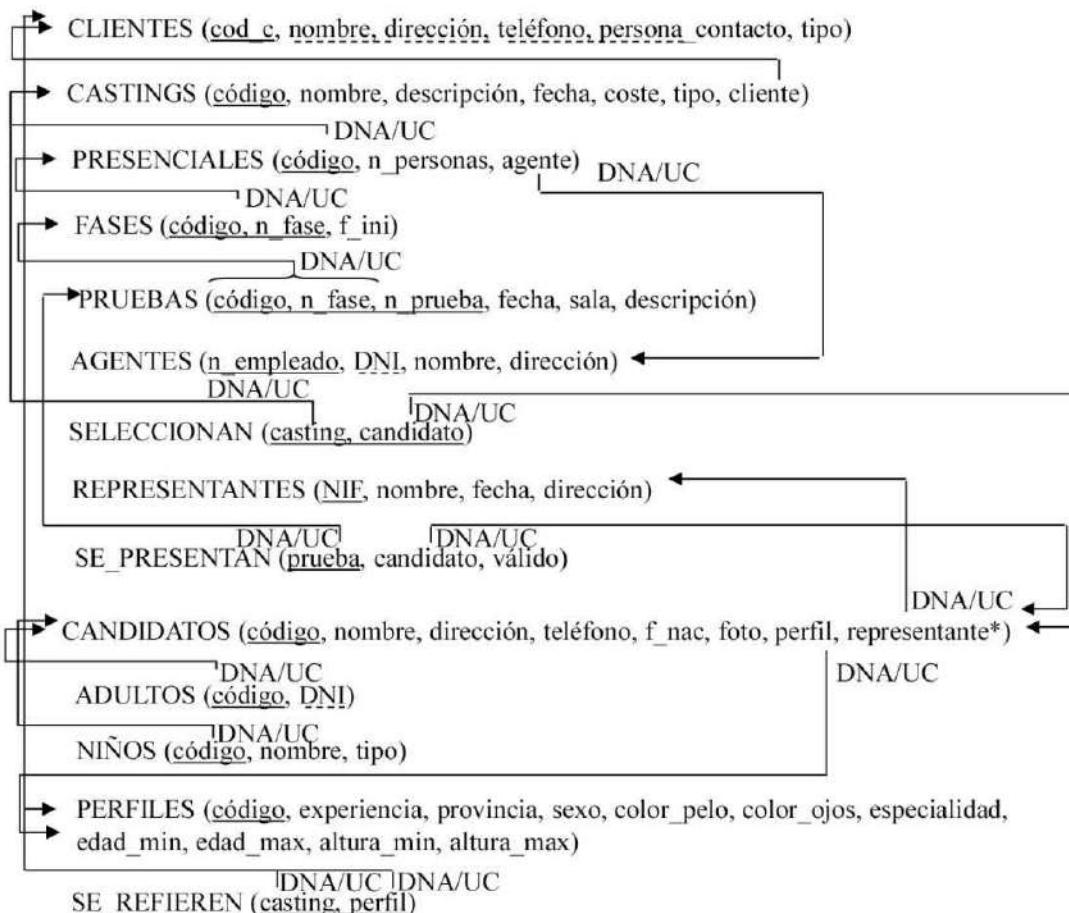


Figura 2.69. Esquema relacional sobre agencia de castings

TRANSFORMACIÓN DE LOS SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

En cuanto a los supuestos semánticos que no se pudieron incorporar en el diagrama E/R final, su transformación se detalla en la siguiente tabla:

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
CLIENTE.Tipo = {publicidad-cine, moda}.	<i>CHECK</i> sobre la tabla <i>CLIENTES</i> .
CANDIDATO.NIÑO.Tutor = {padre, madre}.	<i>CHECK</i> sobre la tabla <i>NIÑOS</i> .
PERFIL.Experiencia = {si, no}.	<i>CHECK</i> sobre la tabla <i>PERFILES</i> .
PERFIL.Especialidad = {modelo, actor}.	<i>CHECK</i> sobre la tabla <i>PERFILES</i> .

Se presenta. Valido = {si, no}.	<i>CHECK</i> sobre la tabla <i>SE_PRESENTAN</i> .
CASTING.Tipo = {presencial, online}.	<i>CHECK</i> sobre la tabla <i>CASTINGS</i> .
FASE.F_ini >= CASTING.Fecha.	<i>ASERCIÓN</i> sobre las tablas <i>FASES</i> y <i>CASTINGS</i> .
PRUEBA.Fecha >= FASE.F_ini.	<i>ASERCIÓN</i> sobre las tablas <i>FASES</i> y <i>PRUEBAS</i> .
CANDIDATO.F_nac <= rango(PERFIL.Edad).	<i>ASERCIÓN</i> sobre las tablas <i>CANDIDATOS</i> y <i>PERFILES</i> .
Comprobar que la fecha de nacimiento del candidato es anterior a la fecha de la prueba de <i>casting</i> a la que se presenta.	<i>ASERCIÓN</i> sobre las tablas <i>CANDIDATOS</i> y <i>PRUEBAS</i> .
PERFIL.Edad.Min <= PERFIL.Edad.Max.	<i>CHECK</i> sobre la tabla <i>PERFILES</i> .
PERFIL.Altura.Min <= PERFIL.Altura.Max.	<i>CHECK</i> sobre la tabla <i>PERFILES</i> .
Comprobar que el candidato que se presenta a una prueba de un determinado <i>casting</i> se ajusta a alguno de los perfiles requeridos por dicho <i>casting</i> .	<i>ASERCIÓN</i> sobre las tablas <i>CANDIDATOS</i> , <i>SE_PRESENTAN</i> y <i>SE_REFIEREN</i> .
Comprobación de que un candidato seleccionado para un <i>casting</i> haya superado todas las pruebas de que consta ese <i>casting</i> .	<i>ASERCIÓN</i> sobre las tablas <i>SELECCIONAN</i> y <i>SE_PRESENTAN</i> .
Exclusividad de la jerarquía <i>CASTINGS</i> .	<i>ASERCIÓN</i> sobre los subtipos de la jerarquía.
Totalidad de la jerarquía <i>CASTINGS</i> .	<i>ASERCIÓN</i> sobre los subtipos y el supertipo de la jerarquía.
Exclusividad de la jerarquía <i>CANDIDATOS</i> .	<i>ASERCIÓN</i> sobre los subtipos de la jerarquía.
Totalidad de la jerarquía <i>CANDIDATOS</i> .	<i>ASERCIÓN</i> sobre los subtipos y el supertipo de la jerarquía.

PROBLEMA 2.20: COMPAÑÍA TELEFÓNICA

Dado el esquema E/R representado en la Figura 2.70.

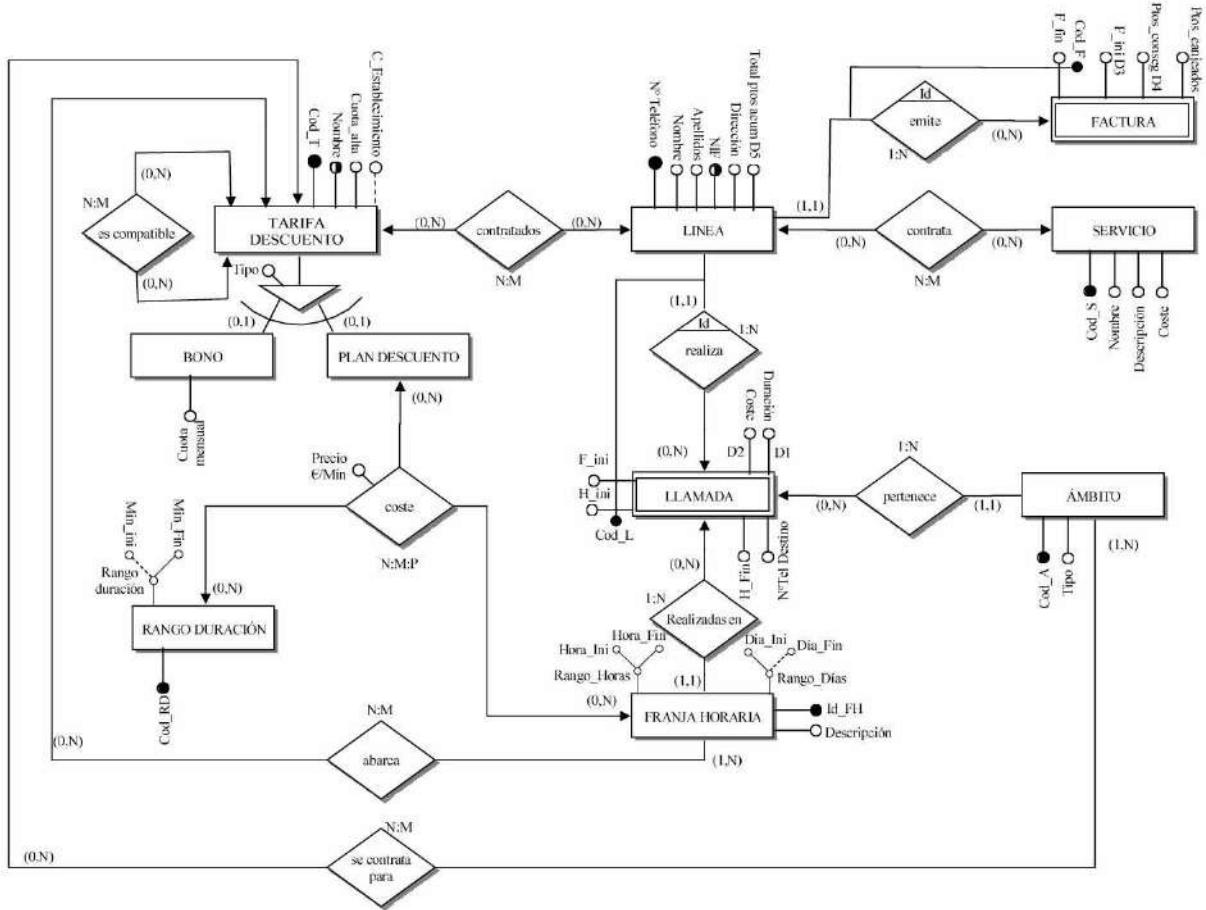


Figura 2.70. Diagrama E/R sobre compañía telefónica

Se pide:

Transformar el esquema E/R al Modelo Relacional (grafo relacional) indicando toda la semántica que no se ha podido incorporar en el grafo en forma de checks, aserciones y disparadores.

PROPUESTA DE SOLUCIÓN

La Figura 2.71 muestra el grafo relacional completo. Las opciones de borrado y modificación se han elegido sin acción y en cascada respectivamente por ser las opciones que permiten controlar los borrados al no decirse nada explícito en el enunciado.

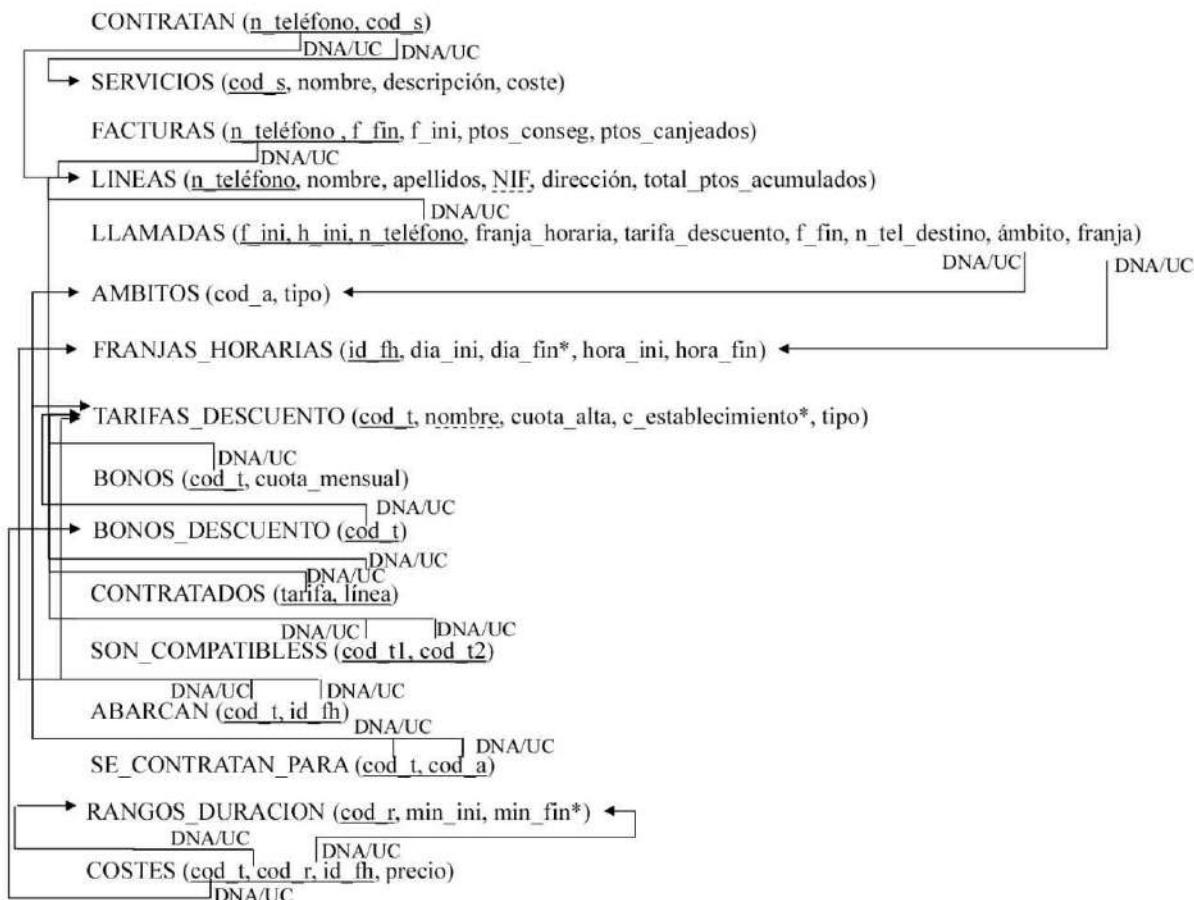


Figura 2.71. Esquema relacional correspondiente a la compañía telefónica

TRANSFORMACIÓN DE LOS SUPUESTOS SEMÁNTICOS ASOCIADOS AL DIAGRAMA E/R

En cuanto a los supuestos semánticos que no se pudieron incorporar en el diagrama E/R final, su transformación se detalla en la siguiente tabla:

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN AL ESQUEMA RELACIONAL
LÍNEA.Tipo= {básica, ADSL, RDSI}.	CHECK sobre la tabla <i>LÍNEAS</i> .
ÁMBITO.Tipo= {provincial, interprovincial, internacional}.	CHECK sobre la tabla <i>ÁMBITOS</i> .
FACTURA.Puntos_X >= 0 y LINEA.Puntos_X >= 0.	CHECK sobre las tablas <i>FACTURAS</i> y <i>LÍNEAS</i> .

Duraciones ≥ 0 .	<i>CHECK</i> sobre la tabla <i>LLAMADAS</i> .
TARIFA.DCTO.Tipo = {Bono, Plan deto}.	<i>CHECK</i> sobre la tabla <i>TARIFAS_DESCUENTO</i> .
LLAMADA.Duración = f_ini - f_fin.	<i>DISPARADOR</i> después de insertar en la tabla <i>LLAMADAS</i> , consulta en la tabla <i>LLAMADAS</i> .
LLAMADA.Coste = f(TarifaDtos, Duracion, Franja_horaria).	<i>DISPARADOR</i> antes de insertar en la tabla <i>LLAMADAS</i> , consulta las tablas <i>TARIFAS_DESCUENTO</i> , <i>RANGOS_DURACIÓN</i> y <i>FRANJA_HORARIA</i> .
FACTURA.F_Fin = FACTURA.f_ini + 2 meses.	<i>CHECK</i> sobre la tabla <i>FACTURAS</i> .
FACTURA.Puntos_conseguidos = f(llamadas (F_Ini - F_Fin)).	<i>DISPARADOR</i> antes de insertar en la tabla <i>FACTURAS</i> , consulta la tabla <i>LLAMADAS</i> .
LÍNEA.Total_Puntos_Acumulados = FACTURA.Puntos_Conseguidos - \sum FACTURA.Puntos_Canjeados.	<i>DISPARADOR</i> antes de insertar en la tabla <i>LÍNEAS</i> , consulta la tabla <i>FACTURAS</i> .
FRANJA_HORARIA.Dia_Ini <= FRANJA_HORARIA.Dia_Fin.	<i>CHECK</i> sobre la tabla <i>FRANJAS_HORARIAS</i> .
FRANJA_HORARIA.Hora_Ini <= FRANJA_HORARIA.Hora_fin.	<i>CHECK</i> sobre la tabla <i>FRANJAS_HORARIAS</i> .
RANGODURACION.Min_Ini <= RANGODURACION.Min_Fin.	<i>CHECK</i> sobre la tabla <i>RANGOS_DURACIÓN</i> .
LLAMADA.F_Ini <= LLAMADA.F_Fin.	<i>CHECK</i> sobre la tabla <i>LLAMADAS</i> .
La fecha y hora de inicio de una llamada coincide con la franja horaria con la que están relacionadas.	<i>ASERCIÓN</i> sobre las tablas <i>LLAMADAS</i> y <i>FRANJAS_HORARIAS</i> .

Una tarifa de descuento nunca es compatible consigo misma.	<i>CHECK</i> sobre la tabla <i>SON_COMPATIBLES</i> .
No se puede realizar una llamada al mismo teléfono destino que inicio.	<i>CHECK</i> sobre la tabla <i>LLAMADAS</i> .
Exclusividad de la jerarquía <i>TARIFAS DESCUENTO</i> .	<i>ASERCIÓN</i> sobre los subtipos de la jerarquía.
Totalidad de la jerarquía <i>TARIFAS DESCUENTO</i> .	<i>ASERCIÓN</i> sobre los subtipos y el supertipo de la jerarquía.

10 Al contrario que en las restricciones declarativas vistas hasta ahora en las que se rechaza la operación (inserción, borrado o modificación) si NO se cumple la condición, en el caso de los disparadores la acción especificada en el cuerpo del disparador se ejecuta siempre y cuando se cumpla la condición especificada.

11 La transformación de Dependencias en Identificación se estudiará en la sección dedicada a las extensiones del Modelo E/R.

12 Recuérdese que una Dependencia en Identificación siempre implica una Dependencia en Existencia; sin embargo, una Dependencia en Existencia no tiene por qué implicar una Dependencia en Identificación.

13 Se trataría de una o a lo sumo dos relaciones; en el primer caso se hablaría de operador unario y en el segundo de operador binario.

Capítulo 3

BASES DE DATOS DISTRIBUIDAS

3.1 INTRODUCCIÓN

Las propiedades y características de las Bases de Datos Distribuidas (BDD) requieren de metodologías de adaptación en el diseño hasta ahora enfocado al de las BD centralizadas. Como veremos, estas adaptaciones pueden abordarse de diferentes maneras y para entender el diseño adaptado a una BDD debemos primero, comprender su definición, propiedades que lo caracterizan y arquitecturas que poseen. Posteriormente analizaremos las metodologías que existen para el diseño de BDD.

La BDD surgieron como una necesidad para dar solución a los Sistemas de Información donde los servicios ofrecidos por una BD centralizada suelen ser poco eficientes, especialmente cuando el Sistema de Información es soportado en red la cual contiene un conjunto limitado de nodos ubicados en distintos sitios. Por ejemplo, una BBD de un banco permite que cada una de sus sucursales pueda procesar y almacenar sus propios datos, al mismo tiempo esta BD no pierde su estructura común o la visión global de los datos.

Una BDD es una colección de datos integrados lógicamente en una BD, pero físicamente pueden ser procesados y almacenados en varios nodos distribuidos sobre una red de ordenadores.

Son dos, pues, los rasgos que caracterizan una BDD. En primer lugar, la integración de los datos, es decir, los datos no son una colección de ficheros que están almacenados individualmente en los nodos de una red, sino que forman parte de una estructura global que los relaciona lógicamente. Y, en segundo lugar, la distribución de los datos en distintos nodos de una red que para los usuarios de la BDD aparecerán como datos homogéneos y únicos.

En resumen podemos caracterizar las BDD de la siguiente forma:

- Los datos deben estar físicamente en más de un ordenador, es decir, los datos se encuentran almacenados en distintas sedes.
- Las sedes deben estar interconectadas mediante una red. Cada sede es un emplazamiento o nodo de la red. Para realizar el diseño no se tendrá en cuenta la topología, tipo y rendimiento de la red que une estas sedes, aunque estas propiedades son relevantes para obtener buenos rendimientos y poder evitar los atascos y retrasos en el tráfico de datos por la red. Consideramos que es un aspecto muy concreto y específico

de cada red y que cada caso particular necesita su propio estudio.

- Los datos han de estar lógicamente integrados en una única estructura o esquema global común.
- Los usuarios han de tener acceso –recuperación y actualización– a los datos pertenecientes a la BDD, ya se encuentren estos en la misma sede (acceso local) o en otra (acceso remoto).
- Cada emplazamiento o nodo proporciona un entorno para la ejecución de transacciones tanto locales como globales.
- En una única operación, tanto de recuperación como de actualización, se puede acceder a datos que se encuentran en más de una sede sin que el usuario sepa la distribución de los mismos en las distintas sedes (propiedad de transparencia de red).

El Sistema de Gestión de la BDD (SGBDD) es un producto software que permite el manejo de BD y hace la distribución transparente a los usuarios, cumple con el mismo papel que un Sistema Gestor de Base de Datos (SGBD) respecto a las BD centralizadas. Se compone principalmente de cuatro componentes:

- **Procesador de datos locales.** Se encarga de la gestión local de los datos de forma parecida al software de un SGBD centralizado, por lo que además de ejecutar transacciones locales se encarga de la concurrencia y la recuperación ante fallos a nivel local.
- **Diccionario o directorio global.** Donde se guardará información acerca de dónde y cómo se almacenan los datos, el modo de acceso y otras características físicas. En resumen, contiene las especificaciones necesarias para pasar de la representación externa o esquema externo de los datos a la representación interna de los mismos.
- **Procesador de aplicaciones distribuidas.** Que es el responsable de las funciones distribuidas. Accede a la información sobre la ubicación de los datos, que se encuentra en el diccionario, y se ocupa de procesar todas las peticiones que involucran más de una sede para generar un plan de ejecución distribuido. Es el elemento diferenciador en los sistemas distribuidos, dada una operación se encargará de repartir el trabajo a los distintos procesadores locales que intervienen en dicha operación.

- **Software y red de comunicaciones.** No forma parte estrictamente del SGBDD, sino que provee al procesador de aplicaciones distribuidas de primitivas y servicios de comunicaciones para que éste lleve a cabo su labor.

Así, por ejemplo, Oracle como Sistema de Gestión de la BDD (SGBDD) dispone de un conjunto de herramientas para la implementación y mantenimiento de una BDD:

- **Acceso y modificación de datos distribuidos.** En primer lugar, es importante destacar la existencia de los DBLinks, son enlaces que permiten realizar una conexión desde una base de datos a otra. Para ocultar la localización de los objetos (transparencia de ubicación) se utilizan los sinónimos. El SQL distribuido es imprescindible para acceder o modificar datos de diferentes bases de datos en una sola sentencia SQL, en ambos casos, tanto en las consultas distribuidas como en las transacciones distribuidas el SGBDD garantiza la corrección e integridad de los datos.
- **Replicación.** Oracle dispone de diferentes mecanismos para la replicación de datos en bases de datos distribuidas:
 - *Actualización inmediata.* Con disparadores y procedimientos almacenados que utilicen las herramientas anteriormente descritas, en particular el SQL distribuido, los cambios se refrescarán inmediatamente en todas las réplicas.
 - *Vistas materializadas.* Permiten mantener réplicas que se actualicen bajo demanda o en determinados momentos del tiempo. Las replicas no están sincronizadas en todo momento, puede resultar útil para datos que son consultados esporádicamente y cuando pueda迫使 la actualización justo antes de la consulta. Las vistas materializadas permiten una mayor autonomía de las bases de datos locales ya que no requieren una conexión constante con el resto de bases de datos.
 - *Oracle Streams.* Esta utilidad de Oracle permite la sincronización de las réplicas casi a tiempo real, es decir, casi inmediatamente. El funcionamiento es el siguiente, existen una serie de objetos compartidos en el entorno de la BDD, cuando se realiza una modificación en uno de estos objetos, en primer lugar se captura el

cambio en la base de datos donde se produjo (utilizando los archivos *redo logs*), después se coloca en la cola de modificaciones a realizar de las bases de datos que poseen el objeto compartido y por último se aplica el cambio correspondiente. Este tipo de replicación es sumamente útil para crear una base de datos consolidada con datos de varias bases de datos, bien para integración de sistemas o bases de datos analíticas.

En el último apartado de este capítulo se muestran dos ejemplos de implementación en Oracle, allí podrán verse en la práctica los puntos aquí expuestos, especialmente el uso de DBLinks para conectar las bases de datos y dos técnicas de replicación, las vistas materializadas y el uso de disparadores con SQL distribuido.

3.2 CLASES Y ARQUITECTURAS DE LOS SGBDD

Dependiendo de la homogeneidad de los SGBD locales, de la distribución de los datos y de su autonomía tendremos distintos tipos de SGBDD (Figura 3.1).

Empecemos por la homogeneidad. Si todos los SGBD son iguales tenemos un SGBDD homogéneo donde se tiene un único producto, y lenguaje de consultas por lo que suelen ser sistemas muy integrados. Si los SGBD son distintos tenemos SGBDD heterogéneos donde, aunque puedan utilizar el mismo modelo de datos, se tienen distintos productos y lenguajes de consultas que requieren su integración.

La distribución determina si los datos están distribuidos físicamente sobre múltiples sitios que se comunican entre sí o si se mantienen en un único lugar, datos centralizados.

Y, por último, la autonomía, que se refiere a la distribución del control. Se pueden distinguir tres niveles de autonomía: de diseño, de comunicación y de ejecución.

Se tendrán sistemas estrechamente integrados (sistemas compuestos) si todo el acceso a los datos se realiza siempre a través del procesador de datos distribuidos con las sedes locales totalmente dependientes de éste, él gestiona todos los accesos y funciones de administración. Los sistemas semiautónomos (sistemas federados) sin embargo, dejan que cada procesador local actúe de forma autónoma e independiente, de hecho, cada uno de ellos tiene sus usuarios, administradores, transacciones y aplicaciones exclusivamente locales, además de poseer porciones específicas de la BDD. Por último, si los procesadores locales no conocen la existencia de otros SGBD ni cómo comunicarse con ellos tendremos sistemas con total autonomía (sistemas multibase de datos) donde al no haber un control global, el acceso a los datos de distintos nodos es especialmente complicado.

Podemos resumir las distintas clases de SGBDD representando cada uno de los factores estudiados anteriormente en un eje de coordenadas, de esta manera podremos observar que distribución, autonomía y heterogeneidad son aspectos ortogonales.

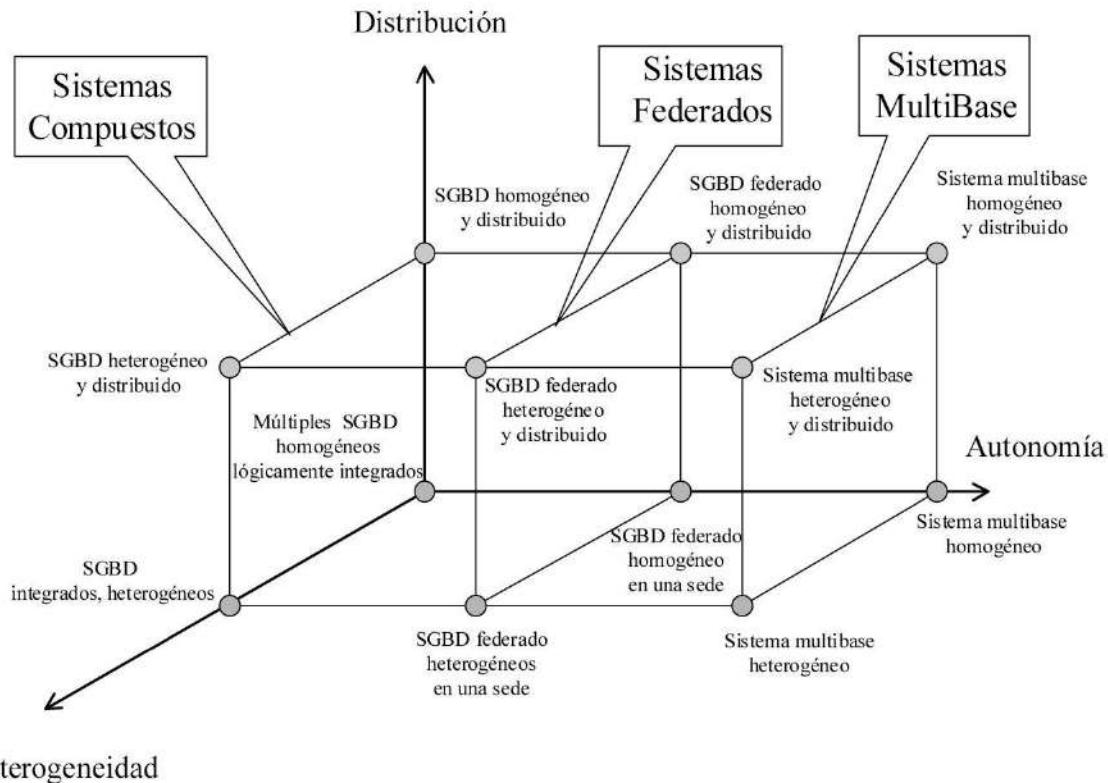


Figura 3.1. Clases de SGBDD

La arquitectura de estos sistemas puede resumirse en dos tipos: la arquitectura ANSI/X3/SPARC (*American National Standard Institute - Standards Planning And Requirements Committee*) para SGBD distribuidos y la arquitectura de los SGBD Multibase. La primera arquitectura es general para SGBD y está basada en tres niveles o esquemas: el esquema conceptual, el esquema lógico o externo y el esquema físico o interno.

La arquitectura ANSI/X3/SPARC (Figura 3.2) se corresponde con los sistemas federados, es una extensión de la dada para los sistemas centralizados donde si se omiten los esquemas externos locales tendríamos un sistema integrado. El esquema conceptual global es un subconjunto de los esquemas conceptuales locales de cada sitio que comparte datos. Los esquemas globales ofrecen una independencia lógica de los datos.

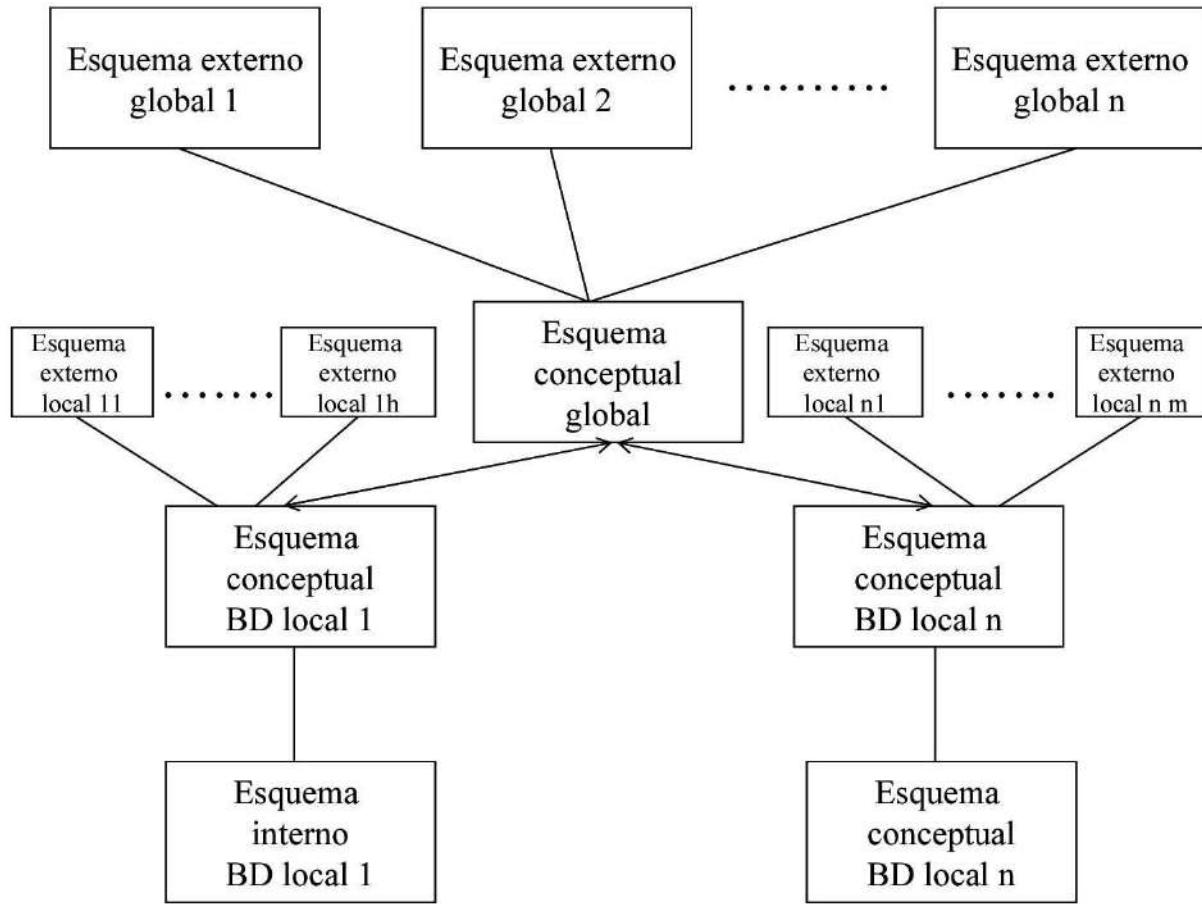


Figura 3.2. Arquitectura ANSI/X3/SPARC para SGBDD

En los sistemas multibase (Figura 3.3) no tenemos esquema conceptual global, por lo que la autonomía local es completa. Por eso, los SGBD pueden ser de diferentes tipos, y la integración de todos ellos se realiza utilizando sistemas de software que permiten a los usuarios globales y locales trabajar sin verse afectados por la distribución de los datos.

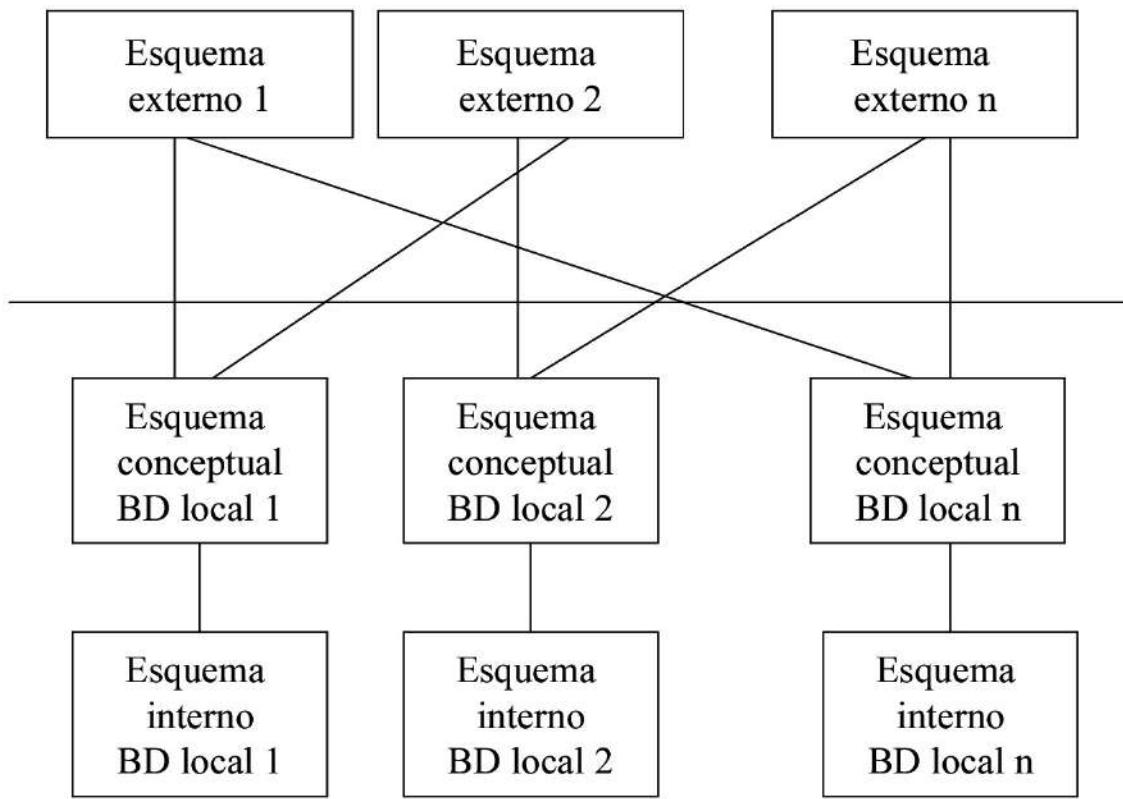


Figura 3.3. Arquitectura de los SGBD Multibase

3.3 DISEÑO DE BDD

En cualquier proceso de diseño hay dos aproximaciones básicas: la ascendente (*Top-Down*) y la descendente (*Bottom-Up*). Veamos estas aproximaciones en el caso del diseño de BDD.

La **metodología ascendente** (Figura 3.4) se utiliza, en general, cuando existen varias BD locales y queremos construir una BDD. Por lo que partimos de distintos esquemas lógicos locales (ELL) que se corresponden a BD ubicadas en diferentes nodos de una red y se integran, parte de ellos o todo, en un único esquema lógico global (ELG).

Aunque seguir esta metodología puede ser aplicable para unir distintas BD existentes, se podría pensar en el caso donde se parte de cero y se avanza en el desarrollo de la BDD siguiendo la metodología descendente.

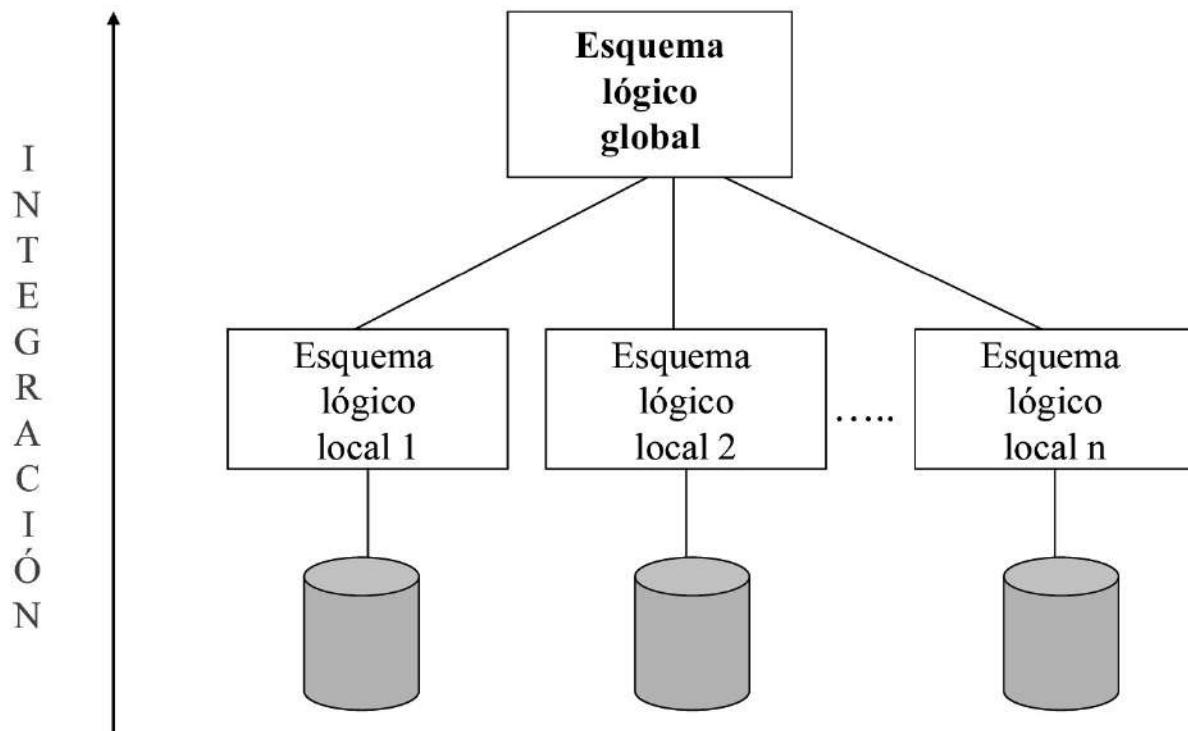


Figura 3.4. Metodología ascendente de diseño de BDD

La **metodología descendente** (Figura 3.5) parte de un esquema lógico global (ELG) y construye los distintos esquemas lógicos que se definen a partir de los esquemas de fragmentación, asignación y replicación, los cuales

determinan la distribución de los datos a través de los nodos de la red.

En la resolución de los problemas planteados en este capítulo, problemas que se corresponden con los vistos en el diseño de BDD relacionales, vamos a utilizar una **metodología descendente**, es decir, partimos de un esquema lógico global (esquema o grafo relacional en nuestro caso) de la BD que queremos distribuir y a partir de éste construimos los esquemas de fragmentación, de asignación y replicación de los datos para distribuirlos en los diferentes nodos de la red.

En general, los esquemas de fragmentación se basan en el análisis de los datos utilizados por las distintas aplicaciones que acceden a la BD para crear relaciones más pequeñas y más adaptadas a las operaciones de recuperación y actualización que las aplicaciones utilizan, es decir, tener los datos divididos según la utilización que de ellos se hace.

Sin embargo, en los esquemas de asignación y replicación se fija desde qué nodo se demandan los datos y el tipo de operación que se realiza (si es de consulta o actualización), para que estas operaciones se puedan llevar a cabo de forma local y minimizar de esta forma el tráfico por la red que las ralentiza.

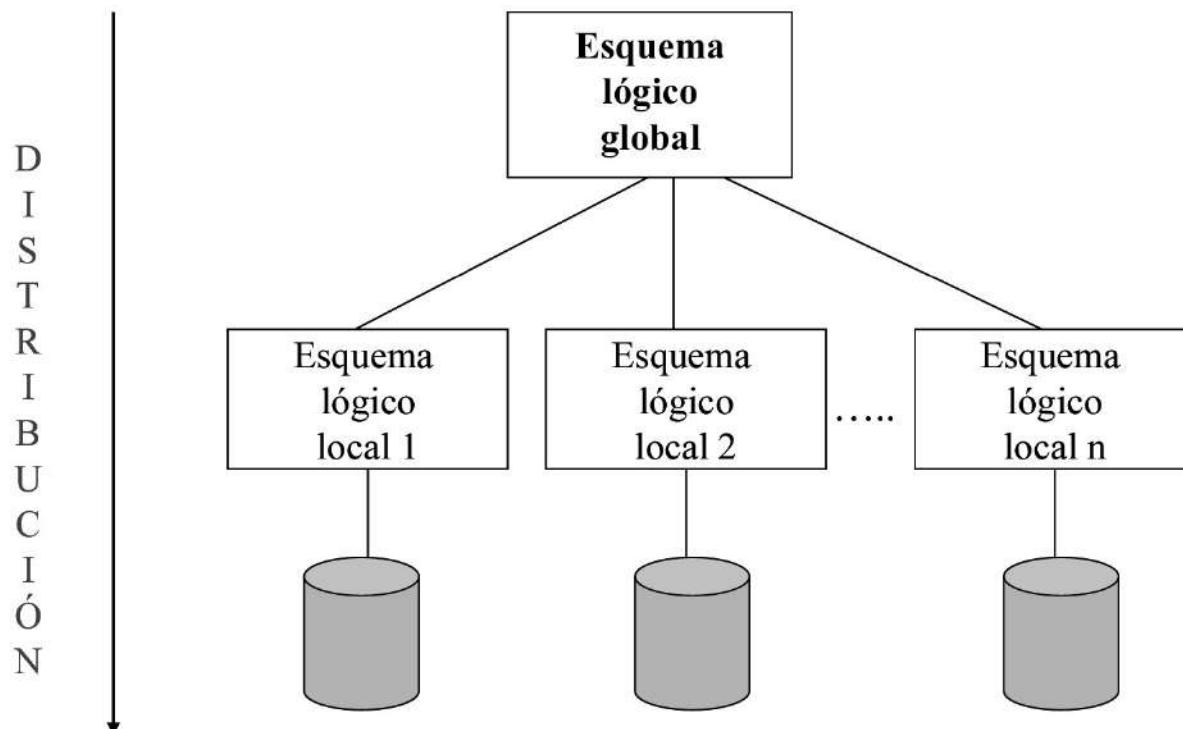


Figura 3.5. Metodología descendente de diseño de BDD

La replicación o duplicación se puede realizar cuando desde distintos nodos se requiere la misma información; si además las operaciones son de consulta no existe ningún problema en duplicar los datos. Ahora bien, si lo que hacemos es actualizarlos el SGBD debe asegurar que todas las copias son consistentes, es decir, que todas mantienen la misma información, por ejemplo, con mecanismos de propagación que actualicen todas las copias de los datos modificados, con el alto coste que esto supone. Por lo tanto, debemos llegar a una solución de compromiso y analizar ventajas y desventajas de replicar los datos.

3.3.1 Esquema de fragmentación

El esquema de fragmentación se compone de las distintas relaciones en las que hemos dividido una relación proveniente del esquema lógico global junto con la condición empleada para esta división, expresado todo ello en álgebra relacional. Las cuatro razones para la fragmentación son:

- Es útil porque las aplicaciones de BDD normalmente funcionan con vistas, por tanto, se puede emplear distintas relaciones en distintos nodos para formar la unidad distribuida.
- Permite obtener mayor eficiencia porque los datos se almacenen cerca del nodo donde se utilizan con mayor frecuencia.
- Permite aumentar el grado de concurrencia porque la fragmentación de las relaciones permite que una transacción pueda dividirse en subconsultas que operan sobre estos fragmentos.
- Proporciona más seguridad porque los datos no requeridos por un nodo local no se almacenen en él, por tanto, no están disponibles para personas no autorizadas.

Al mismo tiempo, hay que tener en cuenta que la fragmentación podría afectar el rendimiento del SGBDD especialmente cuando se utilizan distintos fragmentos ubicados en distintos nodos para construir una vista. También, el control de la semántica de los datos puede ser más complicado por la fragmentación.

Para asegurar que la BD no sufrirá cambios semánticos durante la fragmentación de los datos, se definen las siguientes tres normas que determinan la calidad de la fragmentación de una relación:

- **Completitud.** Todos los datos de una relación fragmentada han de encontrarse en, al menos, un fragmento.
- **Disyunción.** Los fragmentos deberán ser disjuntos, es decir, los datos que aparecen en un fragmento no deben aparecer en otro, excepto las claves primarias donde sí pueden aparecer más de una vez como en el caso de la fragmentación vertical.
- **Reconstrucción.** Siempre se ha de poder reconstruir la BD global a partir de los fragmentos.

Existen distintas formas de fragmentar una relación, con fragmentación horizontal, vertical o mixta.

3.3.1.1 FRAGMENTACIÓN VERTICAL

La fragmentación vertical divide la relación R en conjuntos de columnas, así cada fragmento mantiene ciertos atributos de la relación original. La fragmentación vertical se realiza mediante el operador algebraico de proyección y su notación es la siguiente:

$R_i = \Pi_{L_i}(R)$ donde $i = 1..n$ y R_i es el conjunto de fragmentos en que se divide la relación original R

L_i es la condición por la que fragmentamos, en este caso al ser una proyección se trata de un subconjunto de atributos de R . Para que se cumplan las condiciones anteriormente descritas sobre una correcta fragmentación los subconjuntos de atributos L_i han de cumplir:

- La unión de todos los L_i ($L_1 \cup L_2 \dots \cup L_n$) nos da todos los atributos de R .
- La intersección de todos los L_i ($L_1 \cap L_2 \dots \cap L_n$) nos da la clave primaria de la relación R . Esto supone que todos los L_i tienen en común la clave primaria para de esta forma poder reconstruir a partir de los fragmentos, R_i , la relación inicial, R , mediante la operación de combinación.

Veamos con un ejemplo la aplicación de este tipo de fragmentación. Supongamos que tenemos una BD centralizada de unos grandes almacenes y que

una de sus relaciones es:

EMPLEADOS: (apellidos, nombre, direccion, telefono, funcion, localización, extension)

Se quieren distribuir los datos de EMPLEADOS teniendo en cuenta que se tienen dos sedes o nodos enlazados por una red de área local y que en el NODO1 se encuentra el departamento de contabilidad y desde aquí se manejan los datos personales del empleado (apellidos, nombre, dirección, teléfono y función), mientras que en el NODO2 se encuentra centralita y en este nodo se consultan los datos correspondientes a la localización del empleado (localización y extensión).

Esquema de fragmentación:

Fragmentaremos verticalmente la tabla *EMPLEADOS* creando dos nuevas relaciones:

EMPLEADO_CONTABILIDAD =

$\Pi_{\text{apellidos, nombre, direccion, telefono, funcion}}(\text{EMPLEADOS})$

EMPLEADO_CENTRALITA = $\Pi_{\text{apellidos, nombre, localización, extensión}}(\text{EMPLEADOS})$

La combinación natural (*) de ambas relaciones proporcionará la relación original:

EMPLEADOS =

EMPLEADO_CONTABILIDAD *_{apellidos, nombre} EMPLEADO_CENTRALITA

3.3.1.2 FRAGMENTACIÓN HORIZONTAL

Este tipo de fragmentación divide una relación en subconjuntos de tuplas, cada uno de ellos con un significado lógico.

Existen dos tipos de fragmentación horizontal: primaria o derivada.

- La fragmentación horizontal primaria se define como una selección de la relación R y su notación es la siguiente: $R_i = \delta P_i(R)$ donde P_i es un predicado sobre uno o más atributos de R y es la condición empleada

para seleccionar el contenido de los fragmentos.

- La fragmentación horizontal derivada se realiza en función de predicados definidos sobre atributos de otras relaciones o fragmentos; esto se debe a que la relación R a fragmentar, depende de la relación Q , sobre cuyos atributos está definido el predicado de la fragmentación. Se define empleando el operador relacional de semicombinación (\bowtie), que actúa de la misma forma que el operador de combinación pero la relación resultante solo tiene los atributos del primer operando.

$R_i = R \bowtie Q_i$ donde Q_i corresponde al conjunto de fragmentos en los que se ha dividido la relación Q . La semicombinación se realiza por el atributo o atributos que relacionan estas dos tablas.

Para que la fragmentación sea correcta se tiene que cumplir que la unión de todos los R_i sea la relación R y que su intersección sea vacía.

Veamos una aplicación de estas fragmentaciones.

Supongamos que al ejemplo anterior le añadimos la siguiente especificación:

Cada producto, comercializado por los grandes almacenes, se encuentra en un almacén. No hay productos que se encuentren en más de un almacén. Se sabe que los almacenes están divididos en áreas (área norte, sur, este y oeste) y que además de los nodos de centralita y contabilidad hay otros cuatro nodos más, cada uno de ellos se encuentra en una de estas áreas, y en cada una de estas sedes solo se maneja la información referente a los productos almacenados en dichas áreas.

Por lo que además de la tabla *EMPLEADO* tendremos las siguientes relaciones:

PRODUCTOS (cod_prod, nombre, descripcion, almacen)

↓
ALMACEN (cod_almacen, direccion, telefono, area)

Dado que en cada almacén solo se requiere información de los productos almacenados en el área en que se encuentra, podemos realizar una fragmentación para repartir la información que se encuentra en la relación *PRODUCTOS*. Para ello haremos una fragmentación horizontal primaria en la relación *ALMACÉN*:

$$\text{ALMACEN}_i = \sigma_{\text{area} = i}(\text{ALMACEN}) \text{ donde } i = \{\text{norte, sur, este, oeste}\}$$

Cada uno de estos fragmentos contiene información de los almacenes ubicados en cada una de las áreas.

Si realizamos una fragmentación horizontal derivada en la relación *PRODUCTOS*:

$$\text{PRODUCTOS}_i = \text{PRODUCTOS} \underset{\text{almacen}=\text{Cod_almacen}}{\bowtie} \text{ALMACEN}_i$$

Conseguiremos la información de los productos almacenados en cada área.

Las relaciones originales se obtendrán mediante la unión de los fragmentos.

$$\begin{aligned} \text{ALMACEN} &= \\ \text{ALMACEN}_{\text{norte}} \cup \text{ALMACEN}_{\text{sur}} \cup \text{ALMACEN}_{\text{este}} \cup \text{ALMACEN}_{\text{oeste}} \end{aligned}$$

$$\begin{aligned} \text{PRODUCTOS} &= \\ \text{PRODUCTOS}_{\text{norte}} \cup \text{PRODUCTOS}_{\text{sur}} \cup \text{PRODUCTOS}_{\text{este}} \cup \text{PRODUCTOS}_{\text{oeste}} \end{aligned}$$

3.3.1.3 FRAGMENTACIÓN MIXTA

En algunos casos la fragmentación horizontal o vertical del esquema de la BD no es suficiente para satisfacer los requisitos de las aplicaciones de usuario por lo que puede necesitarse volver a fragmentar los fragmentos obtenidos. Esto es lo que se denomina fragmentación mixta y se clasifica de la siguiente forma:

Fragmentación VH (Figura 3.6). Una fragmentación vertical seguida de una fragmentación horizontal, sobre cada uno de los fragmentos verticales.

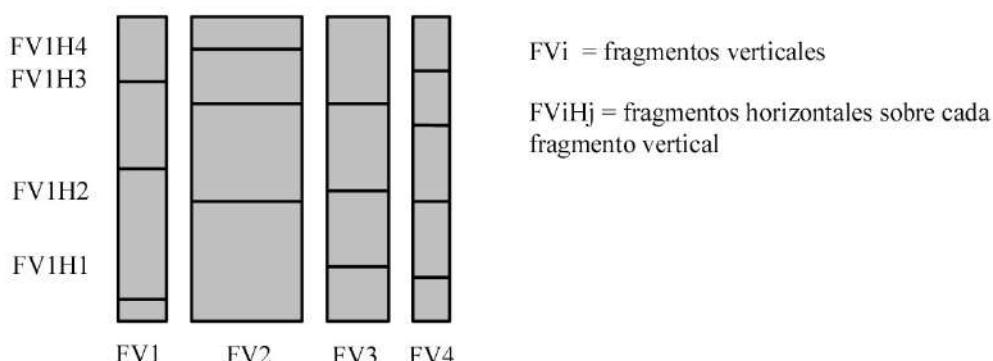


Figura 3.6. Fragmentación VH

Fragmentación HV (Figura 3.7). Una fragmentación horizontal seguida de una fragmentación vertical, sobre cada uno de los fragmentos horizontales.

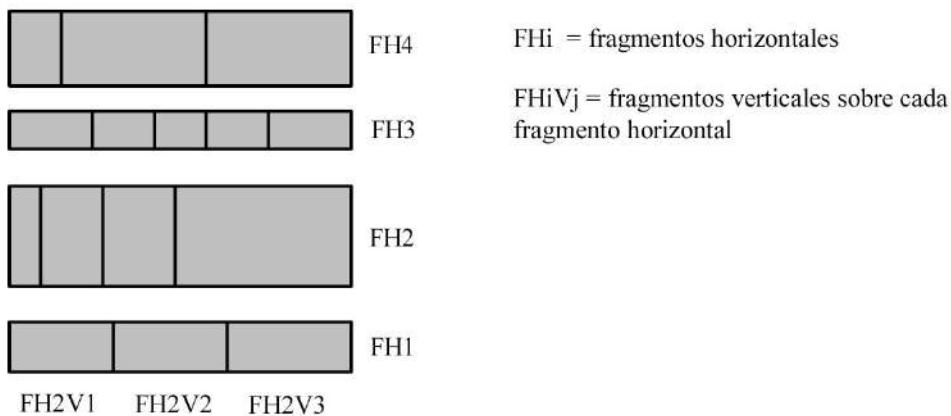


Figura 3.7. Fragmentación HV

Red de celdas (Figura 3.8). Aplicamos simultáneamente una fragmentación vertical y horizontal.

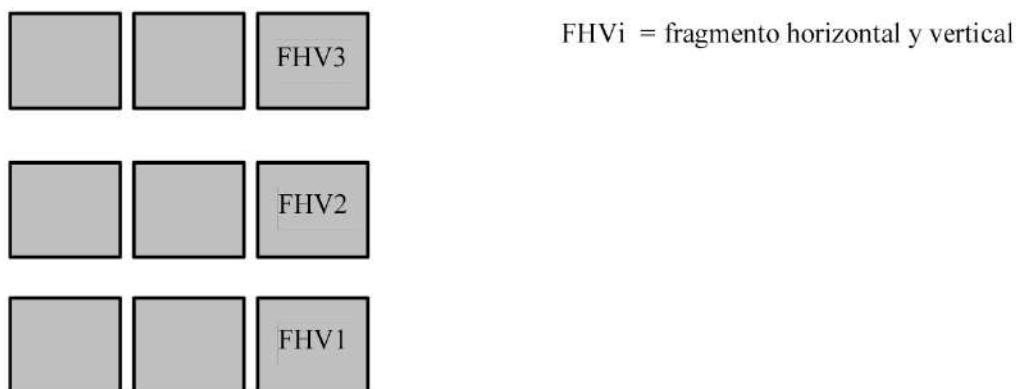


Figura 3.8. Red de celdas

3.3.2 Esquema de asignación y replicación

El esquema de asignación y replicación consiste en la realización de la correspondencia entre los fragmentos y los nodos que constituyen la red de comunicaciones de la BDD. Dicha correspondencia debe hacerse de la manera más óptima, teniendo en cuenta los dos parámetros siguientes:

- **Mínimo coste**: minimiza el coste de almacenamiento de cada fragmento en el nodo correspondiente, el coste de modificar un

fragmento que está replicado en distintos nodos y el coste de la transferencia de datos por la red.

- **Rendimiento:** minimiza los tiempos de respuesta y maximiza la capacidad de procesamiento del sistema en cada nodo.

Aunque existen algoritmos para realizar la asignación teniendo en cuenta estos parámetros, resultan complejos y no proporcionan la solución óptima sino simplemente una solución.

Por lo tanto, en la resolución de los problemas de este capítulo se ha tomado la determinación de realizar la asignación buscando siempre el procesamiento local de los datos y solo utilizar replicación cuando la operación que se realice desde el nodo que solicita los datos sea siempre de consulta.

El esquema de asignación y replicación del ejemplo anterior quedaría de la siguiente forma (Figura 3.9):

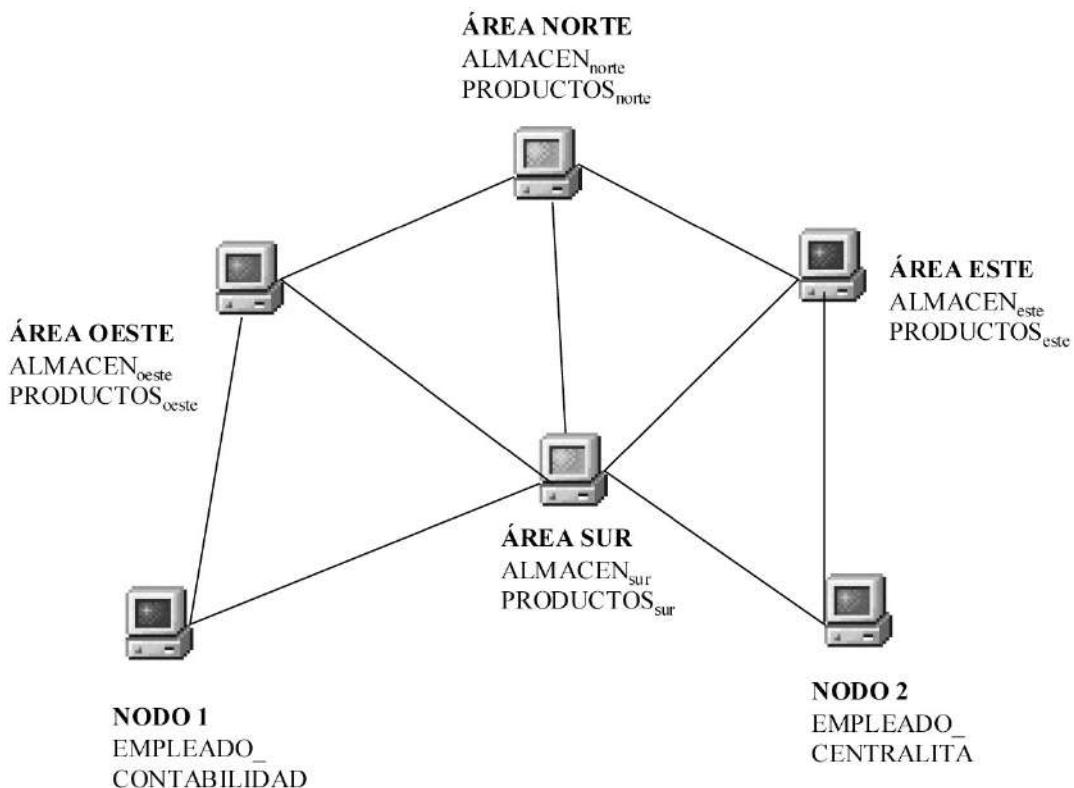


Figura 3.9. Esquema de asignación y replicación del ejemplo

PROBLEMA 3.1: BDD SOCIEDADES MÉDICAS

ENUNCIADO

Una de las sociedades médicas más importantes del país, con sede central en Madrid y otros dos grandes centros médicos en esta misma ciudad. En cada uno de ellos se atienden determinadas especialidades médicas por el personal contratado por dicha sociedad, de manera que una especialidad puede darse en un centro o en varios y todos los centros tienen al menos una especialidad.

En la actualidad tienen una base de datos relacional centralizada, a la cual acceden todos los centros, con el siguiente esquema:

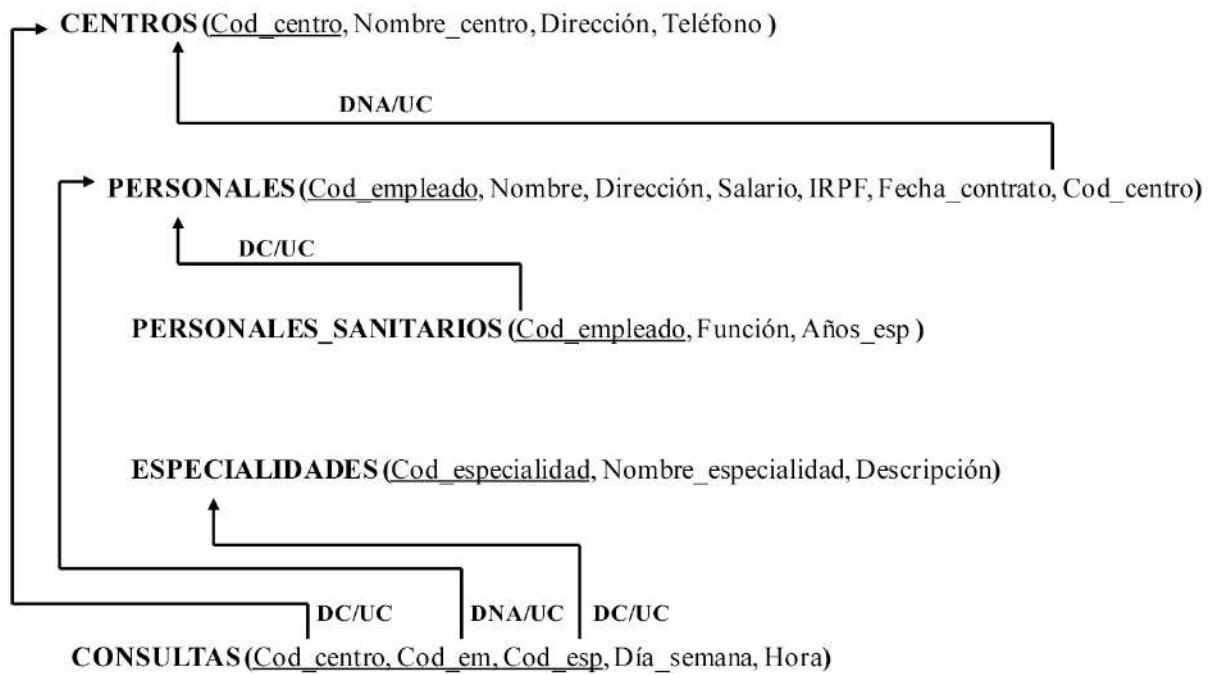


Figura 3.10. Esquema relacional correspondiente a una BD centralizada de sociedades médicas

Realice el diseño de los esquemas de fragmentación y asignación de una base de datos distribuida, sabiendo que las operaciones realizadas en cada uno de ellos son:

En la sede central (Cod_centro = 01) además del servicio sanitario se elaboran las nóminas del personal contratado por la sociedad médica y de aquí se

envían a los centros para que sean repartidas a los trabajadores.

Se desea dar una autonomía local a cada centro para gestionar el personal sanitario que trabaja en el mismo, así como el horario de la consulta y la especialidad que desarrolla.

Supongamos que la BD tiene las tuplas mostradas a continuación:

CENTROS			
<i>Cod_centro</i>	<i>Nombre_centro</i>	<i>Dirección</i>	<i>Teléfono</i>
01	Centro Salud de Madrid Central	Madrid	1111111111
02	Centro Salud de Leganés	Leganés	2222222222
03	Centro Salud de Getafe	Getafe	3333333333

PERSONALES						
Cod_empleado	Nombre	Dirección	Salario	IRPF	Fecha_contrato	Cod_centro
EM01	José Carlos	MADRID	1800	12%	04/05/1991	01
EM02	Mª Isabel	MADRID	1300	12%	12/07/1999	01
EM03	Ramón	MADRID	1500	12%	01/09/1993	02
EM04	Raúl	MADRID	1180	12%	09/11/2004	03
Em05	Juaquin	MADRID	1900	12%	07/07/1990	02

PERSONALES_SANITARIOS		
Cod_empleado	Función	Años_exp
EM01	Médico	26
EM02	Médico	18
EM03	Médico	24
EM04	Médico	10
Em05	Médico	20

ESPECIALIDADES		
Cod_especialidad	Nombre_especialidad	Descripción
ESP040	Cardiología
ESP050	Cirugía General
ESP060	Oftalmología
ESP070	Pediatria
ESP080	Urología

CONSULTAS				
Cod_centro	Cod_emp	Cod_esp	Día_semana	Hora
01	EM01	ESP040	Lunes	15:00
01	EM02	ESP050	Miércoles	17:00
02	EM03	ESP060	Lunes	15:00
03	EM04	ESP070	Lunes	15:00
02	EM05	ESP080	Lunes	15:00
01	EM03	ESP060	Martes	16:00
02	EM04	ESP070	Martes	16:00
03	EM05	ESP080	Miércoles	17:00

DISCUSIÓN DEL ENUNCIADO

Identificación de los sitios de distribución (centros) y sus respectivos roles.

- **Sede central:** con los roles: nóminas y servicio sanitario (*Cod_centro* = 01).
- **Otros centros:** con el rol: servicio sanitario (*Cod_centro* = 02, 03), independencia local de cada centro con respecto a sus datos locales.

ESQUEMA DE FRAGMENTACIÓN

Suponiendo que la información necesaria para realizar las nóminas son los atributos *Salario*, *IRPF* y *Fecha_contrato* se realizará una fragmentación vertical de la relación *PERSONALES* para separar esta información y posteriormente poder asignarla a la sede central. La relación resultante contendrá en cualquier caso la clave primaria de la relación de la que proceden.

ÁLGEBRA RELACIONAL

NOMINAS_PERSONALES: $\Pi_{\text{Cod_empleado}, \text{Salario}, \text{IRPF}, \text{Fecha_Contrato}}$ (PERSONALES)

SQL

```
SELECT Cod_empleado, Salario, IRPF, Fecha_contrato  
FROM PERSONALES;
```

La fragmentación vertical *NOMINAS_PERSONALES* significa que los siguientes datos se deben almacenar en la sede central 01.

NÓMINAS_PERSONALES			
<i>Cod_empleado</i>	<i>Salario</i>	<i>IRPF</i>	<i>Fecha_contrato</i>
EM01	1800	12%	04/05/1991
EM02	1300	12%	12/07/1999
EM03	1500	12%	01/09/1993
EM04	1180	12%	09/11/2004
Em05	1900	12%	07/07/1990

Además, nos interesará que se fragmente de nuevo la relación *PERSONALES* para tener en cada centro la información del personal *INF_PERSONALES* que está a su cargo. Para ello tendremos que aplicar una fragmentación horizontal *PERSONAL_i*:

ÁLGEBRA RELACIONAL

INF_PERSONALES: $\Pi_{\text{Cod_ empleado}, \text{Nombre}, \text{Direccion}, \text{Cod_ centro}} (\text{PERSONALES})$

PERSONALES_i: $\sigma_{\text{Cod_centros} = i} (\text{INF_PERSONALES})$

donde $i = \{01, 02, 03\}$.

SQL

```
SELECT Cod_empleado, Nombre, Direccion, Cod_centro  
      FROM PERSONALES WHERE Cod_centro = i;
```

La fragmentación horizontal *PERSONALES_i* nos indica que los siguientes datos se deben almacenar en sus correspondientes sedes.

PERSONALES_01			
<i>Cod_empleado</i>	<i>Nombre</i>	<i>Dirección</i>	<i>Cod_centro</i>
EM01	José Carlos	MADRID	01
EM02	Mª Isabel	MADRID	01

PERSONALES_02			
<i>Cod_empleado</i>	<i>Nombre</i>	<i>Dirección</i>	<i>Cod_centro</i>
EM03	Ramón	MADRID	02
Em05	Juaquin	MADRID	02

PERSONALES_03			
<i>Cod_empleado</i>	<i>Nombre</i>	<i>Dirección</i>	<i>Cod_centro</i>
EM04	Raúl	MADRID	03

Para tener la relación del personal sanitario en cada centro debemos partir de la tabla *INF_SANITARIO* y mediante los fragmentos *PERSONALES_i* realizar la fragmentación horizontal derivada.

ÁLGEBRA RELACIONAL

INF_SANITARIO: $\Pi_{\text{Cod_empleado}, \text{Funcion}, \text{Años_exp}}(\text{PERSONALES_SANITARIOS})$

PERSONALES_SANITARIOS_i: INF_SANITARIO $\bowtie_{\text{Cod_empleado}}$ PERSONALES_i

donde i = {01, 02, 03}

SQL

```
SELECT Cod_empleado, Funcion, Años_exp  
      FROM PERSONALES_SANITARIOS WHERE Cod_empleado IN  
        (SELECT Cod_empleado FROM PERSONALES  
          WHERE Cod_centro=i;
```

La fragmentación horizontal derivada *PERSONALES_SANITARIOS* nos indica que los siguientes datos se deben almacenar en sus correspondientes sedes.

PERSONAL_SANITARIO_01		
Cod_empleado	Función	Años_exp
EM01	Médico	26
EM02	Médico	18

PERSONAL_SANITARIO_02		
Cod_empleado	Función	Años_exp
EM03	Médico	24
Em05	Médico	20

PERSONAL_SANITARIO_03		
Cod_empleado	Función	Años_exp
EM04	Médico	10

Como además se quiere la información acerca de las consultas de cada centro, cuál es su horario y qué especialidad se trata se tendrá que fragmentar la relación *CONSULTAS* horizontalmente.

ÁLGEBRA RELACIONAL

INF_CONSULTAS: $\Pi_{\text{Cod_centro}, \text{Cod_emp}, \text{Cod_esp}, \text{Día_semana}, \text{Hora}} (\text{CONSULTAS})$

CONSULTAS_i: $\sigma_{\text{Cod_centros} = 'i'} (\text{INF_CONSULTAS})$

donde i = {01, 02, 03}

SQL

```
SELECT Cod_centro, Cod_emp, Cod_esp, Día_semana, Hora  
FROM CONSULTAS WHERE Cod_centro = i;
```

La fragmentación vertical *CONSULTAS_i* nos indica que los siguientes datos se deben almacenar en sus correspondientes sedes.

CONSULTAS_01				
Cod_centro	Cod_emp	Cod_esp	Día_semana	Hora
01	EM01	ESP040	Lunes	15:00
01	EM02	ESP050	Miércoles	17:00
01	EM03	ESP060	Martes	16:00

CONSULTAS_02				
Cod_centro	Cod_emp	Cod_esp	Día_semana	Hora
02	EM03	ESP060	Lunes	15:00
02	EM05	ESP080	Lunes	15:00
02	EM04	ESP070	Martes	16:00

CONSULTA_03				
Cod_centro	Cod_emp	Cod_esp	Día_semana	Hora
03	EM04	ESP070	Lunes	15:00
03	EM05	ESP080	Miércoles	17:00

Una vez aplicada esta fragmentación solo nos falta obtener las especialidades desarrolladas en cada una de las sedes de la sociedad médica, para ello fragmentaremos la tabla *ESPECIALIDADES* mediante una fragmentación horizontal derivada.

ÁLGEBRA RELACIONAL

INF_ESPECIALIDADES: $\Pi_{\text{Cod_especialidad}, \text{Nombre_especialidad}, \text{Descripcion}}$
(ESPECIALIDADES)

ESPECIALIDADES_i:

INF_ESPECIALIDADES $\bowtie_{\text{Cod_especialidad} = \text{Cod_esp}}$ (CONSULTAS_i)

donde i = {01, 02, 03}

SQL

```
SELECT Cod_especialidad, nombre_Especialidad, Descripcion  
FROM ESPECIALIDADES WHERE Cod_especialidad IN  
(SELECT Cod_esp FROM CONSULTAS WHERE Cod_centro = i;
```

ESPECIALIDADES_01		
Cod_especialidad	Nombre_especialidad	Descripción
ESP040	Cardiología
ESP050	Cirugía General
ESP060	Oftalmología

ESPECIALIDADES_02		
Cod_especialidad	Nombre_especialidad	Descripción
ESP060	Oftalmología
ESP070	Pediatria
ESP080	Urología

ESPECIALIDADES_03		
Cod_especialidad	Nombre_especialidad	Descripción
ESP070	Pediatria
ESP080	Urología

ESQUEMA DE ASIGNACIÓN

La asignación se especificará en una tabla indicando en la primera fila las sedes o nodos de la red que forman parte del sistema distribuido, y en la primera columna la tabla base de la que se ha obtenido un determinado fragmento. Debemos aclarar que como la relación *CENTRO* es poco dinámica, es decir, no sufre apenas actualizaciones, se duplicará por todos los centros.

	Centro01 	Centro02 	Centro03 
CENTROS	CENTROS	CENTROS	CENTROS
PERSONALES	NOMINAS_ PERSONALES		
INF_PERSONALES	PERSONALES_01	PERSONALES_02	PERSONALES_03
PERSONALES_ SANITARIOS	PERSONALES_ SANITARIOS_01	PERSONALES_ SANITARIOS_02	PERSONALES_ SANITARIOS_03
CONSULTAS	CONSULTAS_01	CONSULTAS_02	CONSULTAS_03
ESPECIALIDADES	ESPECIALIDADES_01	ESPECIALIDADES_02	ESPECIALIDADES_03

PROBLEMA 3.2: BDD UNIVERSIDAD CARLOS III

ENUNCIADO

La Universidad Carlos III tiene en la actualidad tres campus distribuidos en las siguientes poblaciones de la Comunidad de Madrid: Getafe, Leganés y Colmenarejo. Esta ampliación ha repercutido en los sistemas informáticos de esta Universidad llevándose a cabo el tendido de una red para unir los tres campus y ahora se baraja la posibilidad de diseñar una Base de Datos Distribuida, actualmente centralizada en Getafe, para manejar de forma autónoma información sobre las titulaciones ofertadas.

En cada uno de los campus se quiere crear un departamento para la gestión de las distintas titulaciones impartidas en ese campus, además de manejar los datos acerca de los cursos de los que consta cada una de ellas y los grupos que forman parte de estos cursos. En este departamento también se desea guardar información de los profesores (nombre, *e-mail* y número de despacho) que imparten clases en cada uno de los campus.

El departamento de gestión de nóminas y contratación de profesorado se mantendrá en Getafe. El esquema lógico global de la base de datos es el siguiente:

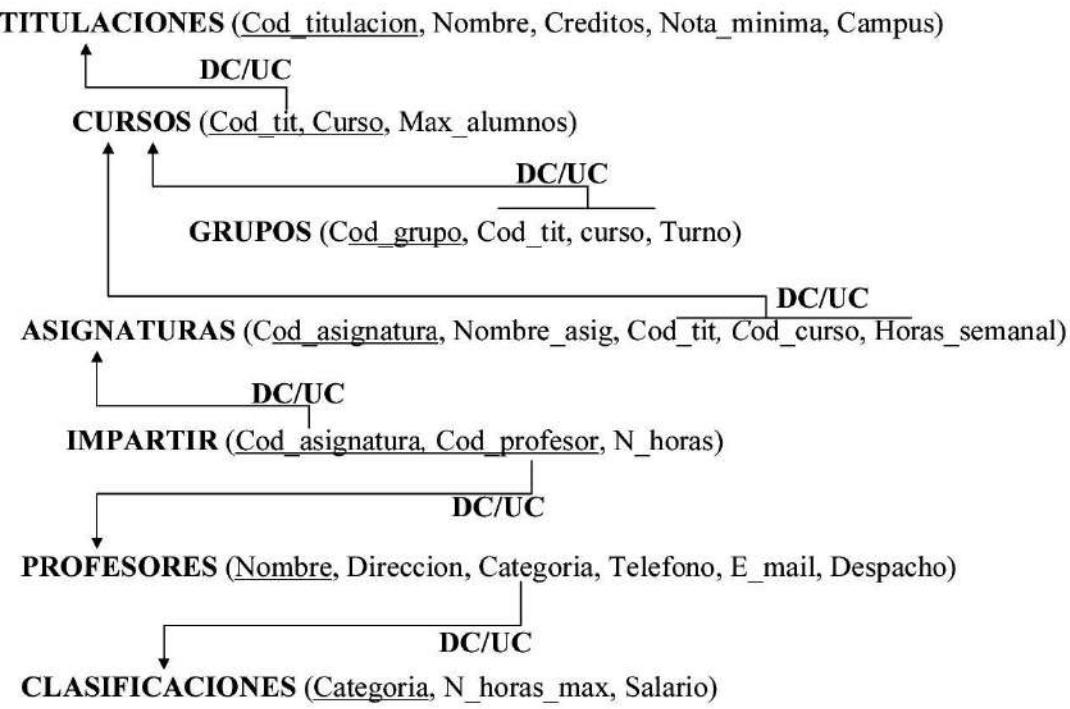


Figura 3.11. Esquema relacional correspondiente a una BD centralizada para la gestión de la Universidad Carlos III

Diseñar los esquemas de fragmentación y asignación de la base de datos distribuida, explicando las consideraciones que se han tenido en cuenta para realizar este diseño.

En relación con el diseño realizado en el apartado anterior explicar qué información se guardaría en el diccionario global de la BDD para los esquemas de fragmentación y asignación.

DISCUSIÓN DEL ENUNCIADO

Identificación de los sitios de distribución (campus) y sus respectivos roles.

- **Sede central:** con los roles: nóminas y contratación de profesores y gestión de estudios. (*Cod_campus = Getafe*)
- **Otros centros:** con el rol: gestión de estudios (*Cod_campus = Leganés, Colmenarejo*), independencia local de cada centro con respecto a sus datos locales.

ESQUEMA DE FRAGMENTACIÓN

En cada uno de los campus se quiere crear un departamento para la gestión de las distintas titulaciones impartidas en ese campus, además de manejar los datos acerca de los cursos de los que consta cada una de ellas y los grupos que forman parte de estos cursos. En este departamento también se desea guardar información de los profesores (nombre, e-mail y número de despacho) que imparten clases en cada uno de los campus.

Como se quiere que en cada campus se maneje la información de las titulaciones que allí se imparten tendremos que aplicar una fragmentación horizontal en la relación *TITULACIONES*.

$\text{TITULACIONES}_i: \sigma_{\text{Campus}='i'}(\text{TITULACIONES})$

Para tener la información completa sobre las titulaciones de cada campus necesitamos conocer qué cursos se imparten y los grupos formados para cada curso, por lo que tendremos, tanto para la relación *CURSOS* como para la relación *GRUPOS*, que realizar una fragmentación horizontal derivada.

$\text{CURSOS}_i: \text{CURSOS} \bowtie_{\text{Cod_Titulacion}} \text{TITULACIONES}_i$

$\text{GRUPOS}_i: \text{GRUPO} \bowtie_{\text{Cod_Titulacion}, \text{Curso}} \text{CURSOS}_i$

donde $i = \{\text{Getafe, Leganés, Colmenarejo}\}$

De la misma forma asignamos las asignaturas impartidas en las distintas titulaciones a los campus correspondientes realizando una fragmentación horizontal derivada con los fragmentos de *CURSOS_i*.

$\text{ASIGNATURAS}_i: \text{ASIGNATURAS} \bowtie_{\text{Cod_Titulación}, \text{Curso}} \text{CURSOS}_i$

donde $i = \{\text{Getafe, Leganés, Colmenarejo}\}$

Por último, para saber los profesores que dan clases en los distintos campus realizamos otra fragmentación derivada con los fragmentos de las asignaturas.

IMPARTIR_i: IMPARTIR $\bowtie_{\text{Cod_asignatura}}$ ASIGNATURAS_i

donde i = {Getafe, Leganés, Colmenarejo}

El departamento de gestión de nóminas y contratación de profesorado se mantendrá en Getafe.

Como solo necesitamos algunos datos acerca de los profesores que imparten asignaturas en las titulaciones de cada campus, primero realizamos una fragmentación vertical para quedarnos con la información que nos interesa.

INF_PROFESORES: $\Pi_{\text{Nombre, E-mail, Despacho}}(\text{PROFESORES})$

NOMINAS_PROFESORES: $\Pi_{\text{Nombre, Dirección, Teléfono, Categoría}}(\text{PROFESORES})$

Para luego realizar una fragmentación horizontal derivada y de esta forma obtener fragmentos con los atributos de los profesores necesarios en cada uno de los campus.

INF_PROF_i: INF_PROFESORES $\bowtie_{\text{Cod_asignatura}}$ IMPARTIR_i

donde i = {Getafe, Leganés, Colmenarejo}

ESQUEMA DE ASIGNACIÓN

A continuación presentamos una tabla con la asignación de cada uno de los fragmentos obtenidos en el apartado anterior. La relación *CLASIFICACIONES* tendrá pocas actualizaciones, por lo que se puede duplicar en todas las sedes.

	Getafe 	Colmenarejo 	Leganés 
TITULACIONES	TITULACIONES_ Getafe	TITULACIONES_ Colmenarejo	TITULACIONES_ Leganés
CURSOS	CURSOS Getafe	CURSOS Colmenarejo	CURSOS Leganés
GRUPOS	GRUPOS Getafe	GRUPOS Colmenarejo	GRUPOS Leganés
ASIGNATURAS	ASIGNATURAS_ Getafe	ASIGNATURAS_ Colmenarejo	ASIGNATURAS_ Leganés
IMPARTIR	IMPARTIR Getafe	IMPARTIR Colmenarejo	IMPARTIR Leganés
PROFESORES	NOMINAS_ PROFESORES		
CLASIFICACIONES	CLASIFICACIONES	CLASIFICACIONES	CLASIFICACIONES
INF_PROFESORES	INF_PROF Getafe	INF_PROF Colmenarejo	INF_PROF Leganés

CONTENIDO DEL DICCIONARIO

El diccionario global de la base de datos distribuida anteriormente diseñada ha de contener información sobre el esquema de fragmentación, de asignación, si existe replicación y además el esquema global de la base de datos.

El esquema global de la BD contiene la misma información que si se tratara de una BD centralizada, por lo tanto vamos a describir solo aquella información que caracteriza a una BDD relacional.

Al ser la estructura del diccionario relacional, la información sobre los fragmentos se guardará en una relación que contiene como mínimo los siguientes campos: *Nombre_Tabla_Base*, es decir, nombre de la relación a la que se aplica un determinado tipo de fragmentación. *Nombre_Fragmento*, *Sede*, *Tipo*, que tomará los valores H, HD o V dependiendo de si la fragmentación es horizontal, horizontal derivada o vertical, *Condición* que nos indicará qué condiciones hemos utilizado en los operadores del álgebra relacional para generar el fragmento y *Replicado* que tomará los valores S o N dependiendo de si el fragmento está replicado o no.

Para los demás fragmentos, la información guardada en el diccionario global sería análoga.

Si los fragmentos estuvieran duplicados se debería guardar la ubicación de las replicaciones, es decir, los nodos o sedes donde se encuentra cada uno de los fragmentos duplicados.

Nombre_Tabla_Base	Nombre_Fragmento	Sede	Tipo	Condición	Replicado
TITULACIONES	TITULACIONES_Getafe	Getafe	H	Campus= 'Getafe'	NO
CURSOS	CURSOS_Getafe	Getafe	HD	Cód_Titulación = TITULACIÓN_Getafe.Cód_Tit	NO
PROFESORES	INF_PROFESOR		V	Nombre, E-Mail,Despacho	NO
INF_PROFESORES	INF_PROF_GETAFE	Getafe	HD	IMPARTIR_GETAFE. Cód_Profesor = PROFESOR_INF. Nombre	NO
PROFESORES	NÓMINAS_PROFESORES	Getafe	V	Nombre, Dirección, Categoría, Teléfono	NO

PROBLEMA 3.3: BDD SERVICIOS INFORMÁTICOS

ENUNCIADO

La empresa de servicios informáticos “INFOSYSTEM” está ubicada en cuatro edificios cuyos códigos de identificación son ‘00’, ‘01’, ‘02’ y ‘03’, conectados entre sí por una red. En cada uno de ellos se encuentra una o varias unidades de desarrollo, por ejemplo, en el edificio cuyo código es ‘00’ se halla la unidad de Recursos Humanos.

Los empleados de la empresa y los proyectos que se realizan en ésta, están asignados a una sola unidad de desarrollo, aunque un empleado puede trabajar en un proyecto que no pertenezca a su unidad.

Los proyectos se realizan para una determinada empresa de la cual nos interesa saber el nombre y el teléfono para que dicha información esté disponible en cada uno de los edificios donde se desarrollan proyectos para estas empresas, y los demás datos se los quedará Recursos Humanos para tramitar las facturas correspondientes.

Pueden existir proyectos internos de los cuales no se tramita factura, por lo que no es necesario que Recursos Humanos posea información de estos. Estos proyectos tienen como código de empresa cliente el ‘00000’.

Suponiendo que el esquema de la base de datos relacional es el que muestra en la Figura 3.12:

- Diseñar los esquemas de fragmentación y asignación de la base de datos distribuida. Tener en cuenta que cada unidad de desarrollo gestiona las nóminas de sus empleados y necesita la información de los proyectos en los que trabaja para poder pagarles conforme a las horas que han computado en cada uno de ellos.
- Conforme al diseño realizado en el apartado anterior, modificar si se cree necesario los esquemas de fragmentación y de asignación para conseguir que la siguiente consulta se realice de manera local: “código de los proyectos que desarrollan las unidades de un determinado edificio”.



Figura 3.12. Esquema relacional correspondiente a una BD centralizada para la gestión de una empresa de servicios informáticos

DISCUSIÓN DEL ENUNCIADO

APARTADO 1

Identificación de los sitios de distribución y sus respectivos roles:

- **Sede central:** con los roles: Recursos Humanos y Servicios Informáticos (*Cod_centro = 00*).
- **Otros centros:** con el rol: Servicios Informáticos (*Cod_centro = 01, 02, 03*), independencia local de cada centro con respecto a sus datos locales.

ESQUEMA DE FRAGMENTACIÓN

La empresa de servicios informáticos INFOSYSTEM está ubicada en cuatro edificios cuyos códigos de identificación son ‘00’, ‘01’, ‘02’ y ‘03’, conectados entre sí por una red. En cada uno de ellos se encuentra una o varias unidades de desarrollo, por ejemplo, en el edificio cuyo código es ‘00’ se halla la unidad de Recursos Humanos.

Los empleados de la empresa y los proyectos que se realizan en ésta, están asignados a una sola unidad de desarrollo, aunque un empleado puede trabajar en un proyecto que no pertenezca a su unidad.

Como se observa en el esquema relacional, en un edificio existe una o varias unidades de desarrollo, pero una determinada unidad solo se encuentra en un edificio en concreto. Por tanto, nos interesaría tener en cada uno de los edificios tanto el personal como los proyectos que se llevan a cabo en él.

La relación *UNIDADES* la fragmentamos horizontalmente para saber las unidades asignadas a cada edificio.

$$\text{UNIDADES}_i: \sigma_{\text{Cod_edificio} = 'i'}(\text{UNIDADES})$$

donde $i = \{00, 01, 02, 03\}$

Para saber los empleados designados a las distintas unidades realizamos una fragmentación horizontal derivada.

$$\text{EMPLEADOS}_i: \text{EMPLEADOS} \bowtie_{\text{Cod_unidad}} \text{UNIDADES}_i$$

donde $i = \{00, 01, 02, 03\}$

De igual forma, obtenemos los fragmentos con los proyectos que se desarrollan en cada unidad.

$$\text{PROYECTOS}_i: \text{PROYECTOS} \bowtie_{\text{Cod_unidad}} \text{UNIDADES}_i$$

donde $i = \{00, 01, 02, 03\}$

Los proyectos se realizan para una determinada empresa de la cual nos interesa saber el nombre y el teléfono para que dicha información esté disponible en cada uno de los edificios donde se desarrollan proyectos para estas empresas, y los demás datos se los quedará Recursos Humanos para tramitar las facturas correspondientes.

Para que Recursos Humanos tenga la información necesaria de la empresa cliente fragmentamos verticalmente la tabla *EMPRESAS_CLIENTES*.

$\text{INF_EMPRESA: } \prod_{\text{Cod_empresa}, \text{Nombre}} (\text{EMPRESAS_CLIENTES})$

FACT_EMPRESA:

$\prod_{\text{Cod_empresa}, \text{Dirección}, \text{Apartado_correo}} (\text{EMPRESA_CLIENTES})$

Los proyectos internos que tienen como (*Cod_empresa* = ‘00000’) no se mandan a facturar, pero esta información es muy pequeña (una tupla) y podemos dejarla en la tabla *FACT_EMPRESA* y en *INF_EMPRESA*.

La información de las empresas que participan en proyectos asociados a cada una de las unidades se obtiene:

$\text{INF_EMPRESA_i: INF_EMPRESA } \bowtie_{\text{Cod_Empresa}} \text{PROYECTOS_i}$

donde $i = \{00, 01, 02, 03\}$

Tener en cuenta que cada unidad de desarrollo gestiona las nóminas de sus empleados y necesita la información de los proyectos en los que trabaja para poder pagarles conforme a las horas que han computado en cada uno de ellos.

Para que cada edificio tenga el código de los proyectos en los que trabaja el personal de las distintas unidades asignadas a éste realizamos la siguiente fragmentación horizontal derivada.

$\text{REALIZA_PROYECTOS_i: }$

$\text{REALIZA_PROYECTOS } \bowtie_{\text{Cod_Proyecto}} \text{PROYECTOS_i}$

donde $i = \{00, 01, 02, 03\}$

ESQUEMA DE ASIGNACIÓN

A continuación presentamos una tabla con la asignación de cada uno de los fragmentos obtenidos en el apartado anterior. La relación *EDIFICIOS*, que aparece en el esquema relacional, es una relación que suponemos estática, es decir, en ella no se van a realizar operaciones de actualización, solo de consulta, por lo que decidimos duplicarla en cada una de las sedes.

	Edf.00 	Edf.01 	Edf.02 	Edf.03 
UNIDAD	UNIDAD_00	UNIDAD_01	UNIDAD_02	UNIDAD_03
EMPLEADO	EMPLEADO_00	EMPLEADO_01	EMPLEADO_02	EMPLEADO_03
PROYECTO	PROYECTO_00	PROYECTO_01	PROYECTO_02	PROYECTO_03
EMPRESA_CLIENTE	FACT_EMPRESA			
INF_EMPRESA	INF_EMPRESA_00	INF_EMPRESA_01	INF_EMPRESA_02	INF_EMPRESA_03
REALIZA_PROYECTO	REALIZA_PROYECTO_00	REALIZA_PROYECTO_01	REALIZA_PROYECTO_02	REALIZA_PROYECTO_03
EDIFICIO	EDIFICIO	EDIFICIO	EDIFICIO	EDIFICIO

APARTADO 2

ESQUEMA DE FRAGMENTACIÓN

Para poder mantener el esquema de fragmentación anterior y además poder dar la información referente al código de los empleados que trabajan en los proyectos desarrollados por las unidades asignadas a los distintos edificios, realizaremos la siguiente fragmentación horizontal derivada.

\bowtie
EMP_i_j : REALIZA_PROYECTO_i _{Cod_Emp} EMPLEADO_j

donde i, j = {00,01,02,03}

Por ejemplo, la información que contiene el fragmento *EMP_00_01* sería los empleados asignados al edificio 01 que participan en los proyectos llevados a cabo por las unidades asociadas al edificio 00.

ESQUEMA DE ASIGNACIÓN

La tabla descriptora de la asignación de los fragmentos contendrá los nuevos fragmentos hallados en el esquema de fragmentación anterior y que se asignarán de la forma siguiente:

En el EDIFICIO K : EMP_K_j donde $j, K \in \{00, 01, 02, 03\}$ que serán los empleados de todos los edificios que trabajan en proyectos del edificio K . Y además, EMP_i_K donde $i = \{00, 01, 02, 03\} \setminus \{K\}$ y $K \in \{00, 01, 02, 03\}$ que serán los empleados del edificio K que trabajan en proyectos de otros edificios. En este caso el índice i no recorre todos los números de edificio para no repetir el fragmento EMP_K_K .

Tendríamos duplicación en algunos de los fragmentos, pero conseguiríamos que la consulta se realizara de manera local.

	 Edf.00	 Edf.01	 Edf.02	 Edf.03
UNIDAD	UNIDAD_00	UNIDAD_01	UNIDAD_02	UNIDAD_03
EMPLEADO	EMPLEADO_00	EMPLEADO_01	EMPLEADO_02	EMPLEADO_03
PROYECTO	PROYECTO_00	PROYECTO_01	PROYECTO_02	PROYECTO_03
EMPRESA_ CLIENTE	FACT_EMPRESA			
INF_EMPRESA	INF_ EMPRESA_00	INF_ EMPRESA_01	INF_ EMPRESA_02	INF_ EMPRESA_03
REALIZA_ PROYECTO_00	EMP_00_00 EMP_00_01 EMP_00_02 EMP_00_03	EMP_00_01	EMP_00_02	EMP_00_03
REALIZA_ PROYECTO_01	EMP_01_00 EMP_01_01 EMP_01_02 EMP_01_03	EMP_01_00 EMP_01_01 EMP_01_02 EMP_01_03	EMP_01_02	EMP_01_03
REALIZA_ PROYECTO_02	EMP_02_00	EMP_02_01	EMP_02_00 EMP_02_01 EMP_02_02 EMP_02_03	EMP_02_03
REALIZA_ PROYECTO_03	EMP_03_00	EMP_03_01	EMP_03_02	EMP_03_00 EMP_03_01 EMP_03_02 EMP_03_03
EDIFICIO	EDIFICIO	EDIFICIO	EDIFICIO	EDIFICIO

PROBLEMA 3.4: BDD OFICINAS DEL INEM

ENUNCIADO

Las oficinas del INEM de la Comunidad de Madrid se encuentran agrupadas en cinco zonas: Norte, Sur, Este, Oeste y Centro. Cada una de ellas posee un ordenador en el cual se almacenan los datos de las oficinas asignadas a la zona.

En cada una de las oficinas se imparten cursos para los desempleados. A cada oficina se le asignan, de forma exclusiva, distintos cursos, es decir, cada oficina es especialista en la docencia de unos determinados cursos, los cuales no se pueden impartir en ninguna otra oficina.

Los desempleados están adscritos en una única oficina, aunque pueden realizar cursos en cualquier oficina de la Comunidad de Madrid.

Suponiendo que todos los ordenadores de las cinco zonas están conectados entre sí y que la base de datos relacional es la que figura a continuación:



Figura 3.13. Esquema realcialonal correspondiente a una BD centralizada para la gestión de cursos del INEM

- Diseñar los esquemas de fragmentación y asignación de la base de datos distribuida teniendo en cuenta las siguientes especificaciones:

- En cada una de las zonas se precisan los siguientes datos de los desempleados: *Cod_des, nombre, apellidos, dirección, Cod_of.*
- La zona centro es la encargada de realizar las estadísticas de los desempleados de la Comunidad, para lo que necesita su fecha de nacimiento y la fecha en la que se inscribió en el paro.
- Cada zona precisa saber, con total autonomía local, los desempleados que han realizado los cursos asignados en sus oficinas.
- Además de las especificaciones anteriores, cada zona desea saber los cursos que han realizado los desempleados de la misma. Modificar, si se considera conveniente, los esquemas anteriores sabiendo que la red que une las distintas zonas es muy lenta y no es conveniente el tráfico de los datos por la misma, ya que no es una línea dedicada.

DISCUSIÓN DEL ENUNCIADO

APARTADO 1

Identificación de los sitios de distribución y sus respectivos roles:

- **Sede central:** con los roles: Estadísticas de los desempleados (*Cod_of = Centro*).
- **Otros centros:** con el rol: Cursos para los desempleados (*Cod_of = Norte, Sur, Este, Oeste*), independencia local de cada centro con respecto a sus datos locales.

ESQUEMA DE FRAGMENTACIÓN

Vamos a ir construyendo los fragmentos según las especificaciones del problema.

En cada una de las zonas se precisan los siguientes datos de los desempleados: Cod_des, Nombre, Apellidos, Dirección, Cod_of.

La zona centro es la encargada de realizar las estadísticas de los desempleados de la Comunidad, para lo que necesita su fecha de nacimiento y la fecha en la que se inscribió en el paro.

Para quedarnos con aquellos atributos que se necesitan en cada una de las sedes tendremos que realizar una fragmentación vertical en la relación *DESEMPLEADOS*.

DESEMP1: $\Pi_{\text{Cod_des}, \text{F_nacimiento}, \text{F_inscrip}}(\text{DESEMPLEADOS})$

DESEMP2: $\Pi_{\text{Cod_des}, \text{Nombre}, \text{Apellidos}, \text{Dirección}, \text{Cod_of}}(\text{DESEMPLEADOS})$

De esta fragmentación obtenemos dos fragmentos denominados, *DESEMP1* y *DESEMP2*, que cumplen que son disjuntos y cuya unión es la relación *DESEMPLEADOS*, asegurando de esta forma que la fragmentación sea correcta.

La relación *DESEMP1* servirá para que la zona centro obtenga sus estadísticas y con la otra relación, *DESEMP2*, tenemos los datos personales de los desempleados de la Comunidad de Madrid, pero no están por zonas. Para conseguir los datos personales de los desempleados de cada zona necesitaremos saber cuáles son las oficinas correspondientes a cada zona.

En un primer lugar la relación *OFICINAS* sufrirá una fragmentación horizontal cuya condición será que el atributo “zona” tome el valor de cada zona.

OF_i: $\sigma_{\text{Zona} = i}(\text{OFICINAS})$

donde $i = \{\text{Norte, Sur, Este, Oeste, Centro}\}$

Esta fragmentación nos da como resultado cinco relaciones con los mismos atributos que la relación *OFICINAS*, pero con la información de las oficinas

asignadas a cada zona.

Finalmente, para conseguir los fragmentos con la información referente a los desempleados de cada zona se realizará fragmentación horizontal derivada.

DESEMP_ZONA_i: DESEMP2 $\bowtie_{\text{Cod_of}}$ OF_i

donde i = {Norte, Sur, Este, Oeste, Centro}

Se podría pensar en otro tipo de fragmentación para conseguir los desempleados asignados a cada zona. Primero realizando una fragmentación derivada de la relación *DESEMPLEADOS* con las oficinas de cada zona.

DESEMP_i: DESEMPLAADO $\bowtie_{\text{Cod_of}}$ OF_i

donde i = {Norte, Sur, Este, Oeste, Centro}

Estos cinco fragmentos tendrían todos los atributos de la relación *DESEMPLEADOS*, por lo que para que la zona centro pudiera realizar sus estadísticas con total autonomía se debería realizar una nueva fragmentación, pero esta vez vertical en cada uno de ellos.

DESEMP1_i: $\Pi_{\text{Cod_des}, F_{\text{nacimiento}}, F_{\text{inscrip}}}$ (DESEMP_i)

donde i = {Norte, Sur, Este, Oeste, Centro}

De esta forma se aumentaría el número de fragmentos, lo que supone pérdida de eficiencia a la hora de realizar cualquier operación, tanto de actualización como de consulta, sobre la unión de fragmentos que se encuentran en la misma sede.

Cada zona precisa saber, con total autonomía local, los desempleados que han realizado los cursos asignados en sus oficinas.

Se realiza una fragmentación horizontal derivada para saber los cursos que se han impartido en cada zona.

CURSO_i: CURSOS $\bowtie_{\text{Cod_Cursos}}$ OF_i

donde i = {Norte, Sur, Este, Oeste, Centro}

Como cada zona precisa saber los desempleados que han realizado los cursos asignados a sus oficinas, se realiza una fragmentación horizontal derivada:

\bowtie
REALIZA_CURSOS_i: REALIZA_CURSOS Cod_Cursos CURSO_i

donde i = {Norte, Sur, Este, Oeste, Centro}

De esta fragmentación obtenemos el código del desempleado y el código de los cursos que se han realizado en cada zona o sede.

Una vez cubiertan las especificaciones pedidas por el enunciado tenemos que comprobar si las relaciones que nos quedan en el enunciado deben ser también fragmentadas para conseguir más autonomía local.

La relación *DESEMPEÑA* se puede fragmentar para que en cada zona se encuentren los datos de los trabajos desempeñados por los desempleados inscritos en la misma. Para ello utilizaremos una fragmentación horizontal derivada.

\bowtie
DESEMPEÑA_i: DESEMPEÑA Cod_des DESEMP_ZONA_i

donde i = {Norte, Sur, Este, Oeste, Centro}

Por último nos queda la relación *OFICIOS*, que si suponemos que es una tabla con pocas operaciones de actualización y que solo se realizan consultas, duplicaremos en todas las sedes.

ESQUEMA DE ASIGNACIÓN

Describimos a continuación, mediante una tabla, la asignación de cada fragmento o relación a las sedes correspondientes, indicando en la primera columna la relación base de la que procede cada fragmento.

	Norte 	Sur 	Centro 	Este 	Oeste 
OFICINAS	OF_Norte	OF_Sur	OF_Centro	OF_Este	OF_Oeste
DESEM- PLEADOS			DESEMP1		
DESEMP2	DESEMP_ ZONA_ Norte	DESEMP_ ZONA_ Sur	DESEMP_ ZONA_ Centro	DESEMP_ ZONA_ Este	DESEMP_ ZONA_ Oeste
CURSOS	CURSO_ Norte	CURSO_ Sur	CURSO_ Centro	CURSO_ Este	CURSO_ Oeste
DESEMPEÑA	DESEMPEÑA_ Norte	DESEMPEÑA_ Sur	DESEMPEÑA_ Centro	DESEMPEÑA_ Este	DESEMPEÑA_ Oeste
REALIZA_ CURSOS	REALIZA_ CURSO_ Norte	REALIZA_ CURSO_ Sur	REALIZA_ CURSO_ Centro	REALIZA_ CURSO_ Este	REALIZA_ CURSO_ Oeste
OFICIOS	OFICIO	OFICIO	OFICIO	OFICIO	OFICIO

APARTADO 2

Para saber en cada zona o sede los cursos que han realizado los desempleados de la misma, se pueden considerar dos soluciones:

La primera de ellas sería volver a fragmentar cada uno de los fragmentos de *REALIZA_CURSO*. Por ejemplo:

$G_{Norte_j} : \text{REALIZA_CURSO}_{Norte \underset{\text{Cod_des}}{\times} DESEMP_ZONA_j}$

donde $j = \{\text{Norte, Sur, Este, Oeste, Centro}\}$

La información que obtendríamos con cada uno de estos fragmentos sería los desempleados de la zona j que han realizado algún curso de los ofrecidos en la zona Norte.

$G_{i_j} : \text{REALIZA_CURSO}_{i \underset{\text{Cod_des}}{\times} DESEMP_ZONA_j}$

donde $i, j = \{\text{Norte, Sur, Este, Oeste, Centro}\}$

Y, por tanto, el fragmento G_{i_j} contendrá los desempleados de la zona j que han realizado algunos de los cursos ofertados en la zona i .

La asignación de estos fragmentos se realiza de la siguiente forma:

En la zona K tendríamos los fragmentos G_{K_j} y los G_{i_K} donde $i = \{\text{Norte, Sur, Este, Oeste, Centro}\} - \{K\}$ y donde $j, K = \{\text{Norte, Sur, Este, Oeste, Centro}\}$.

	Norte 	Sur 	Centro 	Este 	Oeste 
OFICINA	OF_Norte	OF_Sur	OF_Centro	OF_Este	OF_Oeste
DESEM- PLEADO			DESEMP1		
DESEMP2	DESEMP_ ZONA_Norte	DESEMP_ ZONA_Sur	DESEMP_ ZONA_Centro	DESEMP_ ZONA_Este	DESEMP_ ZONA_Oeste
CURSO	CURSO_Norte	CURSO_Sur	CURSO_Centro	CURSO_Este	CURSO_Oeste

DESEMPEÑA	DESEMPEÑA_Norte	DESEMPEÑA_Sur	DESEMPEÑA_Centro	DESEMPEÑA_Este	DESEMPEÑA_Oeste
REALIZA_CURSO_Norte	G_Norte_Norte G_Norte_Sur G_Norte_Centro G_Norte_Este G_Norte_Oeste	G_Norte_Sur	G_Norte_Centro	G_Norte_Este	G_Norte_Oeste
REALIZA_CURSO_Sur	G_Sur_Norte	G_Sur_Norte G_Sur_Sur G_Sur_Centro G_Sur_Este G_Sur_Oeste	G_Sur_Centro	G_Sur_Este	G_Sur_Oeste
REALIZA_CURSO_Centro	G_Centro_Norte	G_Centro_Sur	G_Centro_Norte G_Centro_Sur G_Centro_Centro G_Centro_Este G_Centro_Oeste	G_Centro_Este	G_Centro_Oeste
REALIZA_CURSO_Este	G_Este_Norte	G_Este_Sur	G_Este_Centro	G_Este_Norte G_Este_Sur G_Este_Centro G_Este_Este G_Este_Oeste	G_Este_Oeste
REALIZA_CURSO_Oeste	G_Oeste_Norte	G_Oeste_Sur	G_Oeste_Centro	G_Oeste_Este	G_Oeste_Norte G_Oeste_Sur G_Oeste_Centro G_Oeste_Este G_Oeste_Oeste
OFICIO	OFICIO	OFICIO	OFICIO	OFICIO	OFICIO

Como se puede observar en la tabla tendremos duplicación de algunos fragmentos pero aseguramos total autonomía en cada una de las sedes.



Otra solución posible sería la duplicación, en cada una de las zonas, de la

tabla *REALIZA_CURSOS*, ya que esta tabla no sufre actualizaciones continuas y al tener pocos atributos su almacenamiento no sería muy costoso.

PROBLEMA 3.5: BDD OFICINAS DE SEGUROS

ENUNCIADO

Segurtodo S. A. es una compañía de seguros que desea implantar una base de datos distribuida para dar independencia en la gestión a su red de oficinas. La información que manejan es la de sus empleados o agentes, clientes y pólizas de cada una de sus oficinas. Los datos de las diferentes oficinas estarán almacenados en cuatro localidades dependiendo de la ciudad en la que esté ubicada la oficina. Las localidades de almacenamiento son Barcelona, Tarragona, Lleida y Girona, que como capitales darán servicio directo a todas las oficinas de su provincia, aunque existen algunas excepciones que por proximidad no son gestionadas por su capital de provincia. Cada oficina está ubicada en una localidad y se identifica con un código, un nombre, por ejemplo “Oficina Port Olimpic”, tiene asociados un único director y varios agentes. Tanto el director como los agentes son empleados de empresa y se identifican mediante un código único, se almacena su DNI, nombre, número de teléfono, fecha de inicio en la empresa, oficina a la que pertenece, fecha de inicio en la oficina, banda salarial (intervalo de salario) y el acumulado en salario por cumplimiento de objetivos en la oficina actual. Un empleado puede cambiar de oficina pero, de momento, en esta base de datos solo nos interesa almacenar la situación actual.

Las pólizas pueden ser de varios tipos, vida, hogar, salud, automóvil y viaje. Cada póliza se identifica por un número y se requiere información sobre la fecha de contratación, las coberturas y la prima. También se quiere almacenar los pagos de las cuotas asociadas a dichas pólizas, las cuotas se identifican por el número de póliza y la fecha del recibo, y se reflejará el importe de la cuota, la forma de pago y la fecha del próximo pago.

Una póliza puede tener varios titulares y beneficiarios. Los titulares de las pólizas son clientes de la oficina y se les identifica mediante un código de cliente. Además se requiere conocer el DNI, el nombre y las pólizas de las que es titular. Todas las pólizas que contrate un cliente pertenecerán a la oficina del cliente, podría solicitar un cambio de oficina, en cuyo caso todos sus datos y sus pólizas se reasignarían. Los beneficiarios pueden no tener ninguna relación con la oficina o con la empresa por esto, de momento, tan solo se almacenan en la base de datos en forma no estructurada.

Por último, es importante destacar que los datos salariales de todos los empleados son actualizados por la oficina de Barcelona, son ellos los que asignan la banda salarial y el variable cobrado se actualiza automáticamente por la aplicación de nóminas que está centralizada en Barcelona. Estos datos pueden ser consultados por las oficinas pero bajo ningún concepto pueden modificarlos.

El diseño de la base de datos global es el siguiente:

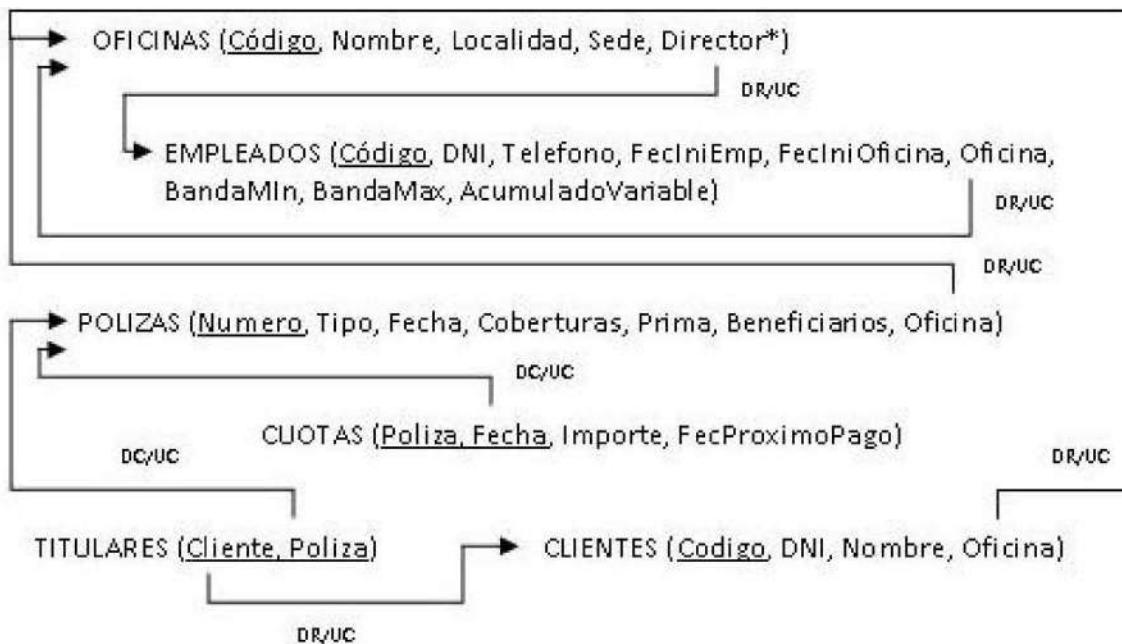


Figura 3.14. Esquema relacional correspondiente a una BD centralizada para la gestión de las oficinas de una aseguradora

Siguiendo los requisitos del enunciado y el esquema de la base de datos relacional, se pide:

- **Realizar el análisis de la distribución** indicando las sedes y los roles que intervienen y diseñar los esquemas de fragmentación, asignación y replicación de la BDD.
- **Indicar los pasos a seguir para la implementación en Oracle** incorporando los códigos que se consideren necesarios.

DISCUSIÓN DEL ENUNCIADO

APARTADO 1

Identificación de los sitios de distribución y sus respectivos roles:

Las sedes son Barcelona, Tarragona, Lleida y Girona. Los roles que se distinguen en el enunciado son dos:

- **GestiónOficinasLocal:** en cada sede se realiza la gestión de un conjunto de oficinas asociadas a esta sede. Se mantienen los datos de sus oficinas, empleados, pólizas, clientes y cuotas. Las cuatro sedes desempeñan este rol.
- **GestiónSalarialGlobal:** solamente la sede de Barcelona mantiene los datos salariales de todos los empleados de la compañía. Requerirá por tanto control absoluto de estos datos y los datos de todos los empleados de la empresa.

ESQUEMA DE FRAGMENTACIÓN

Para comenzar con la fragmentación conviene realizar un análisis de los datos que nos proporciona el enunciado, las sedes disponen de bastante autonomía en la gestión local de las oficinas que tienen asignadas, de sus clientes y pólizas asociadas a esas oficinas. Por tanto, es importante acercar a cada una de las sedes estos datos, serán necesarias fragmentaciones horizontales sin necesidad de acceso remoto ya que parece que estos datos no interesan a ninguna de las otras sedes. Sin embargo, esto no ocurre con los empleados, algunos datos son gestionados globalmente por la sede de Barcelona, por tanto parte de la información de un empleado cualquiera será propiedad de una sede y otra parte de otra, será necesaria por tanto un fragmentación vertical para separar atributos en diferentes sedes. Además, Barcelona accederá a todos los datos generales de los empleados para poder realizar las modificaciones salariales aunque no modificará ninguno, estos datos los hemos ubicado en otras sedes, será necesario por tanto un acceso remoto a los datos locales de los empleados, o más adelante se analizará si es conveniente replicar o no.

Comenzamos a recorrer el esquema global decidiendo, en función de los requisitos, si es necesaria o no la fragmentación. Es conveniente comenzar de las más sencillas a las más complejas, como normal general, dada una relación, si tiene claves ajena deberán analizar en primer lugar las relaciones a las que éstas hacen referencia. Por ejemplo, comenzaremos por la relación *OFICINAS* que es el origen de todas y una de las más sencillas, tiene una clave ajena con *EMPLEADOS* pero esta a su vez también la tiene con *OFICINAS*.

El enunciado dice que cada sede mantiene de forma exclusiva los datos de todas sus oficinas, como debemos acercar el dato a su propietario realizaremos una fragmentación horizontal primaria por sede:

$$\text{OFICINAS_T} = \sigma_{\text{Sede} = \text{'Tarragona'}} (\text{OFICINAS})$$

$$\text{OFICINAS_B} = \sigma_{\text{Sede} = \text{'Barcelona'}} (\text{OFICINAS})$$

$$\text{OFICINAS_L} = \sigma_{\text{Sede} = \text{'Lleida'}} (\text{OFICINAS})$$

$$\text{OFICINAS_G} = \sigma_{\text{Sede} = \text{'Girona'}} (\text{OFICINAS})$$

La fragmentación es correcta ya que se verifican las tres reglas, todo dato está en un fragmento puesto que el atributo *Sede* es obligatorio y solo puede tomar los valores del dominio, un dato no está en dos sedes porque el criterio de selección es único y podemos reconstruir con una unión de los cuatro fragmentos.

Continuamos con la relación *CLIENTES*, es el mismo caso que las oficinas, el objetivo es realizar una fragmentación horizontal por *Sede*, sin embargo, en la relación *CLIENTES* no tenemos un atributo *Sede*, pero sí *Oficina* y por tanto podremos identificar la sede del cliente, este es un claro ejemplo de fragmentación horizontal derivada:

$$\begin{aligned} \text{CLIENTES_T} &= \text{CLIENTES} \underset{\text{Oficina}}{\bowtie} \text{OFICINAS_T} \\ \text{CLIENTES_B} &= \text{CLIENTES} \underset{\text{Oficina}}{\bowtie} \text{OFICINAS_B} \\ \text{CLIENTES_L} &= \text{CLIENTES} \underset{\text{Oficina}}{\bowtie} \text{OFICINAS_L} \\ \text{CLIENTES_G} &= \text{CLIENTES} \underset{\text{Oficina}}{\bowtie} \text{OFICINAS_G} \end{aligned}$$

La fragmentación es correcta ya que se verifican las tres reglas, todo dato está en un fragmento puesto que el atributo *Oficina* es obligatorio y solo puede tomar los valores del fragmento correspondiente, un dato no está en dos sedes porque un cliente pertenece a una única oficina y reconstruimos con la unión de los cuatro fragmentos.

Para la relación *POLIZAS*, del mismo modo que los clientes, realizaremos una fragmentación horizontal derivada:

$$\begin{aligned} \text{POLIZAS_T} &= \text{POLIZAS} \underset{\text{Oficina}}{\bowtie} \text{OFICINAS_T} \\ \text{POLIZAS_B} &= \text{POLIZAS} \underset{\text{Oficina}}{\bowtie} \text{OFICINAS_B} \\ \text{POLIZAS_L} &= \text{POLIZAS} \underset{\text{Oficina}}{\bowtie} \text{OFICINAS_L} \\ \text{POLIZAS_G} &= \text{POLIZAS} \underset{\text{Oficina}}{\bowtie} \text{OFICINAS_G} \end{aligned}$$

La fragmentación es correcta ya que, del mismo modo que en clientes, se verifican las tres reglas.

La fragmentación de la relación *EMPLEADOS* es un poco más compleja ya

que en el análisis previo que se realizó se detectó que era necesario dividir los datos locales gestionados por cada oficina de los globales que pertenecen a Barcelona. Para esto realizaremos una fragmentación vertical:

$\text{SALARIOS_EMP} = \Pi_{\text{Código}, \text{BandaMin}, \text{BandaMax}, \text{AcumuladorVariable}}(\text{EMPLEADOS})$

$\text{DATOS_EMP} = \Pi_{\text{Código}, \text{DNI}, \text{Telefono}, \text{FiniEmp}, \text{FecIniOficina}, \text{Oficina}}(\text{EMPLEADOS})$

Por otro lado, los datos de los empleados han de estar distribuidos en cada una de las sedes, entonces será necesaria una fragmentación horizontal derivada sobre el fragmento DATOS_EMP :

$\text{DATOS_EMP_T} = \text{DATOS_EMP} \bowtie_{\text{Oficina}} \text{OFICINAS_T}$

$\text{DATOS_EMP_B} = \text{DATOS_EMP} \bowtie_{\text{Oficina}} \text{OFICINAS_B}$

$\text{DATOS_EMP_L} = \text{DATOS_EMP} \bowtie_{\text{Oficina}} \text{OFICINAS_L}$

$\text{DATOS_EMP_GI} = \text{DATOS_EMP} \bowtie_{\text{Oficina}} \text{OFICINAS_G}$

En resumen, sobre la relación EMPLEADOS se propone una fragmentación mixta (primero vertical y después horizontal), esto supone que la verificación de las tres reglas debe realizarse en dos partes, en primer lugar la fragmentación vertical y a continuación la horizontal.

Fragmentación vertical: todo dato está en un fragmento puesto que cada atributo de la relación está en al menos uno de ellos, un dato no está en dos sedes porque los atributos no están repetidos en las proyecciones, exceptuando el atributo *Código* que al tratarse de la clave primaria está permitida su duplicidad con el objeto de poder reconstruir la relación global. Por último, reconstruimos con un *join* entre los dos fragmentos utilizando los atributos de la clave primaria, en este caso *Código*.

Fragmentación horizontal: se realiza siguiendo la misma estrategia que con CLIENTES , por tanto la fragmentación horizontal es correcta.

Las CUOTAS han de distribuirse por las sedes acompañando a las pólizas a las que pertenecen, dado que ya tenemos las pólizas fragmentadas podemos realizar una fragmentación horizontal derivada:

$$\begin{aligned}
 \text{CUOTAS_T} &= \text{CUOTAS}_{\text{Póliza} = \text{Número}} \text{POLIZAS_T} \\
 &\quad \times \\
 \text{CUOTAS_B} &= \text{CUOTAS}_{\text{Póliza} = \text{Número}} \text{POLIZAS_B} \\
 &\quad \times \\
 \text{CUOTAS_L} &= \text{CUOTAS}_{\text{Póliza} = \text{Número}} \text{POLIZAS_L} \\
 &\quad \times \\
 \text{CUOTAS_G} &= \text{CUOTAS}_{\text{Póliza} = \text{Número}} \text{POLIZAS_G}
 \end{aligned}$$

De nuevo, la fragmentación es correcta ya que se verifican las tres reglas, todo dato está en un fragmento puesto que el atributo *Póliza* es obligatorio y solo puede tomar los valores del fragmento correspondiente, un dato no está en dos sedes porque una cuota pertenece a una única póliza y reconstruimos con unión de los cuatro fragmentos.

Por último, los *TITULARES* de las pólizas deben estar, del mismo modo que las cuotas, junto con sus pólizas. Es importante destacar que podemos hacer la fragmentación horizontal con *CLIENTES* o con *POLIZAS*, según el enunciado el cliente lo es de una sola oficina y por tanto todas sus pólizas pertenecerán a esa oficina, por tanto, podemos elegir cualquiera de las dos para realizar la fragmentación derivada:

$$\begin{aligned}
 \text{TITULARES_T} &= \text{TITULARES}_{\text{Póliza} = \text{Número}} \text{POLIZAS_T} \\
 &\quad \times \\
 \text{TITULARES_B} &= \text{TITULARES}_{\text{Póliza} = \text{Número}} \text{POLIZAS_B} \\
 &\quad \times \\
 \text{TITULARES_L} &= \text{TITULARES}_{\text{Póliza} = \text{Número}} \text{POLIZAS_L} \\
 &\quad \times \\
 \text{TITULARES_G} &= \text{TITULARES}_{\text{Póliza} = \text{Número}} \text{POLIZAS_G}
 \end{aligned}$$

Del mismo modo que en *CUOTAS*, la fragmentación horizontal es correcta.

ESQUEMA DE ASIGNACIÓN Y REPLICACIÓN

Describimos a continuación, mediante una tabla, la asignación de cada fragmento o relación a las sedes correspondientes, indicando en la primera columna la relación base de la que procede cada fragmento.

	Tarragona 	Barcelona 	Lleida 	Girona 
OFICINAS	OFICINAS_T	OFICINAS_B	OFICINAS_L	OFICINAS_G
CLIENTES	CLIENTES_T	CLIENTES_B	CLIENTES_L	CLIENTES_G
POLIZAS	POLIZAS_T	POLIZAS_B	POLIZAS_L	POLIZAS_G
EMPLEADOS	DATOS_EMP_T	DATOS_EMP_B SALARIOS_EMP	DATOS_EMP_L	DATOS_EMP_G
CUOTAS	CUOTAS_T	CUOTAS_B	CUOTAS_L	CUOTAS_G
TITULARES	TITULARES_T	TITULARES_B	TITULARES_L	TITULARES_G

Por último, queda representar la necesidad que tiene la oficina de Barcelona de consultar todos los datos de los empleados, independientemente de la sede a la que pertenezcan, dado que Barcelona nunca modificará los datos generales de los empleados que no sean de su sede y para dotar de mayor autonomía decidimos replicar los fragmentos locales *DATOS_EMP* de las sedes de Tarragona, Lleida y Gerona a Barcelona, de forma que podremos construir una relación completa. Modificamos la tabla anterior para indicar estas réplicas (el fragmento replicado tiene el mismo nombre que el origen pero con un subíndice R):

	Tarragona 	Barcelona 	Lleida 	Girona 
OFICINAS	OFICINAS_T	OFICINAS_B	OFICINAS_L	OFICINAS_G
CLIENTES	CLIENTES_T	CLIENTES_B	CLIENTES_L	CLIENTES_G
POLIZAS	POLIZAS_T	POLIZAS_B	POLIZAS_L	POLIZAS_G
EMPLEADOS	DATOS_EMP_T	DATOS_EMP_B SALARIOS_EMP DATOS_EMP_T_R DATOS_EMP_L_R DATOS_EMP_G_R	DATOS_EMP_L	DATOS_EMP_G
CUOTAS	CUOTAS_T	CUOTAS_B	CUOTAS_L	CUOTAS_G
TITULARES	TITULARES_T	TITULARES_B	TITULARES_L	TITULARES_G

APARTADO 2

Para la implementación será necesario realizar los siguientes pasos:

- **Creación de las bases de datos de las diferentes sedes.**
- **Creación de los DBLinks para las conexiones de unas bases de datos con otras.**
- **Creación de los modelos locales**, donde se crearán los objetos que se han definido en el esquema de fragmentación sin olvidar la semántica de la base de datos.
- **Otorgar los privilegios sobre los objetos que corresponda para el acceso a datos remotos.**
- **Implementar la replicación que se haya considerado necesaria.**
- **Creación de las bases de datos de las diferentes sedes.** Cada una de las sedes tendrá una base de datos, el entorno puede simularse en un solo nodo creando diferentes instancias, una por cada una de las sedes (puede utilizarse el Asistente de Configuración de Bases de Datos),

supongamos que cada una de las instancias se llama como el nombre de la sede, tendremos *BARNA*, *TARRA*, *LLEIDA*, *GIRONA*.

En cada una de las sedes ha de crearse un usuario que será el propietario de todos los objetos, para identificarlos fácilmente podemos llamarlos *userBarna*, *userTarra*, *userLleida*, *userGirona*.

- **Creación de los DBLinks para las conexiones de unas bases de datos con otras.** Una vez creadas las bases de datos y los usuarios locales estamos en condiciones de conectar unas bases de datos con otras, es decir utilizar los DBLinks. Es importante destacar que los DBLinks son unidireccionales, por tanto si el acceso es bidireccional (habitual en las BDD) se deberán crear dos enlaces, uno para cada sentido.

Por ejemplo, si queremos conectar Barcelona y Tarragona deberíamos crear dos DBLinks, uno en la base de datos de Barcelona:

```
CREATE PUBLIC DATABASE LINK TO_TARRA  
CONNECT TO userTarra IDENTIFIED BY <contraseña>  
USING 'TARRA';  
COMMIT;
```

donde:

- *TO_TARRA* es el nombre del enlace que se crea.
- *userTarra* y <contraseña> son los datos de la cuenta (nombre y contraseña) que se conectará a la BD remota. Para que esto sea posible el usuario debe tener cuenta con permisos de creación de sesión en la instancia *TARRA*.
- *TARRA* es el nombre del servicio (en la BD remota) al que se conecta mediante este enlace.
- Oracle utiliza el protocolo de confirmación en dos fases, es por ello que emplea la sentencia *COMMIT* para disparar ese mecanismo de confirmación y que los enlaces funcionen.

Del mismo modo nos conectamos a Tarragona y creamos el enlace a Barcelona:

```
CREATE PUBLIC DATABASE LINK TO _BARNA
CONNECT TO userBarna IDENTIFIED BY <contraseña>
USING 'BARNA';
COMMIT;
```

Para comprender mejor el funcionamiento de los enlaces, podemos crear una tabla prueba en la sede de Tarragona e intentar acceder a ella desde Barcelona:

Nos conectamos a la sede de Tarragona y creamos la tabla:

```
CREATE TABLE Prueba (Campo1 char(1) primary key);
```

Ahora nos conectamos a la sede de Barcelona y consultamos la tabla:

```
SELECT * FROM UserTarragona.Prueba@TO_TARRA;
```

Ya veremos más adelante que tan solo necesitaremos que las sedes no principales como Tarragona, Lleida y Girona accedan a Barcelona, es decir, tan solo un enlace en cada una de ellas a Barcelona para actualizar los datos generales de sus empleados en Barcelona.

Creación de los modelos locales, donde se crearán los objetos que se han definido en el esquema de fragmentación sin olvidar la semántica de la base de datos. Crearemos los modelos de datos locales, tablas, vistas, sinónimos, disparadores y procedimientos almacenados necesarios para la distribución.

Existen dos tipos de oficinas, para no duplicar demasiado vamos a implementar la oficina de Tarragona y la de Barcelona, Lleida y Girona serán equivalentes a Tarragona pero cambiando las etiquetas correspondientes.

```

CREATE TABLE OFICINAS_TARRAGONA (
   Codigo          VARCHAR2(4) NOT NULL,
   Nombre          VARCHAR2(20) NOT NULL,
   Localidad       VARCHAR2(20) NOT NULL,
   Sede            VARCHAR2(10) NOT NULL,
   Director        VARCHAR2(4) NULL,
CONSTRAINT pk_oficinas PRIMARY KEY (Codigo),
CHECK (Sede ='Tarragona')

);

CREATE TABLE CLIENTES_TARRAGONA (
   Codigo          VARCHAR2(7) NOT NULL,
   DNI             VARCHAR2(10) NOT NULL,
   Nombre          VARCHAR2(20)NOT NULL,
   Oficina         VARCHAR2(4) NOT NULL,
CONSTRAINT pk_clientes PRIMARY KEY (Codigo),
CONSTRAINT fk_clientes_oficinas FOREIGN KEY (Oficina)
REFERENCES OFICINAS_TARRAGONA
);

CREATE TABLE POLIZAS_TARRAGONA (
   Numero          VARCHAR2(7) NOT NULL,

```

```

        Tipo          VARCHAR2(10) NOT NULL,
        Fecha         DATE NOT NULL,
        Coberturas   VARCHAR2(2000) NOT NULL,
        Prima         NUMBER(10,2) NOT NULL,
        Beneficiarios VARCHAR2(2000) NOT NULL,
        Oficina       VARCHAR2(4) NOT NULL,
CONSTRAINT pk_polizas PRIMARY KEY (Numero),
CONSTRAINT fk_polizas_oficinas FOREIGN KEY (Oficina)
    REFERENCES OFICINAS_TARRAGONA,
CHECK (Tipo IN ('Vida','Hogar','Salud','Automovil','Viaje')))

);

CREATE TABLE CUOTAS_TARRAGONA (
        Numero          VARCHAR2(7) NOT NULL,
        Fecha           DATE NOT NULL,
        Importe         NUMBER(10,2) NOT NULL,
        FechaProximoPago DATE NOT NULL,
CONSTRAINT pk_cuotas PRIMARY KEY (Numero, Fecha),
CONSTRAINT fk_cuotas_polizas FOREIGN KEY (Numero)
    REFERENCES POLIZAS_TARRAGONA ON DELETE CASCADE,
CHECK (FechaProximoPago>=Fecha))

);

CREATE TABLE DATOS_EMP_TARRAGONA (
        Codigo          VARCHAR2(7) NOT NULL,
        DNI             VARCHAR2(10) NOT NULL,
        Telefono        VARCHAR2(12) NOT NULL,
        FecIniEmp      DATE NOT NULL,
        FecIniOficina   DATE NOT NULL,
        Oficina         VARCHAR2(4) NOT NULL,
CONSTRAINT pk_datos_emp PRIMARY KEY (Codigo),
CONSTRAINT fk_datos_emp_oficinas FOREIGN KEY (Oficina)
    REFERENCES OFICINAS_TARRAGONA,
CHECK (FecIniOficina >= FecIniEmp))

);

ALTER TABLE OFICINAS_TARRAGONA ADD (
CONSTRAINT fk_oficinas_empleados FOREIGN KEY (Director)
    REFERENCES DATOS_EMP_TARRAGONA
);

CREATE TABLE TITULARES_TARRAGONA (
        Cliente        VARCHAR2(7) NOT NULL,

```

```

    Poliza          VARCHAR2(7) NOT NULL,
CONSTRAINT pk_titulares PRIMARY KEY (Cliente,Poliza),
CONSTRAINT fk_titulares_clientes FOREIGN KEY (Cliente)
    REFERENCES CLIENTES_TARRAGONA,
CONSTRAINT fk_titulares_Polizas FOREIGN KEY (Poliza)
    REFERENCES POLIZAS_TARRAGONA ON DELETE CASCADE
);

```

La sede de Barcelona contiene los mismos objetos que la de Tarragona como sede local, será fácil construir su código reemplazando Tarragona por Barcelona y añadiendo los siguientes objetos propios de esta sede:

```

CREATE TABLE SALARIOS_EMP (
    Codigo          VARCHAR2(7) NOT NULL,
    BandaMin        NUMBER (10,2) NOT NULL,
    BandaMax        NUMBER (10,2) NOT NULL,
    AcumuladoVariable NUMBER (10,2) NOT NULL,
CONSTRAINT pk_salarios_emp PRIMARY KEY (Codigo),
CHECK (BandaMax >= BandaMin)
);

```

Como se puede apreciar en la definición de la tabla no hemos creado la clave ajena a *EMPLEADOS*, el motivo es que si lo realizamos referenciando al fragmento que está en Barcelona no se podrán grabar los datos salariales del resto de personal. Podríamos pensar que la mejor solución está en generar una vista completa y con un disparador simular la integridad de la clave ajena, sin embargo existe otro problema que no sería resuelto con esta solución y es que varias sedes podrían crear un empleado con el mismo código y entonces no podrían asignarle los datos salariales a ambos. La solución que proponemos es la siguiente, sustituimos en esta sede la tabla *DATOS_EMP_BARCELONA* por una tabla que llamamos *DATOS_EMP* y en ella almacenaremos no solo los datos de Barcelona que se grabarán directamente, sino también los datos replicados del resto de sedes. Con esto resolvemos los dos problemas, por un lado aseguramos la unicidad del código de empleado por la clave primaria de esta tabla y nos permite definir la clave ajena en *SALARIOS_EMP*. En esta solución deberemos controlar que los datos de otras oficinas no son modificados, se debe configurar seguridad a nivel de registro con Oracle.

Para implementar esta solución debemos añadir al código de Barcelona las siguientes sentencias:

```

ALTER TABLE OFICINAS_BARCELONA DROP
CONSTRAINT fk_oficinas_empleados;

DROP TABLE DATOS_EMP_BARCELONA;
CREATE TABLE DATOS_EMP (
    Código          VARCHAR2(7) NOT NULL,
    DNI             VARCHAR2(10) NOT NULL,
    Teléfono       VARCHAR2(12) NOT NULL,
    FecIniEmp      DATE NOT NULL,
    FecIniOficina   DATE NOT NULL,
    Oficina         VARCHAR2(4) NOT NULL,
    CONSTRAINT pk_datos_emp PRIMARY KEY (Código),
    CHECK (FecIniOficina >= FecIniEmp)
);

ALTER TABLE OFICINAS_BARCELONA ADD (
    CONSTRAINT fk_oficinas_empleados FOREIGN KEY (Director)
        REFERENCES DATOS_EMP
);

ALTER TABLE SALARIOS_EMP ADD (
    CONSTRAINT fk_salarios_datos FOREIGN KEY (Código)
        REFERENCES DATOS_EMP ON DELETE CASCADE
);

```

En este caso hemos tenido que eliminar de *DATOS_EMP* la clave ajena a *OFICINAS* por el mismo motivo, no existe una tabla en Barcelona que tenga todas las oficinas, debemos por tanto optar por la misma solución.

```

ALTER TABLE POLIZAS_BARCELONA DROP
CONSTRAINT fk_polizas_oficinas;

ALTER TABLE CLIENTES_BARCELONA DROP
CONSTRAINT fk_clientes_oficinas;

DROP TABLE OFICINAS_BARCELONA;

CREATE TABLE OFICINAS (
    Código          VARCHAR2(4) NOT NULL,
    Nombre          VARCHAR2(20) NOT NULL,
    Localidad      VARCHAR2(20) NOT NULL,
    Sede            VARCHAR2(10) NOT NULL,
    Director        VARCHAR2(4) NULL,
CONSTRAINT pk_oficinas PRIMARY KEY (Código),
CHECK (Sede IN ('Tarragona', 'Barcelona', 'Lleida', 'Girona')),

CONSTRAINT fk_oficinas_empleados FOREIGN KEY (Director)
REFERENCES DATOS_EMP

);

ALTER TABLE POLIZAS_BARCELONA ADD
CONSTRAINT fk_polizas_oficinas FOREIGN KEY (Oficina)
REFERENCES OFICINAS;

ALTER TABLE CLIENTES_BARCELONA ADD
CONSTRAINT fk_clientes_oficinas FOREIGN KEY (Oficina)
REFERENCES OFICINAS;

```

También se deberá comprobar que la modificación de datos en esta sede solo se realiza sobre los datos propios de la sede, del mismo modo se debe configurar seguridad a nivel de registro con Oracle.

Otorgar los privilegios sobre los objetos que corresponda para el acceso a datos remotos. En este apartado es importante ver qué objetos van a ser accedidos remotamente y por tanto será necesario otorgarles permisos a los usuarios. Normalmente se crean usuarios no propietarios que acceden a los datos del usuario propietario de los objetos, también se crean roles con una serie de privilegios tanto sobre objetos del esquema local como sobre objetos del

esquema remoto. Para simplificar el contenido de este apartado y por centrarnos en BDD y no en gestión de la seguridad, nosotros no vamos a crear usuarios parásitos ni roles y por tanto no será necesario dar privilegios especiales.

Implementar la replicación que se haya considerado necesaria. Por último, implementaremos la replicación necesaria con la herramienta que mejor se adapte a las necesidades concretas del caso de negocio.

De acuerdo con el diseño de la BDD, en Barcelona tendremos datos replicados de todas las oficinas, se requiere que la actualización sea inmediata ya que así controlaremos la unicidad de los códigos y la coherencia de los datos. Por tanto, vamos a optar por utilizar disparadores en cada una de las sedes que recojan los cambios en los fragmentos y actualicen las réplicas. Así en cada base de datos local, sobre los objetos que se requiere replicar incorporaremos disparadores que grabarán los datos en los sitios remotos. Nos conectamos a Tarragona y creamos los siguientes disparadores:

```

CREATE OR REPLACE TRIGGER TG_OFICINAS_TARRAGONA
BEFORE INSERT OR UPDATE OR DELETE ON OFICINAS_TARRAGONA
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO OFICINAS@TO_BARNA A VALUES
      (:New.Codigo, :New.Nombre, :New.Localidad,
       :New.Sede, :New.Director);
  END IF;
  IF UPDATING THEN
    UPDATE OFICINAS@TO_BARNA A SET
     Codigo=:New.Codigo, Nombre=:New.Nombre,
      Localidad=:New.Localidad, Sede=:New.Sede,
      Director=:New.Director
    WHERE A.Codigo = :New.Codigo;
  END IF;
  IF DELETING THEN
    DELETE OFICINAS@TO_BARNA A
    WHERE A.Codigo = :Old.Codigo;
  END IF;
END;
/

```



```

CREATE OR REPLACE TRIGGER TG_DATOS_EMP_TARRAGONA
BEFORE INSERT OR UPDATE OR DELETE ON DATOS_EMP_TARRAGONA
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO DATOS_EMP@TO_BARNA A VALUES
      (:New.Codigo, :New.DNI, :New.Telefono, :New.FecIniEmp,
       :New.FecIniOficina, :New.Oficina);
  END IF;
  IF UPDATING THEN
    UPDATE DATOS_EMP@TO_BARNA A SET
      Codigo=:New.Codigo, DNI=:New.DNI, Telefono=:New.Telefono,
      FecIniEmp=:New.FecIniEmp, FecIniOficina=:New.FecIniOficina,
      Oficina=:New.Oficina
    WHERE A.Codigo= :New.Codigo;
  END IF;
  IF DELETING THEN
    DELETE DATOS_EMP@TO_BARNA A
    WHERE A.Codigo= :Old.Codigo;
  END IF;
END;
/

```

Como último paso se debe comprobar que la base de datos funciona correctamente realizando modificaciones a los datos y verificando que las réplicas se actualizan. Por ejemplo, podemos crear una oficina en Tarragona y asegurarnos que también se ha creado en Barcelona:

```
INSERT INTO OFICINAS_TARRAGONA VALUES  
('01','Playa Larga','Tarragona', 'Tarragona',NULL);  
  
SELECT * FROM OFICINAS@TO_BARNA;
```

PROBLEMA 3.6: BDD EMPRESA DE COSMÉTICOS

ENUNCIADO

Una empresa de cosméticos dispone de varias sedes para gestionar y dar servicio a sus clientes, cada sede dispone de una plataforma de distribución donde se almacenan los productos y se distribuyen a los clientes. Las sedes son Barcelona, para lo que llaman Región Este, Bilbao para la Región Norte, Sevilla para la Región Sur y, por último, Madrid, que además de ser la representante de la Región Centro es la central de la compañía donde se realizan estudios analíticos y de mercado.

La empresa tiene diferentes productos, estos se identifican por un código de 7 dígitos, y se recoge información sobre el nombre del producto, el público al que va dirigido (hombre o mujer), la categoría (cuidado facial o corporal, maquillaje, perfumes, productos solares y capilares), la fecha de lanzamiento al mercado, si procede se grabará la fecha de retirada y el precio actual. Tanto las sedes como los productos se dan de alta de forma centralizada por un departamento de compras situado en Bilbao. Además, Madrid proporciona varios indicadores para realizar análisis de aceptación del producto, o estos son, facturación y beneficios tanto totales, es decir desde el lanzamiento, como mensuales, estos datos son calculados por el sistema de contabilidad e introducidos manualmente en el departamento financiero de la central y han de ser accesibles a todas las sedes. Madrid está autorizada a realizar modificaciones en los datos generales de los productos para cubrir al departamento de compras en las festividades de Bilbao.

Es importante destacar que no todas las sedes distribuyen todos los productos, pueden decidir cuáles tendrán más éxito entre sus clientes, además el coste de almacenamiento es grande por lo que no les interesa tener el almacén lleno de productos que nunca se solicitan. Así la base de datos deberá registrar qué productos vende cada sede, desde cuándo, si dejó de distribuirlo cuándo lo hizo y el volumen de ventas acumulado. Estos datos son muy interesantes para el resto de sedes ya que puede dar información de productos que están teniendo mucha demanda en otras zonas y que pueden ser interesantes para su región.

Los clientes se identifican por su NIF, tienen un nombre, una dirección social y están asociados a una sede. Los clientes realizan pedidos de productos a

la sede que les corresponde. En cada pedido el cliente indicará la fecha y la dirección de entrega, si ésta es distinta de la dirección del cliente, también se almacenará el importe total del pedido sin descuento, el descuento a aplicar y el importe final. Para cada producto del pedido, debe indicar las unidades y se guardará tanto el precio actual como el importe de esa línea de pedido. Los pedidos son locales, es decir las otras sedes no requieren esta información, sin embargo con sus datos se realizan los resúmenes y totales que sí son relevantes para otras sedes.

El diseño propuesto para la base de datos global se representa en el siguiente grafo relacional:

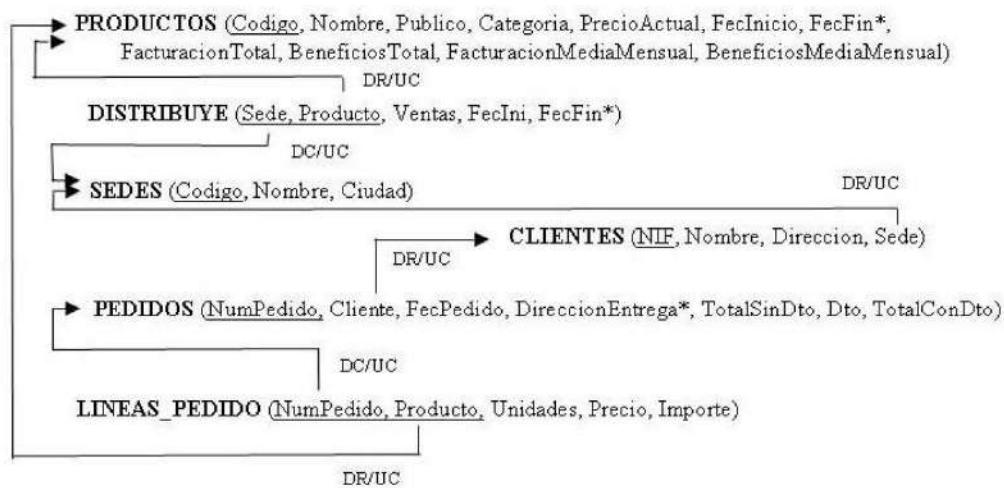


Figura 3.15. Esquema relacional correspondiente a una BD centralizada para la gestión de productos y pedidos de una empresa de cosméticos

Es importante destacar que el modelo no recoge que un cliente no puede solicitar productos que no distribuya su sede, se pueden crear disparadores que realicen este control, no es necesario que se implemente en este ejercicio.

De acuerdo a los requerimientos expuestos y siguiendo el modelo proporcionado:

- Realizar el análisis de la distribución indicando las sedes y los roles que intervienen y diseñar los esquemas de fragmentación, asignación y replicación de la BDD.
- Indicar los pasos a seguir para la implementación en Oracle incorporando los códigos que se consideren necesarios.

DISCUSIÓN DEL ENUNCIADO

APARTADO 1

Identificación de los sitios de distribución y sus respectivos roles:

Las sedes son cuatro, Región Este, Norte, Sur y Centro. Según los requerimientos del enunciado se distinguen tres roles:

- **GestiónLocal:** en cada sede o región se realiza la gestión de sus clientes, los productos que pone a su disposición y los pedidos que realizan de forma independiente.
- **GestiónFinanciera:** la sede de Madrid mantiene unos datos financieros de los productos que pone a disposición de todas las sedes.
- **GestiónDeCompras:** la sede de Bilbao de forma centralizada actualiza todos los datos generales de los productos (excepto los financieros) y por supuesto estos datos son imprescindibles para todas las sedes.

ESQUEMA DE FRAGMENTACIÓN

Comenzaremos con un análisis de la distribución de los datos en las diferentes sedes. Los clientes pertenecen exclusivamente a su sede y con ellos sus pedidos con sus detalles, estas tres relaciones serán por tanto de ámbito local y no requieren ningún acceso remoto.

Los datos referentes a las sedes se mantienen en la oficina de Bilbao y se accede desde cualquier sede.

Los productos tienen dos tipos de información, la general gestionada por Bilbao y esporádicamente por Madrid y la financiera que se mantiene exclusivamente desde Madrid. No obstante, todas las sedes deben consultar todos los productos y todos sus datos.

Cada sede decide qué productos distribuye y comparte esta información con el resto de sedes, es decir, cada sede mantiene localmente sus datos de distribución y consulta los datos de las otras sedes.

Teniendo en cuenta el análisis realizado podemos comenzar con la fragmentación, como siempre partiendo de la relación más sencilla, en este caso *SEDES*. El enunciado dice que las sedes se mantienen globalmente desde Bilbao y que todas las sedes acceden a toda la información. Por tanto, no es necesario realizar ninguna fragmentación.

Continuamos con la relación *CLIENTES*, según los requisitos cada sede o región mantiene y accede exclusivamente a sus clientes. Como en la relación tenemos un atributo *Sede* podemos realizar una fragmentación horizontal primaria por sede:

$$\text{CLIENTES_N} = \sigma_{\text{Sede} = \text{'Norte'}} (\text{CLIENTES})$$

$$\text{CLIENTES_S} = \sigma_{\text{Sede} = \text{'Sur'}} (\text{CLIENTES})$$

$$\text{CLIENTES_E} = \sigma_{\text{Sede} = \text{'Este'}} (\text{CLIENTES})$$

$$\text{CLIENTES_C} = \sigma_{\text{Sede} = \text{'Norte'}} (\text{CLIENTES})$$

La fragmentación es correcta ya que se verifican las tres reglas, todo dato está en un fragmento puesto que el atributo *Sede* es obligatorio y solo puede

tomar los valores del fragmento correspondiente, un dato no está en dos sedes porque un cliente pertenece a una única sede y reconstruimos con la unión de los cuatro fragmentos.

Para fragmentar la relación *PRODUCTOS* tenemos que prestar atención a los propietarios de los datos, los datos generales pertenecen a Bilbao (aunque Madrid acceda) que es una sede distinta que los datos financieros, por tanto, será necesaria una fragmentación vertical:

$$\text{PRODUCTOS_GEN} = \Pi_{\text{Codigo}, \text{Nombre}, \text{Publico}, \text{Categoria}, \text{PrecioActual},}$$
$$\text{FecInicio, FecFin} (\text{PRODUCTOS})$$
$$\text{PRODUCTOS_FIN} = \Pi_{\text{Codigo}, \text{FacturacionTotal}, \text{BeneficiosTotal},}$$
$$\text{FacturacionMediaMensual, BeneficiosMediaMensual} (\text{PRODUCTOS})$$

La fragmentación es correcta ya que todo dato está en un fragmento puesto que todos los atributos de la relación están en uno de los dos fragmentos, un dato no está en dos sedes porque los atributos no están repetidos en las proyecciones. A excepción del atributo *Codigo* que al tratarse de la clave primaria está permitida su duplicidad con el objeto de poder reconstruir la relación global. Por último, reconstruimos con un *join* entre los dos fragmentos utilizando los atributos de la clave primaria, en este caso *Codigo*.

La relación *DISTRIBUYE* debe fragmentarse por sede para acercar el mantenimiento a su propietario, dado que tenemos un atributo *Sede* podemos realizar una fragmentación horizontal primaria por sede:

$$\text{DISTRIBUYE_N} = \sigma_{\text{Sede} = \text{'Norte'}} (\text{DISTRIBUYE})$$
$$\text{DISTRIBUYE_S} = \sigma_{\text{Sede} = \text{'Sur'}} (\text{DISTRIBUYE})$$
$$\text{DISTRIBUYE_E} = \sigma_{\text{Sede} = \text{'Este'}} (\text{DISTRIBUYE})$$
$$\text{DISTRIBUYE_C} = \sigma_{\text{Sede} = \text{'Norte'}} (\text{DISTRIBUYE})$$

La fragmentación es correcta ya que, del mismo modo que en *CLIENTES*, se verifican las tres reglas.

Respecto a la relación *PEDIDOS*, estos están asociados a los clientes y las

LINEAS_PEDIDO a los pedidos, por tanto se fragmentarán basándose en esta relación con fragmentaciones horizontales derivadas:

$$\begin{aligned} \text{PEDIDOS_N} &= \text{PEDIDOS}_{\text{Cliente} = \text{NIF}} \text{CLIENTES_N} \\ \text{PEDIDOS_S} &= \text{PEDIDOS}_{\text{Cliente} = \text{NIF}} \text{CLIENTES_S} \\ \text{PEDIDOS_E} &= \text{PEDIDOS}_{\text{Cliente} = \text{NIF}} \text{CLIENTES_E} \\ \text{PEDIDOS_C} &= \text{PEDIDOS}_{\text{Cliente} = \text{NIF}} \text{CLIENTES_C} \\ \text{LINEAS_PEDIDO_N} &= \text{LINEAS_PEDIDO}_{\text{NumPedido}} \text{PEDIDOS_N} \\ \text{LINEAS_PEDIDO_S} &= \text{LINEAS_PEDIDO}_{\text{NumPedido}} \text{PEDIDOS_S} \\ \text{LINEAS_PEDIDO_E} &= \text{LINEAS_PEDIDO}_{\text{NumPedido}} \text{PEDIDOS_E} \\ \text{LINEAS_PEDIDO_C} &= \text{LINEAS_PEDIDO}_{\text{NumPedido}} \text{PEDIDOS_C} \end{aligned}$$

Estas fragmentaciones son correctas ya que todo dato está en un fragmento puesto que el atributo es obligatorio y al ser clave ajena solo puede tomar los valores del fragmento correspondiente, un dato no está en dos sedes porque los valores son únicos y reconstruimos con unión de los cuatro fragmentos.

EL ESQUEMA DE ASIGNACIÓN Y REPLICACIÓN

Representamos el esquema de asignación con la tabla siguiente (la primera columna indica la relación base de la que procede cada fragmento):

	Norte	Sur	Este	Centro
SEDES	SEDES			
CLIENTES	CLIENTES_N	CLIENTES_S	CLIENTES_E	CLIENTES_C
PRODUCTOS	PRODUCTOS_GEN			PRODUCTOS_FIN
DISTRIBUYE	DISTRIBUYE_N	DISTRIBUYE_S	DISTRIBUYE_E	DISTRIBUYE_C
PEDIDOS	PEDIDOS_N	PEDIDOS_S	PEDIDOS_E	PEDIDOS_C
LINEAS_PEDIDOS	LINEAS_PEDIDOS_N	LINEAS_PEDIDOS_S	LINEAS_PEDIDOS_E	LINEAS_PEDIDOS_C

Nos queda recoger las necesidades de acceso remoto a la información. Siguiendo de nuevo los requerimientos podemos afirmar que cada sede debe tener todos los datos de *SEDES* y *PRODUCTOS*, los locales para modificarlos y los remotos solo para consultarlos (Madrid también podrá modificar los datos generales de los productos), por tanto la solución más óptima es la replicación y así lo indicamos en la siguiente tabla (el fragmento replicado tiene el mismo nombre que el origen pero con un subíndice R).

	Norte 	Sur 	Este 	Centro 
SEDES	SEDES	SEDES _R	SEDES _R	SEDES _R
CLIENTES	CLIENTES_N	CLIENTES_S	CLIENTES_E	CLIENTES_C
PRODUCTOS	PRODUCTOS_GEN PRODUCTOS_FIN _R	PRODUCTOS_GEN _R PRODUCTOS_FIN _R	PRODUCTOS_GEN _R PRODUCTOS_FIN _R	PRODUCTOS_GEN _R PRODUCTOS_FIN
DISTRIBUYE	DISTRIBUYE_N	DISTRIBUYE_S	DISTRIBUYE_E	DISTRIBUYE_C
PEDIDOS	PEDIDOS_N	PEDIDOS_S	PEDIDOS_E	PEDIDOS_C
LINEAS_PEDIDOS	LINEAS_PEDIDOS_N	LINEAS_PEDIDOS_S	LINEAS_PEDIDOS_E	LINEAS_PEDIDOS_C

APARTADO 2

Para la implementación será necesario realizar los mismos cinco pasos que en el ejercicio anterior:

- Bases de datos.
- Enlaces.
- Esquemas locales.
- Privilegios sobre objetos remotos.
- Replicación.

Bases de datos. Crearemos una base de datos en cada una de las sedes (utilizando Asistente de Configuración de Bases de Datos) y un usuario en cada una de ellas que será el propietario de las tablas. Las sedes se pueden llamar *NORTE*, *CENTRO*, *ESTE* y *SUR* y los usuarios podremos llamarlos *userNorte*, *userCentro*, *userEste* y *userSur*, respectivamente. Dado que Sur y Este son equivalentes no vamos a implementar la sede Este.

Enlaces. Crearemos los enlaces que se consideren necesarios para la BDD, en este caso necesitaremos:

- **Centro:** Crearemos DBLinks a Norte, Este y Sur para actualizar las réplicas de los datos financieros de los productos desde Madrid al

resto. En la base de datos de la sede de la región Centro y con el usuario userCentro creamos los enlaces:

```
CREATE PUBLIC DATABASE LINK TO_NORTE
CONNECT TO userNorte IDENTIFIED BY <contraseña>
USING 'NORTE';
COMMIT;

CREATE PUBLIC DATABASE LINK TO_SUR
CONNECT TO userSur IDENTIFIED BY <contraseña>
USING 'SUR';
COMMIT;
```

- **Norte**: Igual que en Centro, crearemos DBLinks a Centro, Este y Sur para actualizar las réplicas de los datos generales de los productos y las sedes desde Bilbao al resto. Desde la base de datos de la región Norte y conectados como el usuario userNorte:

```
CREATE PUBLIC DATABASE LINK TO_CENTRO
CONNECT TO userCentro IDENTIFIED BY <contraseña>
USING 'CENTRO';
COMMIT;

CREATE PUBLIC DATABASE LINK TO_SUR
CONNECT TO userSur IDENTIFIED BY <contraseña>
USING 'SUR';
COMMIT;
```

- **Sur y Este**: Necesitarán acceso a las *SEDES* que están en *NORTE*, crearemos los dos *links* ejecutando la siguiente sentencia conectados a distintas bases de datos.

```
CREATE PUBLIC DATABASE LINK TO_NORTE
CONNECT TO userNorte IDENTIFIED BY <contraseña>
USING 'NORTE';
COMMIT;
```

Esquemas locales. Vamos a crear los tres tipos de sedes diferentes que existen. Una sede totalmente local, por ejemplo la Región Sur y las otras dos sedes que actúan como central de parte de los datos de la BDD. Antes de comenzar si observamos el esquema de asignación y replicación podemos ver que todas las sedes tienen los mismos fragmentos, la diferencia es en algunos casos los registros que contienen, ya que son solo los propios de la sede o si se

trata de una réplica. Por esto, para reducir el mantenimiento de los esquemas locales hemos decidido crear un único código local que lanzaremos en todas las sedes y un complemento a dicho código que incluirá las particularidades de la sede. Comenzamos por el código compartido. Ya veremos más adelante que la tabla *SEDES* no estará físicamente en los esquemas que la necesitan replicada, por esto, no la incluiremos en el esquema general:

```
CREATE TABLE CLIENTES (
    NIF          VARCHAR2(10) NOT NULL,
    Nombre       VARCHAR2(20) NOT NULL,
    Direccion    VARCHAR2(20) NOT NULL,
    Sede         VARCHAR2(4) NOT NULL,
    CONSTRAINT pk_clientes PRIMARY KEY (NIF)
);

CREATE TABLE PRODUCTOS_GEN (
    Codigo       VARCHAR2(7) NOT NULL,
    Nombre       VARCHAR2(20) NOT NULL,
    Publico      VARCHAR2(6) NOT NULL,
    Categoria    VARCHAR2(15) NOT NULL,
```

```

        PrecioActual      NUMBER(10,2) NOT NULL,
        FecInicio         DATE NOT NULL,
        FecFin            DATE,
CONSTRAINT pk_productos_gen PRIMARY KEY (Codigo),
CHECK (Publico IN ('Hombre', 'Mujer')),
CHECK (Categoria IN ('Facial', 'Corporal', 'Solares',
'Capilares', 'Maquillaje', 'Perfumes')),
CHECK (FecInicio< FecFin)
);

CREATE TABLE PRODUCTOS_FIN (
        Codigo          VARCHAR2(7) NOT NULL,
        FacturacionTotal NUMBER(10,2) NOT NULL,
        BeneficiosTotal NUMBER(10,2) NOT NULL,
        FacturacionMediaMensual NUMBER(10,2) NOT NULL,
        BeneficiosMediaMensual NUMBER(10,2) NOT NULL,
CONSTRAINT pk_productos_fin PRIMARY KEY (Codigo),
CONSTRAINT fk_productos_fin_productos_gen FOREIGN KEY
(Codigo) REFERENCES PRODUCTOS_GEN
);

CREATE TABLE DISTRIBUYE (
        Sede           VARCHAR2(7) NOT NULL,
        Producto       VARCHAR2(7) NOT NULL,
        Ventas         NUMBER(10,2) NOT NULL,
        FecIni         DATE NOT NULL,
        FecFin         DATE,
CONSTRAINT pk_distribuye PRIMARY KEY (Sede, Producto),
CONSTRAINT fk_distribuye_productos_gen FOREIGN KEY (Producto)
REFERENCES PRODUCTOS_GEN,
CHECK (FecIni < FecFin)
);

CREATE TABLE PEDIDOS (
        NumPedido      VARCHAR2(7) NOT NULL,
        Cliente        VARCHAR2(10) NOT NULL,
        FecPedido      DATE NOT NULL,
        DireccionEntrega VARCHAR2(2000) NULL,
        TotalSinDto   NUMBER(10,2) NOT NULL,
        Dto            NUMBER(10,2) NOT NULL,
        TotalConDto   NUMBER(10,2) NOT NULL,
CONSTRAINT pk_pedidos PRIMARY KEY (NumPedido),
CONSTRAINT fk_pedidos_clientes FOREIGN KEY (Cliente)
REFERENCES CLIENTES
);

```

```

CREATE TABLE LINEAS_PEDIDO (
    NumPedido      VARCHAR2(7) NOT NULL,
    Producto       VARCHAR2(7) NOT NULL,
    Unidades       INTEGER NOT NULL,
    Precio         NUMBER(10,2) NOT NULL,
    Importe        NUMBER(10,2) NOT NULL,
CONSTRAINT pk_lineas_pedido PRIMARY KEY (NumPedido,
Producto),
CONSTRAINT fk_lineas_pedido_pedidos FOREIGN KEY (NumPedido)
REFERENCES PEDIDOS,
CONSTRAINT fk_lineas_pedido_sur_productos FOREIGN KEY
(Producto) REFERENCES PRODUCTOS_GEN
);

```

Lanzaremos el código anterior en cada una de las sedes y después conectándonos de nuevo sede a sede lanzaremos las siguientes sentencias para particularizar sus fragmentos.

Para la sede *SUR*:

```

ALTER TABLE CLIENTES ADD CHECK (Sede='Sur');
ALTER TABLE DISTRIBUYE ADD CHECK (Sede='Sur');

```

Para la sede *CENTRO*:

```

ALTER TABLE CLIENTES ADD CHECK (Sede='Centro');
ALTER TABLE DISTRIBUYE ADD CHECK (Sede='Centro');

```

Para la sede *NORTE* es equivalente a las anteriores pero añadimos la tabla *SEDES* porque es la tabla maestra para la replicación:

```

ALTER TABLE CLIENTES ADD CHECK (Sede='Norte');
ALTER TABLE DISTRIBUYE ADD CHECK (Sede='Norte');

```

```

CREATE TABLE SEDES (
    Codigo          VARCHAR2(7) NOT NULL,
    Nombre          VARCHAR2(20) NOT NULL,
    Ciudad          VARCHAR2(20) NOT NULL,
CONSTRAINT pk_sedes PRIMARY KEY (Codigo),
CHECK (Codigo IN ('Centro', 'Norte', 'Este', 'Sur'))
);

```

Privilegios sobre objetos remotos. Del mismo modo que en el ejercicio anterior dado que no vamos a tener usuarios parásitos a los propietarios de las

tablas no es necesario la gestión de privilegios ni sinónimos.

Replicación. Las réplicas afectan a las relaciones *SEDES* y *PRODUCTOS*. La tabla *SEDES* ha de estar completa en todos los nodos, la frecuencia de actualización es mínima y los datos no parecen necesarios que estén sincronizados en todo momento ya que no son críticos. Por esto, en este caso debemos optar por una réplica diferida, bien sea el uso de **vistas materializadas** o **Oracle streams**, en este caso elegimos las vistas materializadas. Tendremos la tabla solo en la sede propietaria y en el resto de sedes utilizaremos vistas materializadas.

Crearemos la vista materializada en cada una de las sedes que necesitaban una réplica de *SEDES*, esto es:

```
CREATE MATERIALIZED VIEW V_SEDES
REFRESH FORCE START WITH SYSDATE
NEXT SYSDATE + 1
AS SELECT * FROM SEDES@TO_NORTE;
```

Como los datos de *SEDES* son muy pocos realizamos una actualización completa (*FORCE*), la realizaremos en el momento de la creación de la vista y todos los días a la misma hora. También podremos forzar el refresco con la siguiente sentencia:

```
exec dbms_refresh.refresh('V_SEDES');
```

Podemos comprobar que funciona insertando una sede en Norte y verificar que la vista materializada de Centro no se ha actualizado, después forzamos la actualización y veremos que sí está actualizada si volvemos a consultarla.

```
INSERT INTO SEDES VALUES ('Centro', 'Región Centro', 'Madrid');

SELECT * FROM V_SEDES@TO_CENTRO; /* Sin datos */
```

Nos conectamos a Centro para forzar la actualización de la vista materializada:

```
exec dbms_refresh.refresh('V_SEDES');

SELECT * FROM V_SEDES; /* 1 registro */
```

La replicación de *PRODUCTOS* es más crítica, se requiere disponer de la información actualizada en cada momento, por esto se opta por un mecanismo

de sincronización directa mediante disparadores. En el enunciado se indica que los datos financieros de productos solo se mantienen desde la sede Centro y se replican a todas las sedes, por tanto, en Centro se requiere el siguiente disparador:

```

CREATE OR REPLACE TRIGGER TG_PRODUCTOS_FIN
BEFORE INSERT OR UPDATE OR DELETE ON PRODUCTOS_FIN
FOR EACH ROW
BEGIN

IF INSERTING THEN
  INSERT INTO PRODUCTOS_FIN@TO_NORTE VALUES
    (:New.Codigo,:New.FacturacionTotal,:New.BeneficiosTotal,
     :New.FacturacionMediaMensual,:New.BeneficiosMediaMensual);

  INSERT INTO PRODUCTOS_FIN@TO_SUR VALUES
    (:New.Codigo,:New.FacturacionTotal,:New.BeneficiosTotal,
     :New.FacturacionMediaMensual,:New.BeneficiosMediaMensual);

END IF;
IF UPDATING THEN
  UPDATE PRODUCTOS_FIN@TO_NORTE A SET
    Codigo=:New.Codigo,FacturacionTotal =:New.FacturacionTotal,
    BeneficiosTotal =:New.BeneficiosTotal,
    FacturacionMediaMensual =:New.FacturacionMediaMensual,
    BeneficiosMediaMensual =:New.BeneficiosMediaMensual
  WHERE A.Codigo = :New.Codigo;

  UPDATE PRODUCTOS_FIN@TO_SUR A SET
    Codigo=:New.Codigo,FacturacionTotal =:New.FacturacionTotal,
    BeneficiosTotal =:New.BeneficiosTotal,
    FacturacionMediaMensual =:New.FacturacionMediaMensual,
    BeneficiosMediaMensual =:New.BeneficiosMediaMensual
  WHERE A.Codigo = :New.Codigo;

END IF;
IF DELETING THEN
  DELETE PRODUCTOS_FIN@TO_NORTE A
  WHERE A.Codigo = :Old.Codigo;

  DELETE PRODUCTOS_FIN@TO_SUR A
  WHERE A.Codigo = :Old.Codigo;

END IF;
END;
/

```

En el enunciado se indica que Centro también puede de forma esporádica modificar los datos generales de los productos. Dado que se trata de una modificación esporádica y que complicaría la gestión de las réplicas la mejor

opción es no replicar, es decir que las modificaciones se realicen accediendo directamente a la tabla de la sede Norte utilizando el enlace correspondiente.

En la sede Norte se deben crear disparadores para actualizar los *PRODUCTOS_GEN* en todas las sedes:

```
CREATE OR REPLACE TRIGGER TG_PRODUCTOS_GEN
BEFORE INSERT OR UPDATE OR DELETE ON PRODUCTOS_GEN
FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO PRODUCTOS_GEN@TO_SUR VALUES
            (:New.Codigo, :New.Nombre, :New.Publico, :New.Categoría,
            :New.PrecioActual, :New.FecInicio, :New.FecFin);
        INSERT INTO PRODUCTOS_GEN@TO_CENTRO VALUES
            (:New.Codigo, :New.Nombre, :New.Publico, :New.Categoría,
            :New.PrecioActual, :New.FecInicio, :New.FecFin);
    END IF;

    IF UPDATING THEN
        UPDATE PRODUCTOS_GEN@TO_SUR A SET
            Código=:New.Código, Nombre=:New.Nombre, Publico=:New.Publico,
            Categoría=:New.Categoría, PrecioActual=:New.PrecioActual,
            FecInicio =:New.FecInicio, FecFin =:New.FecFin
            WHERE A.Código = :New.Código;
        UPDATE PRODUCTOS_GEN@TO_CENTRO A SET
            Código=:New.Código, Nombre=:New.Nombre,
            Publico=:New.Publico, Categoría=:New.Categoría,
            PrecioActual=:New.PrecioActual, FecInicio=:New.FecInicio,
            FecFin =:New.FecFin
            WHERE A.Código = :New.Código;
    END IF;

    IF DELETING THEN
        DELETE PRODUCTOS_GEN@TO_SUR A
        WHERE A.Código = :Old.Código;
        DELETE PRODUCTOS_GEN@TO_CENTRO A
        WHERE A.Código = :Old.Código;
    END IF;
END;
/
```

Antes de finalizar se realizarán las comprobaciones necesarias introduciendo datos en las sedes y verificando que la BDD funciona correctamente. Por ejemplo, nos conectamos a Norte:

```
INSERT INTO PRODUCTOS_GEN VALUES ('01','Pantalla  
Total','Mujer','Solares','20',sysdate,NULL);  
SELECT * FROM PRODUCTOS_GEN@TO_SUR; /* 1 registro */  
SELECT * FROM PRODUCTOS_GEN@TO_CENTRO; /* 1 registro */
```

Capítulo 4

ORGANIZACIONES DE FICHEROS

A menudo se define dato como la “representación de una información de manera adecuada para su tratamiento informatizado”. Ahora bien, para posibilitar ese tratamiento es necesario recoger los datos de modo que estén disponibles cuando se les requiera. Por otro lado, el tratamiento de datos y su aprovechamiento no siempre ocurren en un momento o lugar cercano al de la adquisición. Por lo tanto, el registro de datos debe presentar dos vertientes: orientado al proceso (o almacenamiento principal, de inmediata disponibilidad), y orientado a la perdurabilidad y mayor alcance de los datos (o almacenamiento secundario).

El almacenamiento secundario se diferencia del primario en no ser volátil, tener un coste (por unidad de almacenaje) mucho menor, y ofrecer tiempos de respuesta mucho mayores (que se podrán ver compensados por la cantidad de información transferida en cada operación). Las diferencias en características y fines explican el distinto uso que se hará de cada uno, y en particular, el modo de estructurar la información que debe plantearse sobre soporte secundario para maximizar los procesos que afecten a esa información.

En las próximas secciones se hará un breve repaso a las estructuras de datos sobre soporte secundario orientado a la resolución de problemas de evaluación del rendimiento de conjuntos de procesos sobre organizaciones concretas. Los problemas propuestos y resueltos se plantearán de un modo simple y directo acerca de organizaciones individuales concretas, y no como parte de sistemas de archivos complejos ni esquemas internos de una base de datos, a fin de ganar en claridad de exposición y mejorar el aprovechamiento de los contenidos.

4.1 CONCEPTOS BÁSICOS

Para evaluar el rendimiento del manejo de información, se hace preciso definir unidades y su ámbito. De este modo, se tendrá que la unidad básica de información será el byte o palabra de máquina (habitualmente asimilado al octeto o conjunto de 8 bits por cuestiones históricas). Ésta puede ser en algunas máquinas la unidad de acceso a memoria, pero generalmente la cantidad de información que consume un proceso es mayor, y la cantidad que interviene en una operación sobre soporte secundario es siempre mayor.

Un archivo será una colección de informaciones que registran ocurrencias del mundo real, descritas mediante un conjunto de campos o elementos de datos. Los elementos de datos pueden ser simples (un campo) o compuestos, que a su vez se diferencian según sea la interpretación de sus elementos: si es complementaria se trata de un vector (por ejemplo, una fecha); pero si es común, se hablará de un grupo repetitivo (por ejemplo, el elemento de datos hijos). Cada una de esas ocurrencias se denominará registro (o registro lógico), y constituye la unidad de proceso del archivo. La definición por intención del archivo consistirá en la descripción del conjunto de campos que conforman el registro genérico, mientras que su definición por extensión la constituye el conjunto de registros que contenga en un momento determinado. Por otro lado, un fichero se puede definir como un área de almacenaje sobre soporte secundario. Un fichero puede contener uno o varios archivos (colecciones de informaciones de distinta naturaleza), pero también un archivo puede estar distribuido entre varios ficheros físicos. El fichero está compuesto por un conjunto de registros físicos o bloques. El bloque es la unidad mínima de acceso a soporte secundario, que lo caracteriza. El tamaño de bloque será el número de bytes que contiene, y en su relación con el tamaño del registro surge el factor de bloqueo (fb): número de registros lógicos que caben en un bloque.

El tiempo de acceso a bloque es el tiempo necesario para operar un bloque (para lectura o escritura indistintamente) y determina el coste en tiempo de los procesos, que se medirá como el número de accesos a bloque multiplicado por el tiempo de acceso a bloque. Para comparar organizaciones sobre el mismo soporte, a menudo se prescinde de este concepto y se compara directamente el número medio de accesos del conjunto de procesos sobre cada organización. Se definirá así el coste global $C(Oi,P)$ de una organización Oi sobre el conjunto de procesos P como la media aritmética del coste (en número de accesos) de todos

los procesos (media ponderada en caso de que no presenten la misma frecuencia).

Cualquier acceso a soporte (ya sea de lectura o de escritura) implica operar un bloque completo. Para manejar el bloque de un modo cómodo, se contará con memorias de inmediata disponibilidad (*buffers*) capaces de albergar el último bloque leído hasta que se complete su procesamiento, o de recoger el próximo bloque que se va a escribir hasta que esté preparado para esa operación. Solo un buffer de cada tipo supone una limitación muy grande, especialmente teniendo en cuenta que pueden ser muchos los ficheros abiertos en un sistema en un momento dado. Por eso es habitual contar con memorias intermedias, que posibilitan mantener disponible un mayor número de bloques (tanto de lectura como de escritura) y mejorar así un buen número de procesos. En particular, para garantizar la eficiencia de algunas organizaciones se precisa que determinados bloques tengan una disponibilidad inmediata (sin necesidad de acceder al soporte cada vez que se requieran), por lo que deberá procurarse que permanezcan en memoria intermedia.

El diseño lógico de un archivo consistirá en la descripción de la estructura lógica de sus registros, y debe cubrir las necesidades para las que se propone (debe ser eficaz). Por otro lado, el diseño físico del fichero determinará la organización de los registros y los mecanismos necesarios para resolver los procesos que lo afectan, también con el menor consumo de recursos posible (tiempo, espacio y coste). El diseño físico siempre será transparente al usuario (se variará la eficiencia, pero no la eficacia). Por último, el registro deberá ser implementado en secuencias de bytes que posibiliten su posterior recuperación y correcta interpretación. El diseño físico del registro lógico buscará reducir los recursos consumidos (espacio) para cubrir este objetivo (deberá ser eficiente). La solución trivial consistiría en reservar el máximo espacio que puede llegar a ocupar cada campo y grupo repetitivo, y las ocurrencias que no lo necesiten se completarían con bytes de relleno (diseño físico-lógico pésimo). Para mejorar esta aproximación, se podrán introducir campos de control o marcas, como las que se describen a continuación:

- Marca de **longitud**: fija el número de bytes de la ocurrencia del campo.
- Marca de **existencia**: señala si un campo opcional ocurre o no.
- Marca de **reiteración**: indica cuántas veces ocurre un grupo repetitivo en el registro (cuántos valores o conjuntos de valores presenta).

- Marca de **fin de campo**: se sitúa después del último byte de un campo de longitud indefinida.

Además de introducir marcas, se procurará reducir el volumen del registro mediante la codificación de campos. Consiste en sustituir el valor proporcionado por el usuario para cierto campo por otro de menor coste de almacenaje, siendo posible el proceso inverso para la recuperación. De este modo, se podrán codificar campos numéricos (en base 256, signo/magnitud, etc.); las fechas (con ventana de fechas, fecha juliana, etc.); y los campos alfanuméricos de dominio cerrado y reducido, se podrán enumerar los valores posibles (por ejemplo, dominios que contengan hasta 256 valores podrán enumerarse con un byte).

El número de bytes necesarios para el diseño físico de un registro es, por lo general, superior al número de bytes que realmente son útiles. La relación porcentual entre ambos (útil/real) en media se denominará *densidad ideal* del archivo, y es independiente de la organización del mismo (caracteriza la bondad del diseño físico-lógico del registro, no del diseño físico del archivo). Por otro lado, la relación entre el número de bytes útiles de todo el archivo (calculado multiplicando la media útil por el número de registros) y el número de bytes reales que consume en el soporte (número de bloques multiplicado por el tamaño del bloque), recibe el nombre de *densidad real*, y será siempre mayor (o a lo sumo igual) que la ideal. Finalmente, la *densidad de ocupación* se define como la relación porcentual entre los registros que contiene el archivo y los que podría contener (habida cuenta de su tamaño y su organización física).

Los registros pueden clasificarse, según la variabilidad del tamaño de las ocurrencias, en tres tipos: fijos (aquellos que siempre tienen el mismo tamaño); variables (distintos tamaños pero dentro de un margen); e indefinidos (de tamaño no necesariamente acotado). El tipo de registro según esta caracterización puede afectar a su manejo y a la organización que para ellos se disponga en el archivo.

Para interactuar con el archivo se utilizan claves, que son un campo o conjunto de campos con una misión específica. En los procesos selectivos (que afectan a un subconjunto de registros, en contraposición a los procesos a la totalidad que afectan a todo el archivo), tendrá que establecerse el criterio de selección, consistente en valuar un campo o conjunto de campos que recibirán el nombre de *clave de búsqueda*. Si la clave no presenta valores repetidos en el archivo (se puede establecer una biyección entre registros y valores de la clave) se tratará de una *clave de identificación*. Si la organización del archivo posibilita mecanismos especiales para la localización a través de una clave, se hablará de

clave privilegiada, denominando al resto *claves alternativas*. De este modo, en una organización secuencial la clave privilegiada es la *clave de ordenación* (física), en una organización direccionada la *clave de direccionamiento*, y en una organización indizada la *clave de indización*. En un proceso, los registros podrán ser requeridos aleatoriamente (proceso sin orden), o siguiendo un cierto criterio o *clave de ordenación lógica* (proceso ordenado). Toda clave debe cumplir la condición de minimalidad, que impone que ningún subconjunto suyo pueda desarrollar la misión de la clave completa. Por ejemplo, sobre un archivo de libros con los campos *signatura* (únívoco) y *año* (no único), la clave de ordenación *signatura+año* no es mínima porque la clave *signatura* por sí sola define el mismo orden, mientras que la clave de ordenación *año+signatura* sí es mínima (la clave *año* no define un orden completo y la clave *signatura* define otro orden).

La naturaleza de los procesos puede ser de dos tipos: recuperación (consulta), o actualización (inserción, borrado y actualización). Los de este último tipo determinan el volumen de cambios (volatilidad) que sufre el fichero por unidad de tiempo, que se reflejará en las tasas de volatilidad (porcentaje del volumen inicial de registros afectados por cada tipo de actualización), y en la tasa de crecimiento (físico del fichero o lógico del archivo).

4.2 ORGANIZACIONES BASE

La organización de un archivo determina la ubicación de cada uno de sus registros respecto del resto, y por tanto, el mecanismo necesario para su operación. Partiendo de esta idea, se podrán diferenciar dos tipos de organizaciones: las que determinan el posicionamiento físico de los registros en el momento de la inserción (organizaciones base) y aquéllas que posibilitan un mecanismo de ubicación alternativo (que mejora el proporcionado por la organización base). En este apartado se tratarán las primeras.

4.2.1 Organización serial

La organización más simple es la organización serial, según la cual los registros se ubican uno detrás de otro a medida que se van insertando. Escribiendo el primer byte del registro n siempre a continuación del último byte del registro $n-1$, se tendría una organización *serial consecutiva*. Un registro cualquiera podría estar completamente contenido en un bloque o, en el peor caso, ocupar varios ya que el registro a insertar podría no caber completamente en su bloque y los bytes que no cupiesen serían escritos al comienzo del bloque siguiente.

Dado que, en general, la posición de inserción es conocida a priori y que probablemente el bloque afectado para la escritura se encuentre en memoria intermedia, el proceso de inserción será óptimo en esta organización (a lo sumo un acceso de escritura en general; sin el efecto de la memoria intermedia, un acceso de lectura y dos de escritura en el peor caso). Los procesos a la totalidad también serán óptimos, debido a que esta organización no implica desaprovechamiento de espacio ni marcas de control (exceptuando las del diseño físico-lógico).

En los procesos selectivos, el número máximo de accesos (peor caso) coincidirá con el número de bloques del archivo (N), realizándose un recorrido serial completo del mismo (*full scan*). Por otro lado, el número medio de accesos para la localización dependerá del tipo de clave de búsqueda. Si la clave es identificativa, el proceso se detendrá al localizar el primer (y único) registro involucrado, por lo que el número medio de accesos responderá a la localización de la mediana: $(N+1) / 2$. Finalmente, si la clave de búsqueda presenta valores repetidos en el archivo, será necesario recorrer todo el archivo (coincidiendo, en

este caso, el número medio de accesos con el número máximo).

En lo referente al resto de operaciones de actualización, si los registros son de tamaño fijo se puede eliminar un registro sobreescribiéndolo con otro (ya que tienen el mismo tamaño). Concretamente, se tomaría el último registro del archivo (ubicado en el último bloque, habitualmente en memoria intermedia), se localizaría el registro a borrar, y se escribiría en su lugar el último registro (adelantando la marca de “fin de archivo” para no repetir ese registro). De este modo, el borrado no afectará significativamente a la densidad del archivo, ya que no se producen huecos, y podrá realizarse en uno o dos accesos de escritura (con memoria intermedia, en general). Para efectuar una modificación en este tipo de archivos, se localizaría el registro afectado y se sobreescriviría actualizado en la misma posición.

Sin embargo, si los registros son variables no se podrá utilizar este mecanismo (los registros no tendrán el mismo tamaño en general) y realizar un borrado físico sería muy costoso: implica adelantar todos los registros posteriores al borrado, tantos bytes como tuviera ese registro). En este caso, lo adecuado es realizar un *borrado lógico*: el registro se *marca* como borrado (escribiendo en sus primeros bytes una marca que indica que el registro está borrado seguida del tamaño del registro) pero no se elimina físicamente. Al recuperar el registro durante un recorrido del archivo, se comprueba que está borrado y se *salta*. Este mecanismo produce *huecos*, que disminuyen la densidad del archivo (se reduce su ocupación útil pero no su tamaño real), por lo que la organización degenera (con el transcurso del tiempo, el rendimiento de los procesos que se efectúan sobre el archivo empeoran). Para recuperar su eficiencia, será necesario realizar operaciones de mantenimiento (la compactación, que elimina todos los huecos en un borrado físico), o bien utilizar mecanismos de reutilización de huecos durante la inserción. Finalmente, las operaciones de modificación sobre archivos variables podrán realizarse borrando el registro (borrado lógico) y reinsertándolo modificado (al final del archivo).

4.2.2 Organización secuencial

Con el planteamiento serial, los procesos selectivos y los ordenados se ven claramente perjudicados. Si estos procesos fueran frecuentes, podría ser adecuado plantear una organización *secuencial* mediante cierta clave de ordenación (CO). En estas, la disposición física de los registros viene determinada por un criterio o *clave de ordenación* (física). Los procesos a la

totalidad seguirían siendo óptimos (por la alta densidad del archivo) pero en este caso, además, también podrán serlo los procesos a la totalidad ordenados, es decir, aquellos que deben recuperar los registros siguiendo un criterio o clave de ordenación (lógica). Los procesos ordenados cuya CO lógica coincide con la CO física del archivo se resolverán con un recorrido de todos los bloques (N accesos). Por otro lado, los procesos selectivos cuya clave de búsqueda sea identificativa y coincida con la clave de ordenación física podrán resolverse mediante búsqueda dicotómica, cuyo coste en número de accesos presenta la cota superior $\log_2(N+1)$.

Para los procesos selectivos cuya clave de búsqueda coincide con la clave de ordenación física pero ésta no sea clave de identificación, podrá aplicarse el algoritmo de búsqueda dicotómica extendida, que consta de tres pasos: 1) búsqueda dicotómica para localizar el primer registro afectado; 2) recorrido serial hacia atrás desde el registro localizado para recuperar todos los registros afectados que estén ubicados físicamente antes que el primero; y 3) recorrido serial hacia delante desde el primer registro localizado para recuperar los registros posteriores. Ambos recorridos seriales finalizarán cuando se recupere un registro que no esté afectado por el proceso (*fallo*), por lo que el número de registros que deberán recuperarse serán $m+2$, siendo m el número medio de registros afectados por el proceso (que encajan con el criterio de selección). El peor caso es que uno de esos fallos (por ejemplo, el anterior) ocupe un solo bloque, y además de este habría que leer los $\lceil(m+1)/f_b\rceil$ bloques que ocupan el resto de registros recuperados (donde f_b es el factor de bloqueo). Teniendo en cuenta que uno de esos bloques ya se había leído en el proceso de búsqueda dicotómica, la cota superior para el número de accesos queda $\log_2(N+1) + \lceil(m+1)/f_b\rceil$. Esta cota puede precisarse (en media) sustituyendo el número de bloques (N) por el número de zonas del archivo que tienen el mismo valor de clave de ordenación (e/m , donde e es el número total de registros del archivo), siempre que esa cantidad fuera inferior al número de bloques del archivo.

Las inserciones ordenadas presentan un grave problema, ya que requieren realizar un desplazamiento físico de todos los registros posteriores al insertado. Para evitarlo, suele crearse un área desordenada, que es un archivo serial independiente donde se realizan las inserciones. Todos los procesos de localización se ven perjudicados, ya que tras procesar el área ordenada (por cualquier proceso dicotómico o recorrido serial) se debe recorrer serialmente el área desordenada (en los mismos términos que cualquier organización serial), afectando así al coste en número de accesos. Por tanto, esta organización

también degenera y deberá contar con un proceso de *reordenación* que *mezcle* los registros de ambas áreas de datos (ordenada y desordenada) en una sola ordenada. Por otro lado, los procesos de borrado y modificación presentarán características similares a los descritos para la organización serial, con la salvedad de que la reutilización de huecos es mucho más compleja debida a la ordenación. Por ello, casi siempre el borrado será lógico y la modificación se hace por borrado y reinserción, incluso con registros fijos.

Finalmente, las organizaciones secuenciales presentan el problema de la localización de registros lógicos en un bloque aleatorio (ya que el primer byte del bloque no es necesariamente el primer byte de un registro lógico). Para solventarlo, se puede recurrir a marcas de separación de registros, a una marca que indique el desplazamiento del primer registro, o bien plantear organizaciones no consecutivas.

4.2.3 Organizaciones no consecutivas

En éstas, se introduce el concepto de *cubo* como el conjunto de bytes con la misma dirección relativa y que consta de bytes ocupados y bytes libres. En un cubo solo se almacenará un registro si cabe completamente (no existirán registros “partidos” entre dos cubos), y por tanto el primer byte ocupado de un cubo corresponde al primer byte de un registro lógico. Todos los cubos de un archivo tendrán el mismo *espacio de cubo* E_c (número de bloques que los componen, uno o más), y el concepto de *factor de bloqueo* se verá sustituido por otro similar: el *tamaño de cubo* T_c , o número medio de registros lógicos que caben en un cubo (que será siempre un número natural). Naturalmente, las organizaciones no consecutivas miden su eficiencia en *accesos a cubo*, y cada uno de ellos implicará tantos accesos a soporte como bloques lo compongan (*espacio de cubo*).

En las organizaciones no consecutivas suele desperdiciarse cierta cantidad de espacio al final del cubo (cuando ya no cabe otro registro en él). Aunque de este modo se reduzca en general la densidad de almacenaje, que implica cierto aumento del área de datos, también conlleva ciertas ventajas. Aumentando ese espacio, según convenga a las necesidades del archivo, se podrá constituir el *espacio libre distribuido* en los cubos (espacio que no se ocupa en cargas masivas ni reorganizaciones). Con este espacio se persiguen dos objetivos distintos, que dan lugar a sendas caracterizaciones: espacio para inserciones (posibilita inserciones ordenadas durante un tiempo, reduciendo el área

desordenada y demorando reorganizaciones); y espacio para modificaciones (posibilita que los registros se modifiquen habitualmente sin sacarlos del cubo, aunque la operación implique un aumento de tamaño del registro modificado).

De este modo, modificaciones y borrados implicarán (por lo general) una localización y un acceso de escritura. Las inserciones ya no se harán en un acceso (constarán de una localización y uno o dos accesos de escritura, según sea el porcentaje de desbordamiento de los cubos). Finalmente, las localizaciones se verán muy beneficiadas, ya que aunque aumenta el área de datos (se reduce la densidad de almacenaje), esto no afecta mucho al rendimiento de la búsqueda dicotómica (por lo general), y se evita el recorrido serial del área desordenada. Además, se reduce la necesidad de procesos de reorganización.

La *no consecutividad* podrá aplicarse también sobre organizaciones seriales, obteniendo así ventajas en la gestión y reutilización de huecos y en los procesos de modificación, sin afectar a los procesos de inserción. No obstante, los procesos que impliquen localización se verán muy perjudicados, y se hará necesario contar con organizaciones auxiliares que mejoren la eficiencia de la localización a través de las claves de búsqueda más frecuentes.

4.2.4 Organizaciones direccionadas

La mayoría de dispositivos actuales de almacenamiento posibilita acceder a bloques aleatoriamente (a través de su *dirección física* en el soporte), cualidad que podría ser aprovechada si se conociera de antemano la dirección del bloque que contiene el registro que se pretende localizar. Para simplificar la localización, se utilizará la *dirección relativa* de cada cubo, definida como su posición en el archivo y a partir de la que se puede calcular su dirección física (o direcciones, si el cubo estuviera formado por varios bloques). Tomando la clave de búsqueda más frecuente en las localizaciones, se buscará una función capaz de asignarle a cada valor de esa clave una dirección relativa.

De este modo, se plantean las organizaciones *direccionadas*, que precisan definir una clave privilegiada o clave de direccionamiento (CD), un espacio de direcciones relativas de cubo (o espacio de direccionamiento), y una función de transformación, como ya se ha expuesto. Se denomina *potencia de direccionamiento* de la CD a la cardinalidad del dominio sobre el que se define. A la hora de escoger el tamaño del espacio de direccionamiento (N) habrá que procurar que sea siempre menor que la potencia de direccionamiento de la CD (porque si fuera mayor, las direcciones excedentes estarían siempre vacías), pero

suficientemente grande para albergar todos los registros del archivo. La Figura 4.1 recoge los procesos de transformación necesarios para el direccionamiento: de una clave (ámbito lógico) se pasa a una dirección de cubo (o dirección relativa), y de esta se pasa a la dirección del bloque, o bloques en su caso (ámbito físico).

El direccionamiento directo será aquél en el que cada valor del dominio sobre el que está definida la CD tenga su propia dirección “reservada” (es decir, se puede establecer una biyección entre valores de la CD y direcciones relativas), bien sea a través de una transformación (directo relativo) o prescindiendo de ella (directo absoluto). El espacio reservado para cada dirección será de tamaño constante (un cubo, o una fracción del mismo denominada *celda*), y se procurará que sea suficientemente grande como para albergar al registro de mayor tamaño. En tales organizaciones, la operación de inserción se resolverá con un único acceso (a cubo), al igual que cualquier operación de consulta selectiva cuya localización se haga a través de la CD. Los borrados y modificaciones a través de la CD precisarán de un acceso de lectura y otro de escritura. No obstante, estas organizaciones suelen presentar densidades de almacenaje muy bajas (por la ocupación de los cubos o celdas, y por el número de cubos vacíos), que desemboca en áreas de datos excesivamente grandes. Por este motivo, el resto de procesos de localización se ven muy perjudicados, ya que deberán resolverse mediante recorrido serial de toda el área de datos. Para recuperar parte de la eficiencia en las localizaciones a través de clave alternativa (no privilegiada), se puede recurrir al uso de organizaciones auxiliares (que se expondrán en la siguiente sección) o al uso de directorios.

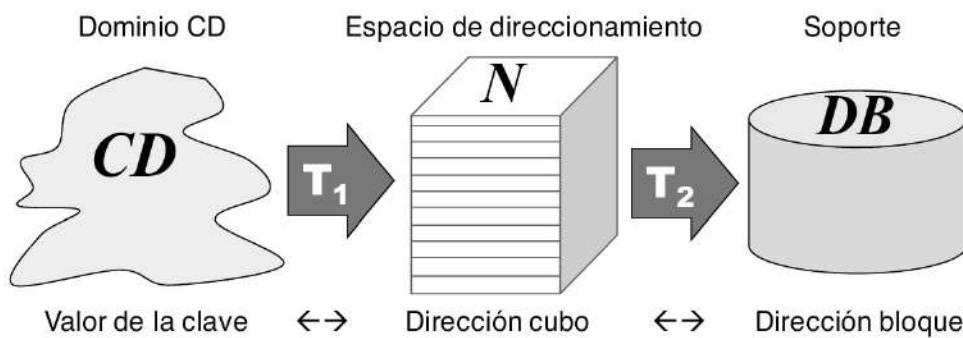


Figura 4.1. Transformaciones en una organización de direccionamiento

El *directorio de ocurrencias* consiste en almacenar un *bit* por cada cubo (o celda) del direccionamiento, en posiciones correlativas al área de datos. Cuando el archivo está vacío (sin ningún registro), todos los bits almacenan el *valor*

vacío (0), y a medida que se insertan registros se actualiza su bit de ocurrencia al *valor ocupado* (1). De este modo, se puede saber qué cubos están vacíos y no acceder a ellos en ningún proceso de localización. De este modo se reduce notablemente el número de accesos en un recorrido serial, aunque no se evita el desperdicio de espacio (los cubos vacíos siguen existiendo). Para evitar esto, también se puede contar con un *directorio virtual*, similar al anterior pero en el que cada posición contiene no un bit sino un puntero a cubo (dirección relativa). Inicialmente, todos los punteros serán nulos (*nil*) y el área de datos no tendrá ningún cubo asignado. Cuando se necesita insertar un registro en una dirección (cuyo puntero en el directorio será *nil*) se reservará un cubo nuevo (el primero disponible para el archivo), se apuntará su dirección en el directorio (en la posición localizada), y se almacenará el registro en el cubo correspondiente. Para localizar un registro, se averigua su dirección relativa (que se denominará *dirección virtual*) y con ella se accede al directorio para averiguar su *dirección real*. Se consigue así evitar no solo el acceso a cubos vacíos, sino también su almacenaje. Las organizaciones direccionaladas que cuentan con este tipo de directorios se denominan *direccionamientos virtuales*, y presentan el problema de que cualquier operación que se realiza sobre ellas precisa acceder previamente al directorio. Para garantizar la eficiencia, por tanto, es necesario disponer de memoria intermedia suficiente para mantener allí almacenado permanentemente el directorio completo.

Por otro lado, el aprovechamiento de los cubos (o *celdas*) en direccionamientos directos será bastante deficiente en archivos de registros variables, por lo que puede ser conveniente prescindir de la biyección entre valores de la CD y direcciones de cubo, utilizando funciones de transformación que asignan la misma dirección a distintos valores de la CD (que se denominarán *sinónimos*), y procurando dispersar lo más uniformemente posible todos los valores del dominio de la CD sobre el espacio de direccionamiento. La organización resultante será un *direccionamiento disperso*, y presenta el inconveniente de que los cubos, eventualmente, puedan recibir más registros de los que pueden almacenar, produciéndose en estos casos un desbordamiento que es preciso gestionar. Para reducir el número de desbordamientos, se puede aumentar el espacio de cubo y/o el espacio de direccionamiento, aunque esto tiene como contrapartida el probable aumento de cubos vacíos (que pueden evitarse aplicando un directorio virtual) y sobre todo el descenso de la densidad de almacenaje, con la consiguiente pérdida de eficiencia en procesos de localización a través de claves no privilegiadas. También puede intentarse mejorar la dispersión (cambiando la función de transformación), pero esta

solución no es definitiva y no es siempre posible. Además, el direccionamiento disperso puede aplicarse sobre CD que no sean identificativas, por lo que al problema de la sinonimia se une el de la homonimia (varios registros con el mismo valor de CD, que hace imposible su dispersión). Por todo ello, es imprescindible implementar mecanismos para la gestión de registros desbordados.

Las políticas de gestión de desbordamientos se dividen en dos tipos: de *saturación*, que almacena los registros desbordados en otra dirección del área de datos; y basadas en área de desbordamiento, que contarán con un área de almacenamiento independiente para los registros desbordados. El primer tipo presenta varios inconvenientes: se eleva el número de registros que no caben en su dirección (y desbordan), los cubos tienen registros de otras direcciones (que se procesarán innecesariamente) y se hace necesario un mecanismo para localizar registros desbordados. El *direccionamiento abierto* consiste en calcular la nueva dirección a partir de la dirección desbordada (mediante una función). En estos términos y en el peor caso, el proceso de localización a través de CD implicaría recorrer todo el área de datos: primero se accedería a la dirección obtenida transformando la CD, y luego se sondearían el resto de cubos hasta que se encuentre el registro buscado (cuando la CD sea identificativa), se llegue a un cubo no desbordado, o se termine de sondear todos los cubos del área de datos. Otra alternativa es almacenar en el cubo desbordado la dirección (puntero con partes alta y baja) del primer registro desbordado, y en cada registro desbordado incluir un puntero al siguiente registro desbordado. A esta técnica se le denomina saturación encadenada y, aunque implica aumentar ligeramente el tamaño de los registros, mejora, en general, el acceso a través de CD. En el peor caso requiere tantos accesos como registros sinónimos más uno, pero en realidad se comporta bastante mejor dado que permite escoger cualquier dirección para almacenar el registro desbordado (en lugar de utilizar una función predefinida). Por ejemplo, se podrá tomar un cubo de baja ocupación, o bien un cubo que ya tenga elementos desbordados de esa dirección, etc.

Incluir un área de desbordamiento, por otro lado, implica aumentar el número de cubos de datos e inevitablemente afecta negativamente a los procesos de localización a través de claves no privilegiadas. Al ser un archivo lógico en sí mismo, el área de desbordamiento puede tener una organización independiente (por ejemplo, serial). En el peor caso, la localización a través de la CD implicaría un acceso al área de datos y tantos accesos al área de desbordamiento como fueran necesarios (según su organización; en el caso de ser serial, tantos como cubos la conformen). Para calcular el coste medio de localización a través de CD

identificativa, sería necesario saber cuántos registros desbordan y cuál es el coste medio de su localización en el área de desbordamiento. De este modo se podrá calcular ese coste medio como la media ponderada del coste a cada registro. Es decir, el número de registros no desbordados multiplicado por uno (localizarlos cuesta un acceso a cubo), más el número de registros desbordados multiplicado por el coste medio de acceso a esos registros, y todo ello dividido entre el número total de registros del archivo.

Para mejorar la gestión mediante área independiente, se puede utilizar el encadenamiento (en términos similares a los expuestos en el párrafo anterior), planteando así un área de desbordamiento encadenada a registro cuyo coste es similar al de la saturación encadenada, con la salvedad de que en este caso todos los registros que se encontrarán en el primer cubo accedido (en el área de datos) son sinónimos del buscado (y eventualmente, relevantes en la búsqueda). Una alternativa de encadenamiento consistiría en tomar un cubo entero del área de desbordamiento para cada dirección desbordada, apuntando su dirección en el cubo desbordado, y almacenando allí solo elementos sinónimos (de esa dirección). Esta política de gestión, sobre área de desbordamiento encadenada a cubo, mejorará la localización de registros a través de la CD. El coste de este proceso en el peor caso (y en todo caso si la CD es no identificativa) se calculará como un acceso a cada cubo relevante (el cubo de datos y cada cubo de desbordamiento encadenado). En caso de que la CD sea clave de identificación, el coste medio se calcularía como el total de cubos relevantes más uno y todo ello dividido entre dos. Aunque este proceso se ve claramente beneficiado, hay que señalar que la densidad de almacenaje del área de desbordamiento desciende, perjudicando gravemente al coste de localización a través de cualquier clave de búsqueda no privilegiada.

Además de las organizaciones direccionaladas por una CD (simples), también existen las que contemplan varias CD_i simultáneamente (*direccionalamiento multiclave*). En ellas, cada CD_i tiene su propia transformación que produce una dirección sobre un subespacio N_i . En la inserción, se calcula cada sub-dirección y luego son concatenadas obteniendo una dirección global sobre el espacio $N = \Pi N_i$. Por ejemplo, considérese CD_1 : nombre y CD_2 : apellido con $N_1 = N_2 = 10$, y sendas funciones de transformación t_1 y t_2 . Se tiene un registro (*John, Doe*) para el que $t_1(John) = 7$ y $t_2(Doe) = 2$. El registro se almacena, por tanto, en la dirección 72, y si al buscarlo se cuenta con ambas claves de búsqueda (CD_1 y CD_2) se accederá solo al cubo 72. Pero en el caso de utilizar solamente la clave

de búsqueda CD_1 ya no será necesario recorrer toda el área de datos, pues se sabe que la dirección es $7x$, con $x \in N_2$ (es decir, las direcciones relevantes abarcan desde la 70 hasta la 79). Análogamente, si se conoce solo el valor para la CD_2 , se sabrá que las direcciones relevantes son de la forma $x2$ con $x \in N_1$ (las que terminan en 2).

Existe otro tipo de organizaciones direccionadas que gestionan automáticamente el espacio de direccionamiento. Se basan en una transformación que produce direcciones de l bits que son inicialmente truncadas a $l_0 < l$ bits. Cuando un cubo desborda, el truncamiento se realiza con un bit más $l_i < l$, repartiéndo todos los registros sobre un área de datos del doble de tamaño. Esta operación se denomina *desdoblamiento*, y su inversa (reconcentrar los registros en la mitad del área de datos utilizando un bit menos) es el *plegado*. El desdoblamiento produce demasiados accesos y cubos vacíos. Para reducirlos se añade un directorio virtual, y la organización resultante se denomina *dispersión virtual extensible*. Así, el desdoblamiento duplica el tamaño del directorio y el cubo desbordado (sin alterar el resto de cubos de datos). Con clave de direccionamiento no unívoca, la homonimia (muchos registros con el mismo valor de CD) puede producir desdoblamientos en cadena, contingencia que debe ser controlada y resuelta con gestión de desbordamientos (típicamente, en área independiente con encadenamiento a cubo).

Este directorio puede también almacenarse siguiendo una estructura de árbol binario, que posibilita que el desdoblamiento no afecte a todo el directorio sino solo al nodo que representa al cubo desbordado. Se trata entonces de una organización direccionada con dispersión dinámica. Si además es multclave, la dispersión extensible presenta la desventaja de tener que fijar para todo el fichero en cada desdoblamiento a qué dimensión (CD_i) pertenece el nuevo bit, siguiendo un patrón regular o almacenado que se denomina *rejilla*. En contraposición, la dispersión dinámica permite decidir para cada dirección virtual desbordada (representada por un nodo en el árbol binario) a qué dimensión pertenece el siguiente bit, produciendo dispersiones más versátiles.

4.3 ORGANIZACIONES AUXILIARES

Como se ha expuesto en la sección anterior, la organización base de un archivo determina los mecanismos que deberán utilizarse para resolver cada uno de los procesos que le afectan. En el peor caso, un proceso debería realizar un recorrido serial de toda el área de datos para resolver determinada operación. En otros casos, la organización base proporciona mecanismos de mayor eficiencia para resolver determinadas operaciones a través de una clave privilegiada. Sin embargo, a menudo son frecuentes los procesos que deben resolverse a través de claves alternativas (no privilegiadas). Por este motivo, se busca incluir organizaciones auxiliares que mejoren la eficiencia de esos procesos, procurando equilibrar el beneficio con el coste (ya que aunque se mejore la eficiencia de algunos procesos de selección, los procesos de actualización se verán perjudicados por la necesidad de incluir accesos de escritura para actualizar estas estructuras auxiliares). Las organizaciones auxiliares son independientes de la organización base, pudiéndose incorporar o eliminar en cualquier momento sin que afecte en absoluto a la organización base (aunque, lógicamente, los beneficios en eficiencia dependen de su existencia). Las organizaciones auxiliares más comunes son los índices.

Además del beneficio directo por reducir el número de accesos en determinados procesos, estas estructuras auxiliares suelen presentar un tamaño muy reducido (en comparación con los archivos de datos), pudiendo habitualmente estar completa o parcialmente almacenados en memoria intermedia (*caché*). Los accesos a este tipo de almacenamiento son muy rápidos (cercanos a la memoria principal y muy lejos del coste de acceso a soporte secundario), por lo que habitualmente no son tenidos en cuenta al calcular el coste efectivo (o real) de un proceso. Así, definiendo la tasa de acierto en memoria intermedia (*cache hit ratio*) como la relación de accesos resueltos en caché entre el número total de accesos, el coste efectivo de un proceso se calcula como el coste global del mismo (accesos totales) multiplicado por uno menos la tasa de acierto: $CE = CG * (1 - \text{hit ratio})$. El uso de índices aumenta notablemente esta tasa de acierto, por lo que el rendimiento final del sistema se ve muy mejorado.

4.3.1 Índices primarios, secundarios y *clusters*

Un índice es una estructura auxiliar formada por *entradas* que contienen dos punteros: uno lógico o *clave de indización (CI)*, y otro relativo. Se utilizará para *traducir* valores de la *CI* en su dirección relativa, y mejorar así la eficiencia de la localización a través de esa clave. En general, no se plantean índices sobre claves ya privilegiadas, salvo cierto tipo de índices (*no densos*) y aquellos otros que supongan un beneficio sobre procesos concretos. Una *CI* puede contener varios campos e incluso transformaciones sobre los mismos, como por ejemplo el resultado de aplicar una operación aritmética sobre una o más columnas, o el resultado de aplicar una función sobre cadenas de caracteres (p.ej., *uppercase*).

Si la *CI* es clave de identificación se tendrá un índice primario, y el resultado esperado es único (una entrada o ninguna). En cambio, procesando un índice secundario (cuya *CI* no es identificativa) se podrá obtener un resultado, muchos o ninguno. Por otro lado, un caso particular es la estructura agrupada o *cluster*, cuyas entradas contienen un valor para la *CI* y dos o más cadenas de punteros definidas sobre otros tantos archivos (o tablas en una BD). El *cluster*, además de los beneficios de un índice simple, presenta la ventaja de agilizar las operaciones de combinación de varios archivos relacionados a través de la *CI* en procesos de selección de registros que involucran la combinación de esos archivos (por ejemplo, una combinación natural en *BD* relacionales; véase Capítulo 2).

El coste de acceso de un proceso a través de un índice se calculará como la suma del número de accesos al índice más el número de accesos a cubos de datos. Debe observarse que los procesos de actualización podrán afectar a los índices aun cuando no intervengan en la localización: las inserciones y borrados de registros podrán implicar la consecuente inserción o el borrado de entradas en todos los índices (dependiendo del tipo de índice); por otro lado, la modificación de un registro puede alterar su posición física, por lo que habría que actualizar esa información en todos los índices. Además, la eventual modificación del valor en una *CI* implicaría la actualización de su entrada en el índice. En definitiva, la inclusión de índices produce ventajas pero también claros inconvenientes, por lo que habrá que analizar su conveniencia respecto a las frecuencias de todos los procesos que afecten al archivo.

El índice es un objeto independiente con su propia organización. La estructura básica es el índice serial, en el que las entradas se organizan como un archivo serial, presentando los costes asociados a esa organización: recorrido completo en el peor caso, y el acceso a la mediana como coste medio para índices primarios seriales. Si el índice es secundario, su uso implicará siempre

un recorrido completo a menos que se cambie la definición de la entrada.

En efecto, para índices secundarios se pueden plantear entradas formadas por CI y una *lista de punteros* (que a su vez consta de marca de longitud de la lista, y tantos punteros como indique la marca). Cada valor del dominio sobre el que esté definida la CI aparecerá una sola vez en el índice, y vendrá acompañado de un puntero a cada registro que presente ese valor de CI . De este modo, al insertar una entrada se comprobaría su existencia y, en caso de existir la clave, se añadiría solo su puntero. En consecuencia, el tamaño de las entradas varía frecuentemente, y por ello esta aproximación vendrá soportada por la no consecutividad del archivo índice, lo que posibilitará que la entrada crezca cuando lo necesite. En caso de no caber en su cubo, se trasladaría la entrada completa al final del archivo, generando un hueco en ese cubo (para la modificación del resto de entradas). En tales términos, las localizaciones, borrados, y modificaciones se harán como si el índice fuera primario.

El índice también puede ser ordenado, ofreciendo un rendimiento análogo al de las organizaciones de este tipo. Por otro lado, si el área de datos estuviera también ordenada por el mismo criterio (organización secuencial con clave de ordenación física igual a la clave de indización) se podría averiguar en qué cubo está cualquier registro con solo conocer el valor de la CI del primer registro de cada cubo. Con estas premisas se plantea el índice no denso, que solo registra la primera entrada de cada cubo y tiene, por tanto, un tamaño reducido. El acceso a la totalidad (ordenado o no) se realizará mediante recorrido serial, mientras que el selectivo (para un valor o un rango) se hará a través del índice.

Para los índices secundarios ordenados también se podrán aplicar listas, con la salvedad de que si desborda un cubo habrá que llevarse una de sus entradas completa (la de menor cantidad de punteros) a un área desordenada, causando la consiguiente necesidad de reordenación periódica. Finalmente, el índice también puede ser disperso, aunque esta opción estará sujeta a la existencia de una buena dispersión para la CI .

Si el archivo de índices fuera demasiado grande, el rendimiento del acceso indizado podría ser inferior a las expectativas. Por esto, para índices primarios o secundarios con listas, se puede optar por no incluir todas las entradas sino un subconjunto de ellas, obteniendo un índice parcial (en contraposición con un índice exhaustivo, que sí tendría una entrada por cada posible valor del dominio de la CI). Por ejemplo, se pueden indizar los valores más frecuentemente buscados, o los más repetidos (si es secundario), etc. Al utilizar estos índices, si la búsqueda no tiene éxito (fallo) no se podrá determinar si (a) no existen

registros para ese valor, o (b) no están indizados. Por ello, será necesario repetir la búsqueda sobre la organización base (habitualmente la *CI* no estará privilegiada, y esto implicará un recorrido serial). El coste máximo empeorará (al recorrido serial se suma el acceso al índice), pero el coste medio puede mejorar bastante. Este se calculará como la media ponderada del coste de acceso a cada registro (número de registros indizados multiplicado por el coste medio para su acceso, más el número de registros no indizados multiplicado por el coste medio de acceso a estos registros, dividido entre el número total de registros).

4.3.2 Organizaciones indizadas multinivel

Si se tuviera un índice ordenado que fuera demasiado grande, para reducir el coste de procesarlo se podría plantear realizar un índice no denso del índice. Ese índice también estaría ordenado, y podría ser objeto de otro índice, y así sucesivamente podrían formarse varios “niveles de índice” hasta llegar a uno cuyas entradas quepan en un solo cubo. Este cubo sería el nodo *raíz* de una estructura arbórea (nivel 1), mientras que los cubos del índice original serían los nodos hoja (nivel *n*). Este tipo de estructura se denomina índice multinivel. Los punteros de las entradas de nivel *n* se denominan *punteros externos* (apuntan a un cubo de datos), mientras que los del resto de cubos se denominan *punteros internos* (apuntan a otro cubo del índice). El coste de recuperar una entrada en este tipo de índices es igual al número de niveles que contenga (*n* accesos), si bien puede ahorrarse un acceso si se dispone de memoria intermedia para almacenar constantemente el nodo raíz.

El problema que plantean estos índices es su comportamiento ante operaciones de actualización. Aunque existen algunos recursos para reducir su impacto (como el espacio libre distribuido) en caso de que estas operaciones sean frecuentes se suelen utilizar subtipos de este tipo de estructuras: los índices arbóreos que crecen de abajo a arriba (B).

4.3.2.1 ORGANIZACIONES INDIZADAS ARBÓREAS

El primer tipo de índice arbóreo a analizar es el árbol binario. Sus entradas se componen de clave, puntero externo, y dos punteros internos. Presenta, como problema principal, la fuerte dependencia de su comportamiento al orden de almacenamiento de las entradas (que puede provocar el desequilibrio del árbol y que entradas contiguas en procesamiento no se almacenen próximas

físicamente). El desequilibrio se puede reducir con el enfoque AVL, que aplicará procesos de rotación (reorganización local) para garantizar que el árbol siempre sea 1-equilibrado (las ramas de mayor y menor longitud se diferencian a lo sumo en un nivel), e implantando así una cota superior en el número de nodos procesados en cada consulta ($1.44 * \log_2(e + 2)$, donde e es el número de entradas del índice). No obstante, si los nodos que se procesan contiguamente están almacenados en bloques distintos, recuperar una entrada resultará costoso (además, las rotaciones son eficientes sobre memoria principal, pero pueden significar cuatro accesos a bloque sobre soporte secundario). Por tanto, si las operaciones de actualización son frecuentes su rendimiento será muy bajo, a menos que estén completamente contenidos en memoria intermedia. Mención aparte merecen los índices arbóreos AVL paginados, que a todo lo anterior suman mantener cierta proximidad física en las entradas próximas en su procesamiento, manteniendo fija la posición de cada nodo en el índice. Cada página (cubo) del índice almacenará un subárbol completo (restringiendo la necesidad de punteros internos a las hojas de cada subárbol), por lo que se podrán procesar varios nodos seguidos de cada página y las rotaciones estarán más localizadas, reduciendo significativamente el número de accesos (en detrimento de la densidad de almacenamiento del índice que será baja).

Por otro lado, los índices por árboles B ofrecen un comportamiento medio bastante aceptable (incluso sobre archivos muy volátiles). Sus nodos se caracterizan por poder contener un número elevado de entradas (k), que servirán para realizar $k + 1$ particiones del espacio de búsqueda. A cada partición se le asignará un descendiente, y al número máximo de descendientes se le denominará *orden* del árbol ($m = k + 1$). Dado que el espacio de un nodo es conocido y tiene que dar cabida a m punteros y k entradas, se calculará m a partir de la siguiente expresión: $m * T_{\text{ptr_interno}} + k * (T_{\text{clave}} + T_{\text{ptr_externo}}) \leq \text{espacio}_{\text{nodo}}$

La inserción se hará siempre en las hojas, de modo que si al insertar sobre una llena esta desbordara ($k + 1$ entradas), entonces se creará una nueva hoja para repartir sus entradas entre ambas hojas. En el antecesor a la hoja desbordada existe una partición del espacio de búsqueda (representada por un puntero interno a su nodo) que ahora ha de dividirse en dos (la segunda representada por un puntero al nuevo nodo). Para este fin, de las $k + 1$ entradas se escogerá la mediana, que *promocionará* al nodo antecesor para diferenciar ambas particiones. Las k entradas restantes se repartirán entre las dos hojas. Naturalmente si al ingresar la entrada promocionada en el nodo antecesor éste

desbordara, se ejecutaría sobre él un proceso de partición/promoción completamente análogo. Finalmente, si el nodo que desborda es la raíz, se creará una nueva raíz que solo contendrá una entrada (la promocionada) y sendos punteros internos (a los nodos resultantes de la partición). La operación inversa se denomina *plegado*, y ocurre si al borrar una entrada el contenido de dos nodos descendientes contiguos más su entrada discriminante pueden almacenarse en un solo nodo (este criterio suele flexibilizarse, para reducir desbordamientos y plegados).

De este modo, cualquier nodo es el resultado de repartir k entradas entre dos nodos, por lo que la ocupación mínima (k_{\min}) será la parte entera de $k/2$. El número mínimo de descendientes (m_{\min}) será una unidad mayor que k_{\min} . La excepción es el nodo raíz, que puede contener una sola entrada y, por tanto, dos descendientes. Con estas premisas, se pueden calcular cuantos nodos y entradas contiene como mínimo cada nivel del árbol, hasta llegar a un punto (nivel $n + 1$) en el que el número mínimo de entradas sea superior al número total de entradas a indizar, y por tanto el nivel anterior (n) sea el de las hojas. En la Figura 4.2 se muestra este proceso y la cota superior del número de niveles del árbol.

nivel	nodos	entradas	acumulado
1	1	1	1
2	2	$2 \cdot k_{\min}$	$1 + 2 \cdot k_{\min}$
3	$2 \cdot m_{\min}$	$2 \cdot m_{\min} \cdot k_{\min}$...
...
→ n	$2 \cdot m_{\min}^{n-2}$	$2 \cdot m_{\min}^{n-2} \cdot k_{\min}$	$(2 \cdot m_{\min}^{n-1}) - 1$
$n+1$	$2 \cdot m_{\min}^{n-1}$	$2 \cdot m_{\min}^{n-1} \cdot k_{\min}$	$(2 \cdot m_{\min}^n) - 1$

Cota Superior:

$$n \leq 1 + \log_{\lfloor \frac{m+1}{2} \rfloor} \left(\frac{e+1}{2} \right)$$

$\leq e$

$> e$

(este nivel es imposible)

Figura 4.2. Cota superior a la profundidad de un árbol B

Además del acceso al área de datos, una recuperación requiere una localización en el árbol, y cualquier operación de actualización requerirá, en general, una localización más un acceso de escritura. Cuando ocurre un desbordamiento, se harán dos accesos extra de escritura (localización + 3 accesos), y cuando se realiza un plegado se precisan dos accesos extra (uno de lectura y otro de escritura). Finalmente, se puede establecer una cota superior al tamaño del archivo índice calculando el número máximo de nodos no raíz más

uno, es decir, $T_{índice} \leq 1 + \lfloor (e-1) / k_{mín} \rfloor$.

Uno de los problemas que presenta esta estructura está asociado a su baja densidad. Para mejorarla, se plantea el árbol B_* , que incluye procesos de rotación: cuando un nodo desborda y su vecino (siguiente descendiente en el antecesor) tiene espacio para albergar el desbordamiento, la entrada desbordada se promociona al antecesor, y el elemento descriptivo de la partición que este contenía se exonera insertándolo en el nodo vecino (siguiente descendiente). Sin embargo, si el nodo vecino tampoco tiene espacio se producirá la partición/promoción añadiendo un nodo vacío. Se tendrán dos nodos completos ($2 * k$ entradas), la entrada discriminante en el antecesor, y la que provoca el desbordamiento. Las tres particiones (tres nodos resultantes) necesitan 2 entradas en el nodo antecesor para discriminar, y el resto de entradas ($2 * k + 1 + 1 - 2$) se reparten entre los tres nodos. Por lo tanto, la ocupación mínima será $k_{mín} = \lfloor 2 * k/3 \rfloor$, y el número mínimo de descendientes ($m_{mín}$) una unidad mayor. La densidad de ocupación mínima en estos índices será (aproximadamente) de dos tercios, por lo que los índices serán más densos. Todos los cálculos son análogos a los realizados sobre árboles B .

Otra variante es el árbol B_+ , caracterizado por mantener todas sus entradas en los nodos hoja, siendo los elementos discriminantes de los antecesores copias de la primera entrada de cada nodo hijo. En los nodos hoja no habrá punteros internos, salvo el puntero de *encadenamiento de las hojas*, incluido para posibilitar el recorrido ordenado de las hojas. En cambio, los nodos no hoja no tendrán punteros externos, por lo que aumenta el orden (y se reduce la profundidad). Para el cálculo de la ocupación de sus hojas y el número máximo de descendientes (orden del árbol B_+) se tendrán las siguientes inecuaciones:

$$T_{ptro_interno} + k_{hoja} * (T_{clave} + T_{ptro_externo}) \leq espacio_{nodo}$$

$$m * T_{ptro_interno} + (m - 1) * T_{clave} \leq espacio_{nodo}$$

En una hoja, la partición de k elementos más el desbordado se hace repartiéndolos en dos nodos (el que promociona es una copia), y así la ocupación mínima será $k_{mín} = \lfloor (k + 1) / 2 \rfloor$. La partición de nodos no hoja es igual que la de un nodo de árbol B , y por tanto su número mínimo de descendientes se calcula como $m_{mín} = \lfloor (m + 1) / 2 \rfloor$. El número máximo de hojas (nodos de nivel n) se calculará dividiendo el número de entradas (e) entre la ocupación mínima de un

nodo: $\text{nodos}(n) \leq \lfloor e / k_{\min} \rfloor$. El número máximo de nodos de un nivel menor (menor que n) se definirá recursivamente: $\text{nodos}(n - i) \leq \lfloor \text{nodos}(n - i + 1) / m_{\min} \rfloor$. De este modo, se irá calculando el número de nodos de cada nivel hasta llegar a la iteración x en la que se tenga que $\text{nodos}(n - x) \in (0, 2)$. Llegado este caso, se identifica ese nivel como el correspondiente a la raíz del árbol, que tiene un nodo ($\text{nodos}(n - x) = 1$). Dado que el nivel de la raíz ($n - x$) es el nivel 1 y x es conocida, se puede despejar la incógnita del número de niveles del árbol ($n = 1 + x$). El tamaño máximo del archivo índice se calculará como el sumatorio $T_{\text{índice}} \leq \sum_{1..n} \text{nodos}(i)$.

El rendimiento de estos árboles es bueno (similar al de los árboles B pero con mayor orden, que produce árboles de menor profundidad). Además, permite recorrer las hojas ordenadamente para procesos a la totalidad (o selección por rangos grandes). Los árboles para índices secundarios se implementarán en general por listas (como se expuso anteriormente), lo que favorecerá aún más la implementación por árboles B_+ . Por último, la clave en nodos no hoja de estos árboles suele ser un prefijo (la parte mínima de la clave necesaria para discriminar), por lo que su tamaño medio será inferior al total (una fracción de éste).

Finalmente, cabe señalar que se puede utilizar cualquier estructura arbórea de la familia B para almacenar registros completos (en lugar de entradas de índice), convirtiéndose así estas estructuras en una alternativa para las organizaciones base. No obstante, debe observarse que la eficiencia de esta organización dependerá de la profundidad del árbol y ésta de su orden, por lo que el tamaño del registro debe ser muy pequeño con respecto al espacio del cubo (tamaño de página).

4.4 ACCESO MULTICLAVE

La ejecución de algunos procesos requiere de la intervención de varias claves. Por ejemplo, se puede realizar una selección de registros a través de dos o más claves de indización simultáneamente, en cuyo caso el coste será la suma de accesos sobre cada índice y el área de datos. Por lo tanto, la intervención de más claves puede parecer más costosa, aunque en general no lo sea: si se accede a un segundo índice es porque se espera que el ahorro en número de accesos al área de datos sea superior al coste de acceder a ese segundo índice. Como un índice primario solo devolvería un resultado (y este índice sería el primero en ejecutarse), a menos que el segundo índice tuviera coste cero no llegaría a ejecutarse (es preferible recuperar el cubo de datos a consultar el índice). Se concluye que, en general, los accesos multiclave indizados se harán sobre índices secundarios.

En lo referente a las organizaciones base, para un recorrido serial el coste es el mismo, sea cual fuere el número de claves involucradas. Un direccionamiento disperso, por otro lado, se puede organizar en base a varias claves dividiendo el espacio de direcciones relativas (N) en varios sub-espacios (N_i), uno por cada clave de direccionamiento (CD_i). De este modo, se tendrían varias transformaciones ($f_i: CD_i \rightarrow N_i$) que al insertar un registro darían lugar a varias direcciones parciales. Éstas se combinarán en una sola sobre el espacio de direccionamiento global (concatenando subespacios). Si en la localización se cuenta con todas ellas, el coste será como el de una dispersión simple. Sin embargo, si se cuenta con un subconjunto de CD_i disponibles, el número de accesos será tan grande como el espacio global dividido entre el tamaño de cada subespacio resuelto (para cuya CD_i se dispone de un valor). Matemáticamente: $N / \prod_i N_i$.

A la hora de establecer el tamaño de los subespacios en un direccionamiento disperso multiclave, se habrá de tener en cuenta la probabilidad de uso (p_i) de cada clave. Se puede calcular así el tamaño del primer subespacio $N_1 = (N + \log_2(p_1 / \sum_{j \neq 1} p_j)) / 2$. El resto del espacio ($N - N_1$) quedaría para la otra clave si fueran dos, y si fueran más se procedería análogamente descartando la clave ya calculada (p_1) y utilizando $N' = N - N_1$.

4.4.1 Acceso invertido

Los procesos de consulta no siempre precisan recuperar todo el registro, sino que en ocasiones se puede requerir solo un campo o un conjunto de campos. En tales casos, se puede enfocar como un proceso multiclave (con claves de búsqueda y de selección). Si todas las claves involucradas están indizadas, se podría resolver este proceso sin acceder al área de datos, en dos pasos:

- a) **Selección:** accediendo a los índices correspondientes a las claves de búsqueda, se identifican las posiciones relativas de los registros involucrados.
- b) **Proyección:** sobre los índices correspondientes a las claves proyectadas (*lista de selección*), se localizarán las entradas conteniendo los punteros relativos obtenidos anteriormente, recuperando así el valor de la clave proyectada de cada uno de los registros involucrados en la selección.

Para posibilitar este proceso, denominado *acceso invertido*, los punteros relativos de estos índices tienen que tener carácter identificativo, constando de parte alta (en qué bloque está el registro) y de parte baja (cuál es el registro de entre los que contiene ese bloque). Los índices que tienen estos punteros serán índices invertidos, y podrán utilizarse en este tipo de acceso. En la Figura 4.3 se muestra un ejemplo de acceso invertido, correspondiente a la consulta “¿Cuáles son los títulos de los libros cuyo autor es *Dumas*?”.

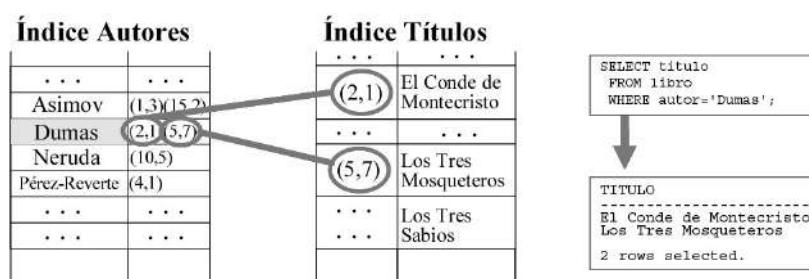


Figura 4.3. Ejemplo de acceso invertido

Los ficheros que cuentan con índices invertidos se denominan ficheros invertidos, siendo un fichero totalmente invertido aquél que tiene índices invertidos para todos sus campos. Hay que señalar que un campo invertido es redundante, ya que su valor puede recuperarse del correspondiente índice invertido, por lo que puede ser obviado al almacenar el registro en el área de

datos. Así, un fichero totalmente invertido puede carecer de área de datos, convirtiéndose así la inversión total en una organización base. Esta organización es eficiente cuando todas las consultas afectan a pocas claves, pero muy peligrosa porque la recomposición del registro (recuperación del registro completo) es una operación muy costosa.

A las entradas de índices secundarios invertidos por listas se les denomina *listas invertidas*. Si las cadenas de punteros son muy grandes, se pueden separar de la clave teniendo así dos ficheros: uno con las cadenas de punteros, y otro con las entradas, que a la sazón constarán de valor para la *CI* y un puntero a su cadena de punteros. De este modo, la recuperación de la entrada (sobre el segundo fichero) es bastante más ágil, reduciéndose el número de accesos totales en muchos casos.

Pero existe otro tipo de índice específico que puede ofrecer buen rendimiento en estos procesos. Son los índices por mapas o *esquemas de bits*, y consisten en una secuencia de bits tan larga como la cardinalidad del dominio sobre el que se define la *CI* (un bit por cada valor del dominio). Cada entrada consistirá en un puntero relativo al registro, y un esquema de bits (con un 1 en los bits correspondientes a valores que verifique el registro, y un 0 en el resto). Para consultar este índice, deberá hacerse un recorrido serial del mismo. No obstante, siempre que la cardinalidad del dominio sea muy reducida, los esquemas serán pequeños y el coste de recorrerlo no será tan grande. Entre las ventajas de esta estructura, se puede indicar que estos índices, al igual que las listas invertidas, permiten la multivaluación de la clave (es decir, varios valores para la clave en el mismo registro). Pero además en este caso se trata de una estructura multiclave, pudiendo concatenar varias claves en el mismo esquema y luego utilizarlas todas o un subconjunto de ellas según requiera la condición de selección. Esto último, sin embargo, no siempre resultará interesante. Por un lado, un índice por esquemas de bits multiclave (por $A \cdot B$) tendrá un tamaño menor que la suma de sendos índices (A y B), y el coste de recorrerlo será menor que el de recorrer los otros dos. Por tanto, en los procesos en los que intervienen ambas claves se tendrá una ganancia. Pero por otro lado, recorrer el índice $A \cdot B$ será en general más costoso que recorrer un índice de una sola clave, y por tanto se perderá eficiencia en los procesos que solo hagan uso de una de las dos claves. Finalmente, disponer los tres índices (A , B y $A \cdot B$) perjudicaría a los procesos de actualización (habría que actualizar los tres índices). De nuevo, la elección dependerá de la frecuencia de uso de las claves.

Una variante del esquema de bits consistiría en suprimir el puntero relativo,

ya que su tamaño es muy grande en relación con el esquema de bits. Para ello, se considerará un tamaño máximo de cubo (de datos) y se dejará reservado en el índice el espacio correspondiente al esquema de bits de cada elemento en el cubo, en la misma posición relativa (respecto al resto de esquemas) que tenga el registro en el área de datos (respecto al resto de registros). De este modo, el puntero relativo se podrá calcular en función de la posición del esquema de bits en el índice. Dado que todos los bits de un esquema pueden adoptar el valor cero (representando el valor nulo en campos opcionales) será necesario añadir un *bit de ocurrencia* al esquema para discriminar los esquemas que representan el valor nulo de aquellos que simplemente no ocurren (el espacio está reservado, pero en esa posición en el área de datos no hay ningún registro). Esta estructura, *esquema de bits de puntero implícito*, suele ser de tamaño muy reducido y, aunque presenta inconvenientes (fijar un tamaño máximo de cubo), suele ofrecer un buen rendimiento. Además, por estar direccionados por su posición, localizar estos esquemas en el segundo paso del acceso invertido es directo, evitando realizar un recorrido serial de todo el archivo índice. Además, gracias al bit de ocurrencia, estas estructuras pueden ser utilizadas como *directorio de ocurrencias* posibilitando mejoras de eficiencia en todos los procesos de recuperación (no solo en los afectados por la *CI*) al evitar el acceso a posiciones vacías.

Para que el acceso invertido sea eficiente, con respecto al acceso indizado, el coste del segundo paso (*traducir* punteros relativos en valores de la clave) tiene que ser menor que el coste de acceder al área de datos. Es decir, la conveniencia de este tipo de acceso deriva de que el número de cubos de índice que deban recorrerse sea inferior al número de cubos de datos que deban recuperarse (en el acceso indizado). Pero incluso cuando esto no es así, debe tenerse en cuenta que estas estructuras son de tamaño razonablemente reducido y que, si la consulta es frecuente, los índices involucrados en la proyección estarán parcial o totalmente contenidos en memoria intermedia, elevando considerablemente la tasa de acierto (*hit ratio*) y reduciendo el coste, por lo que este tipo de acceso es de gran utilidad.

PROBLEMA 4.1: ORGANIZACIONES BASE E INDIZADAS I

Cierto sistema de ficheros de una librería está almacenado sobre un soporte de tamaño de bloque 2 KB y tiempo de acceso a bloque 14 ms. El registro tiene la siguiente descripción:

Tipo	Tamaño		Ocurrencias	
	máximo	medio	máximas	medias
Título	C	50	16,3	
Autor	C	30	15,5	
Año publicación	N	4	4	
Edición	*		10	2,6
(ISBN	C	10	10	
Editorial	C	40	19	
[Colección			1	0,5
(Nombre,	C	40	22	
Tomo	N	2	1,8	
)]				
Fecha 1ª impresión				
(día	N	2	1,7	
mes	N	2	1,25	
año	N	4	4	
)				
Stock	N	2	1,1	
)*				

NOTA: en las casillas de ocurrencias vacías, se asumirá máxima = media = 1.

- a) Realiza un diseño físico del registro lógico con marcas de control y codificación (cada campo deberá optimizarse individualmente). Analiza un archivo de 2.500 registros con organización serial consecutiva para contestar el siguiente cuestionario:

- ¿Cuál es la densidad ideal del registro fijo?
- ¿Cuál es la densidad ideal de tu registro optimizado?
- ¿Cuál es la densidad del fichero (optimizado)?
- Si la tasa de borrado es del 1% semanal (y no existe reutilización

de huecos), calcúlese el tiempo máximo (en semanas) antes de la próxima compactación para evitar que la densidad del archivo descienda por debajo del 80%.

- ¿Cuánto tiempo como máximo costará encontrar el libro “La cruz azul”?

- ¿Cuánto tiempo se emplea de media para encontrar un registro aleatorio?

- ¿Cuál es el coste de sacar el listado “libros cuyo stock sea menor de 3 ud”?

b) Analiza una organización secuencial no consecutiva, con clave de ordenación *Título*. Observa las diferencias (si las hay) con la anterior, y contesta las siguientes preguntas:

- ¿Cuál es la tamaño del fichero? (en KB).

- ¿Cuál es el espacio libre (en bytes) de un cubo completo? (ocupado al 100%).

- ¿Cuál es la densidad de este fichero?

- ¿Y cuánto tiempo como mucho costará encontrar el libro “La cruz azul”?

- Si la tasa de inserción es del 1% semanal (y no existe espacio libre distribuido), calcúlese el tiempo máximo antes de la próxima compactación para evitar que el tiempo máximo de acceso a registro aleatorio supere los 500 milisegundos.

c) Considera ahora una organización direccionada sobre un espacio de direccionamiento $N = 350$, con cubos cuyo espacio es unitario ($E_c = 1$). Se tiene una función de transformación sobre la *CD Título* que arroja una tasa inicial de desbordamientos del 3,4%. Se plantean dos gestiones de desbordamientos: sondeo lineal y área de desbordamiento serial.

Sondeo lineal:

- ¿Cuál es el tamaño del cubo?

- ¿Cuál es la densidad de ocupación del área de datos?

- ¿Cuál es el tamaño del fichero siguiendo la política (i)? (en KB).

- ¿Y cuál es el tiempo de acceso a registro aleatorio (por *CD*) en el mejor caso?
- ¿Y el tiempo de acceso a registro aleatorio (por *CD*) en el peor caso?
- Suponiendo que la tasa de rebotes sea del 0%, ¿cómo queda el dato anterior?

Área de desbordamiento (serial consecutiva):

- ¿Cuál es el tamaño del cubo?
 - ¿Cuál es la densidad de ocupación del área de datos?
 - ¿Cuál es el tamaño del fichero siguiendo la política (ii)? (en KB).
 - ¿Y el tiempo de acceso a registro aleatorio (por *CD*) en el mejor caso?
 - ¿Y el tiempo de acceso a registro aleatorio (por *CD*) en el peor caso?
 - ¿Y cuál es el tiempo medio de acceso a registro aleatorio (por *CD*)?
- d) Se va a realizar un índice para la clave ISBN, manteniendo como organización base la direccionada (sobre $N = 350$, área desbordamiento serial, etc.):
- ¿Cuál es el tamaño del puntero relativo? (en bytes).
 - ¿Cuál es, por tanto, el tamaño de la entrada?
 - Si el índice es serial, ¿cuál es su tamaño? (almacenamiento auxiliar, en KB).
 - Si el índice es serial, ¿cuál es el tiempo máximo de acceso a registro aleatorio a través de esta clave (peor caso)?
 - Si el índice es en árbol B, ¿cuál es el orden de este árbol?
 - ¿Cuál es el tamaño del almacenamiento auxiliar en este caso? (en KB).
 - ¿Y cuál es el tiempo máximo de acceso a registro aleatorio a través de este índice en árbol B (peor caso)?

SOLUCIÓN PROPUESTA

Apartado A: la primera cuestión a resolver es la densidad del registro fijo. La información útil del registro son 163,07 bytes ($16,3 + 15,5 + 4 + 2,6 * (10 + 19 + 0,5 * (22 + 1,8) + 1,7 + 1,25 + 4 + 1,1)$). Su volumen real eran 1.104 bytes ($50 + 30 + 4 + 10 * (10 + 40 + 40 + 2 + 2 + 2 + 4 + 2)$), por lo que la densidad ideal de ese registro es $d_i = 163/1.104 = 14,77\%$. Dado que es bastante baja, es conveniente plantear un diseño físico-lógico optimizado, como el siguiente:

Especificación del campo		Bytes útiles	Bytes reales
longitud_título	B(1)		1
Título	B(longitud_título)	16,3	16,3
longitud_autor	B(1)		1
Autor	B(longitud_autor)	15,5	15,5
Año_publicación	B(2)	2	2
veces_edición	B(1)		1
(ISBN	B(10)	2,6*10	2,6*10
long_editorial	B(1)		2,6*1
Editorial	B(long_editorial)	2,6*19	2,6*19
existe_colección	B(1)		2,6*1
(longitud_nombre	B(1)		2,6*0,5*1
Nombre	B(longitud_nombre)	2,6*0,5*22	2,6*0,5*22
Tomo	B(1)	2,6*0,5*1	2,6*0,5*1
) existe_colección			
día	B(1)	2,6*1	2,6*1
mes	B(1)	2,6*1	2,6*1
año	B(2)	2,6*2	2,6*2
Stock	B(1)	2,6*1	2,6*1
) veces_edición			

El siguiente punto consiste en calcular su densidad ideal, o relación entre información útil (152,1 bytes) y volumen real (161,6 bytes). La densidad ideal resultante es $d_i = 94,12\%$, bastante superior a la anterior. Para el cálculo de la densidad del archivo serial consecutivo que contiene 2500 registros de este tipo, se necesitará averiguar el volumen del archivo (cantidad de bloques que requiere para almacenarse): $T_{\text{archivo}} = \lceil 2.500 * 161,6 / 2.048 \rceil = 198$ bloques. Ahora la densidad del archivo se expresa como $d_r = 2.500 * 152,1 / 198 * 2.048 = 93,77\%$. Como se puede comprobar, la densidad real siempre será inferior (o a lo sumo igual) que la densidad ideal, sea cual fuere el diseño físico. En este caso, la diferencia es mínima, ya que la organización serial consecutiva optimiza la

densidad del archivo.

No obstante, esa densidad puede bajar mucho con los huecos que puedan producirse por borrados (o modificaciones). En este caso, se suprime un 1% de los registros iniciales (25 registros) cada semana, que reducen la cantidad de información útil y no afectan al volumen real. Se establece que en el momento de la compactación (dentro de n semanas) la densidad no deberá haber bajado del 80%, por lo que se puede plantear la siguiente inecuación: $d_r = (2.500 - n*25)*152,1 / 198*2.048 \geq 80\% \rightarrow n \leq 14,69$ semanas. Luego se reorganizará dentro de **14** semanas para evitar que su densidad baje del 80%.

Volviendo a la situación inicial del archivo, el tiempo máximo de acceso a registro aleatorio (P_1) viene determinado por el recorrido serial a la totalidad, y por tanto, por el número de bloques. Así, $t_{\max}(P_1) = 198 \text{ bq} * 14 \text{ ms/bq} = \mathbf{2.772 \text{ ms}}$. Por otro lado, el tiempo medio de acceso a registro aleatorio se calculará como el acceso a la mediana, y de este modo $t_{\text{med}}(P_1) = 198 + 1 / 2 * 14 = \mathbf{1.393 \text{ ms}}$. Los procesos de selección sobre clave no identificativa (P_2) precisan un recorrido serial completo del archivo, y por tanto $t_{\max}(P_2) = t_{\min}(P_2) = t_{\max}(P_1) = \mathbf{2.772 \text{ ms}}$.

A continuación se recopilan las preguntas correspondientes a este primer apartado, con sus correspondientes respuestas:

• ¿Cuál es la densidad ideal del registro fijo?	14,77%
• ¿Cuál es la densidad ideal de tu registro optimizado?	94,12%
• ¿Cuál es la densidad del fichero (optimizado)?	93,77%
• Si la tasa de borrado es del 1% semanal (y no existe reutilización de huecos), calcúlese el tiempo máximo (en semanas) antes de la próxima compactación para evitar que la densidad del archivo descienda por debajo del 80%.	14 semanas
• ¿Cuánto tiempo como máximo costará encontrar el libro “La cruz azul”?	2.772 ms
• ¿Cuánto tiempo se emplea de media para encontrar un registro aleatorio?	1.393 ms
• ¿Cuál es el coste del listado “libros cuyo stock sea menor de 3 ud”?	2.772 ms

Apartado B: al ser la nueva organización no consecutiva (O_2), se tendrá que definir un cubo (se tomará de espacio un bloque). El tamaño de cubo determina el número de registros lógicos que caben (de media) en cada cubo, que en este caso será $T_c = \lfloor 2.048 \text{ B/cubo} / 161,6 \text{ B/reg} \rfloor = 12 \text{ reg/cubo}$. A partir de este dato se podrá averiguar el número de cubos necesarios para almacenar 2.500 registros: $T_{\text{archivo}} = \lceil 2500 \text{ reg} / 12 \text{ reg/cubo} \rceil = 209 \text{ cubos}$. Teniendo en cuenta que cada cubo ocupa 2 KB, el tamaño del fichero será **418 KB**.

En cada cubo quedará cierta cantidad de espacio libre. De media, 12 registros ocupan $161,6 \text{ B} * 12 = 1.939,2$ y en cada cubo hay disponibles B. Por tanto, la cantidad de espacio libre es $2.048 - 1.939,2 = 108,8 \text{ bytes}$. Este gasto de espacio influirá, lógicamente, en la densidad real del archivo, porque la cantidad de información útil se mantiene constante pero el volumen aumenta al cambiar la organización: $d_r = 2.500 * 152,1 / 209 * 2.048 = 88,84\%$. Sin embargo, al estar ordenado por *Título* se puede aplicar búsqueda dicotómica reduciendo el coste máximo del proceso: $t_{\max}(P_1) = \lceil \log_2 (198 + 1) \rceil \text{ accesos} * 14 \text{ ms/acceso} = 112 \text{ ms}$.

Aunque es un buen resultado, esta organización degenera con el proceso de inserción de registros (creando un área desordenada, que deberá recorrerse

serialmente en caso de fallar la búsqueda dicotómica). Según el enunciado, el tiempo máximo de acceso a registro aleatorio no debe superar los 500 ms, y dado que la búsqueda dicotómica consume 112 ms, el tiempo máximo para la búsqueda serial será de $500 - 112 = 388$ ms. Por lo tanto, los bloques leídos en el peor caso serán menos de $388 \text{ ms} / 14 \text{ ms/bq} = 27,7$ bloques (a lo sumo 27 bloques). En esos bloques van a caber $27 * 2.048 / 161,6 = 342,18$ registros (al insertar 343 registros degeneraría por debajo del margen establecido). Dado que cada semana se inserta un 1% del contenido inicial (25 registros), y que en el momento de la reorganización habrán transcurrido n semanas, se puede plantear la siguiente inecuación: $25 * n \leq 342$, de donde se extrae que $n \leq 13,68$ semanas. Por lo tanto, para mantener ese rendimiento habrá que reorganizar en **13** semanas.

Las cuestiones planteadas para este segundo apartado, debidamente resueltas, se muestran a continuación a modo de resumen:

- ¿Cuál es la tamaño del fichero? (en KB). 418 KB
- ¿Cuál es el espacio libre de un cubo completo (ocupado al 100%)? 108,8 B
- ¿Cuál es la densidad de este fichero? 88,84%
- ¿Y cuánto tiempo como mucho costará encontrar el libro *La cruz azul*? 112 ms
- Si la tasa de inserción es del 1% semanal (y no existe espacio libre distribuido), calcúlese el tiempo máximo antes de la próxima compactación para evitar que el tiempo máximo de acceso a registro aleatorio supere los 500 milisegundos. 13 semanas

Apartado C: al cambiar a la organización direccional descrita en el enunciado, desbordarán el 3,4% de los registros, es decir 85 registros. En este apartado se pide comparar dos gestiones de desbordamiento y contestar algunas preguntas, lo que se hará seguidamente.

Gestión por sondeo lineal (O_3): el tamaño del cubo será independiente de la gestión de desbordamientos y, dado que espacio de cubo y tamaño del registro son los mismos que en el apartado (B), el tamaño buscado es el calculado para el caso secuencial ($T_c = 12$). Como todos los registros se van a almacenar en el área de datos (incluso los desbordados, ya que la gestión es de saturación), se tendrán

2.500 registros donde caben 4.200 registros (350 cubos que pueden contener 12 registros cada uno). Así, $d_{oc} = 2.500/4.200 = 59,52\%$.

El tamaño del fichero viene dado directamente por el número de cubos del área de datos (puesto que no tiene más bloques que esos). En este caso en particular, $350 \text{ bq} * 2 \text{ KB/bq} = 700 \text{ KB}$. En el mejor caso, un registro aleatorio sería encontrado en el primer acceso (**14 ms**), que para esta organización es algo bastante probable (el 96,6% de los casos). Sin embargo, en el peor caso se habría formado un cúmulo que contuviera todos los registros del archivo (el cúmulo tendría $\lceil 2.500/12 \rceil = 209$ cubos), y el registro buscado estaría al final (en el último cubo del cúmulo): $t_{\max}(P_1) = 209 \text{ acc} * 14 \text{ ms/acc} = \mathbf{2.926 \text{ ms}}$. Si no existen rebotes (su tasa es del 0%), implica que todo registro desbordado encuentra espacio para almacenarse en el cubo siguiente, y por tanto en el peor caso solo se realizarían dos accesos (**28 ms**). Aunque no se pide en el enunciado, el coste medio de acceso se puede calcular como $C(O_3, P_1) = (2.415*1 + 85*2) / 2.500 = 1,034$ accesos, y por tanto el tiempo medio de acceso a registro aleatorio quedaría $t_{\text{med}}(P_1) = 14,48 \text{ ms}$.

Gestión por área de desbordamiento serial consecutiva (O_4): como se ha dicho anteriormente, el tamaño del cubo será independiente de la gestión de desbordamientos, y por tanto igual al del apartado anterior ($T_c = 12$). No así la densidad de ocupación del área de datos, que en este caso contiene 2.500 - 85 registros (dado que los desbordamientos se almacenan aparte). La densidad de ocupación será ligeramente menor: $d_{oc} = 2.415/4.200 = 57,5\%$.

El tamaño del fichero será algo mayor, dado que a los bloques necesarios para almacenar los cubos de datos habrá que sumarle los utilizados para el área de desbordamiento. En este caso, $T_{\text{área_desbordamiento}} = \lceil 85*161,6/2.048 \rceil = 7$ bloques. En total, serán 357 bloques, y por tanto el tamaño del fichero es de **714 KB**. En el mejor caso, igual que antes, un registro aleatorio se recupera en el primer acceso (14 ms). En el peor caso, tras acceder al cubo correspondiente en el área de datos, se recorrería toda el área de desbordamiento para encontrar el registro (7 acc. más). Por tanto, el tiempo máximo de este proceso sería $14 \text{ ms/acc} * 8 \text{ acc} = \mathbf{120 \text{ ms}}$. Finalmente, el tiempo medio de P_1 se podrá calcular tras averiguar su coste, teniendo en cuenta que el número medio de accesos para obtener un registro desbordado es un acceso (al área de datos) más el acceso a la mediana del área serial ($(7 + 1) / 2 = 4$ accesos). Así, $C(O_4, P_1) = (2.415 * 1 + 85*5) / 2.500 = 1,136$ accesos, y el tiempo medio de acceso a registro aleatorio

es $t_{\text{med}}(P_1) = \mathbf{15,9 \text{ ms}}$.

Igual que en los apartados anteriores, las respuestas a las cuestiones planteadas para este apartado se resumen a continuación:

Sondeo lineal

- ¿Cuál es el tamaño del cubo? 12 reg/cubo
- ¿Cuál es la densidad de ocupación del área de datos? 59,52%
- ¿Cuál es el tamaño del fichero siguiendo la política (i)?
(en KB). 700 KB
- ¿Y cuál es el tiempo de acceso a registro aleatorio
(por CD) en el mejor caso? 14 ms
- ¿Y el tiempo de acceso a registro aleatorio (por CD) en
el peor caso? 2.926 ms
- Suponiendo que la tasa de rebotes sea del 0%, ¿cómo
quedá el dato anterior? 28 ms

Área de desbordamiento (serial consecutiva)

- | | |
|--|-------------|
| • ¿Cuál es el tamaño del cubo? | 12 reg/cubo |
| • ¿Cuál es la densidad de ocupación del área de datos? | 57,5% |
| • ¿Cuál es el tamaño del fichero siguiendo la política (ii)? (en KB). | 714 KB |
| • ¿Y el tiempo de acceso a registro aleatorio (por CD) en el mejor caso? | 14 ms |
| • ¿Y el tiempo de acceso a registro aleatorio (por CD) en el peor caso? | 120 ms |
| • ¿Y cuál es el tiempo medio de acceso a registro aleatorio (por CD)? | 15,9 ms |

Apartado D: para este índice, el puntero externo (a un cubo de datos) tiene que referir más de 256 direcciones, y por lo tanto serán necesarios al menos **2 bytes**. Como la clave es de tamaño fijo (10 bytes), la entrada en total precisará **12 bytes**. Al ser un índice denso, tendrá 2.500 entradas, que en el caso del índice serial ocuparían $\lceil 2.500 * 12 / 2.048 \rceil = 15$ bq. Luego el tamaño auxiliar, para ese caso, será de **30 KB**. Dado que el índice es primario (la CI es clave identificativa), encontrar una clave costará en el peor caso 15 accesos, y de media $(15 + 1) / 2 = 8$ accesos. El tiempo máximo de acceso a registro aleatorio a través de la clave ISBN (P_3) se calcula a partir de esos 15 accesos y sumándole otro más al cubo de datos, y queda $t_{\max}(P_3) = 224$ ms.

Al organizarlo como árbol B, aparecerán punteros internos (a un cubo de índice), y aunque el tamaño de este índice será más del doble que el serial, el reducido tamaño de éste (15 cubos) conduce a la conclusión de que con 1 byte será suficiente para el puntero interno (si después se viera que es insuficiente, deberán repetirse los cálculos cambiando este supuesto). El primer paso para analizar este índice será calcular el orden (m) del árbol: $m * 1 + k * 12 \leq 2.048$, y además $m = k + 1$, por lo que $m \leq 158,46$. En conclusión, el orden $m = 158$, y la ocupación máxima será $k = 157$. Ahora se calculará la ocupación mínima $k_{\min} = \lfloor 157/2 \rfloor = 78$ (y, por tanto, $m_{\min} = 79$). Se podrá así obtener una cota superior a su tamaño: $T_{\text{índice}} \leq \lfloor (e-1)/k_{\min} \rfloor + 1 = 33$ bloques. Expresado en KB, $T_{\text{índice}} \leq 66$ KB.

Para calcular el coste asociado al uso de un índice estructurado como este árbol, en primer lugar deberá averiguar la profundidad:

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	< 2.500
2	2	2*78	157	< 2.500
3	2*79	158*78	12.324	> 2.500

En el peor caso, tendrá dos niveles de profundidad. Con un tercer acceso se recuperaría el cubo de datos, y así el tiempo máximo de acceso a través del índice será $t_{\max}(P_3) = 42 \text{ ms}$. Si se contara con memoria intermedia para bloquear la raíz, solo sería necesario un acceso al índice y otro al cubo de datos, y entonces el tiempo máximo de acceso sería de 28 ms.

Para finalizar este ejercicio, se ofrece el resumen de las preguntas de este último apartado debidamente cumplimentadas.

- ¿Cuál es el tamaño del puntero relativo (en bytes)? 2 bytes
- ¿Cuál es, por tanto, el tamaño de la entrada? 12 bytes
- Si el índice es serial, ¿cuál es su tamaño (almacenamiento auxiliar, KB)? 30 KB
- Si el índice es serial, ¿cuál es el tiempo máximo de acceso a registro aleatorio través de esta clave (peor caso)? 224 ms
- Si el índice es en árbol B, ¿cuál es el orden de este árbol? 158
- ¿Cuál es el tamaño del almacenamiento auxiliar en este caso (en KB)? 66 KB
- ¿Y cuál es el tiempo máximo de acceso a registro aleatorio través de este índice en árbol B (peor caso)? 42 ms

PROBLEMA 4.2: ORGANIZACIONES BASE E INDIZADAS II

Se tiene un archivo cuya organización base (O_0) es secuencial no consecutiva (ordenado por K_1 , que es clave de identificación, con espacio de cubo igual a un bloque). Este archivo contiene 20.000 registros de 250 bytes de media cada uno, y se almacena en un soporte cuyo bloque tiene 2 Kbytes de tamaño y tiempo de acceso 9 ms.

A) En primer lugar, se requiere hallar los tiempos máximo y medio para recuperar un registro aleatorio a través de la clave de ordenación (P_1), y para hacerlo a través de otra clave de identificación (P_2). También se quiere averiguar el tamaño del archivo.

B) Al ver que está sometido principalmente a un proceso (P_3) de recuperación de todos los registros que cumplan con una condición sobre una clave no identificativa K_3 (tamaño fijo, 30 bytes) que presenta tan solo 500 valores distintos (repartidos uniformemente entre todos los registros). Para esta clave se dispone de una función de transformación sobre $N = 500$ cercana a la ideal (prácticamente uniforme). Se plantean las siguientes opciones:

O_1 : misma org. base (O_0) añadiendo un índice por listas ordenadas para K_3 .

O_2 : misma org. base (O_0) incorporando un índice por esquemas de bits para K_3 .

O_3 : organización direccional dispersa ($CD = K_3$; $E_c = 1$ bq; gestión de desbordamientos en área serial independiente).

O_4 : organización direccional dispersa ($CD = K_3$; $E_c = 1$ bq; gestión de desbordamientos en área independiente encadenada a cubo).

O_5 : org. secuencial no consecutiva con K_3 como clave física de ordenación.

Se pide escoger de entre estas opciones la que optimiza la eficiencia en tiempo (medio) del proceso de recuperación través de la clave K_3 . Coméntese qué ventajas aportarían las soluciones O_1 y O_2 sobre el resto.

- C) En otro momento de la vida de este archivo, se comprueba que es sometido a dos procesos: uno de consulta a la totalidad no ordenado (P_4) cuya frecuencia relativa es f_1 , y otro de selección a través de clave identificativa K_4 (P_5) cuya frecuencia relativa es f_2 . Con el objeto de optimizar el rendimiento en tiempo de acceso, se analizan dos posibles organizaciones base: serial consecutiva (O_6) y direccional dispersa (O_7) con $CD = K_4$ (para esta última, se tiene una dispersión cercana a la uniforme sobre $N = 4096$). Tras este análisis, se ha decidido escoger la organización serial. Se pide calcular la relación entre las frecuencias (f_1 y f_2) que ha soportado esta decisión.
- D) Para mejorar el coste del proceso P_5 se incorpora un índice en árbol B sobre la $CI = K_4$ (que tiene tamaño fijo igual a 15 caracteres). Suponiendo la disponibilidad de una página de memoria intermedia bloqueada para el índice, se pide averiguar el nuevo tiempo máximo de acceso para el proceso P_5 y el tamaño máximo del almacenamiento auxiliar.

SOLUCIÓN PROPUESTA

Apartado A: la organización base correspondiente a este apartado (O_0) es ‘secuencial no consecutiva por K_1 ’. Dado que se trata de un archivo no consecutivo, el primer paso será averiguar el *tamaño de cubo*, que será $T_c = \lceil 2.048 / 250 \rceil = 8$.

A partir del tamaño de cubo se podrá averiguar el tamaño del archivo: $T = (20.000 \text{ reg} / 8 \text{ reg/cubo}) * 2 \text{ KB/cubo} = \mathbf{5.000 \text{ KB}}$.

En el mejor caso, el número de cubos que deben leerse para localizar un registro será uno solamente (se encontraría el registro en el primer cubo recuperado). Sin embargo, la magnitud más significativa es la correspondiente al peor caso (el coste máximo).

$$C_{\min}(O_0, P_1) = C_{\min}(O_0, P_2) = 1 \text{ acceso} * 9 \text{ ms/acceso} = \mathbf{9 \text{ ms}}$$

$$C_{\max}(O_0, P_1) = \lceil \log_2(2.500+1) \rceil \text{ accesos} * 9 \text{ ms/acceso} = \mathbf{117 \text{ ms}}$$

$$C_{\max}(O_0, P_2) = 2.500 \text{ accesos} * 9 \text{ ms/acceso} = \mathbf{22,5 \text{ s}}$$

Apartado B: para analizar el coste de las organizaciones O_1 y O_2 hay que sumar los accesos sobre las organizaciones auxiliares a los que deben realizarse sobre los cubos de datos. Primero, por tanto, debe calcularse el tamaño de cada una de ellas. Los punteros relativos que utilizarán son de 2 bytes ($\lceil (\log_2 5.000)/8 \rceil = 2$) para la parte alta, y sin parte baja (dado que no se van a utilizar estos índices para realizar accesos invertidos).

En el caso de las listas de punteros, el índice constará de 500 entradas (una por valor en el dominio de la *CI*, es decir, de K_3). Cada una de las entradas estará formada de valor de la *CI* (30 bytes), marca de longitud de la lista de punteros, y lista de punteros. De media, cada lista tendrá 40 reg/ent (20.000 reg/500 ent). Dado que los registros se reparten uniformemente habrá poca variabilidad en el tamaño de la lista, su marca de longitud será de un byte (que soportará listas de hasta 255 punteros). El tamaño de la entrada será:

$$T_{\text{entrada}} = \text{clave} + \text{contador} + \text{punteros} = 30 + 1 + 2 * (40) = 111 \text{ bytes/lista.}$$

El índice es no consecutivo y en cada cubo cabrán 18 listas ($\lfloor 2.048 \text{ B/cubo}$

/ 111 B/lista]). Se calcula el tamaño del índice como $T_{índice}(O_1) = \lceil 500 \text{ listas} / 18 \text{ listas/bloque} \rceil = 28 \text{ bq}$. Para el proceso P_3 , se deberá recuperar una entrada mediante búsqueda dicotómica, y los 40 registros coincidentes de media (en general, se supondrán almacenados en cubos distintos): $C(O_1, P_3) = \text{acc_índice} + \text{acc_datos} = \lceil \log_2(28+1) \rceil \text{ acc} + 40 \text{ acc} = \mathbf{45 \text{ accesos}}$.

Los esquemas de bits serán demasiado grandes porque la cardinalidad del dominio es alta. En primer lugar se halla el tamaño de la entrada y, a partir de éste, el del archivo índice. Así, $T_{\text{entrada}} = \text{esquema} + \text{puntero} = \lceil 500/8 \rceil + 2 = 65 \text{ B/ent}$. Este índice ocupa $T_{índice}(O_2) = \lceil 20.000 \text{ ent} * 65 \text{ B/ent} / 2.048 \text{ B/bq} \rceil = 635 \text{ bloques}$. Dado que el uso de este índice implica su recorrido serial completo y además recuperar los registros apuntados (que por generalidad y siguiendo el peor caso, se supondrán almacenados en cubos de datos distintos), se calcula el coste como $C(O_2, P_3) = \text{acc_índice} + \text{acc_datos} = 635 \text{ acc} + 40 \text{ acc} = \mathbf{675 \text{ accesos}}$.

Para la organización O_3 se tiene que la dispersión es uniforme sobre 500 direcciones, de donde se sigue que a cada dirección le corresponden 40 registros (20.000 reg / 500 dir). Del apartado anterior se sabe que $T_c = 8$, de donde se sigue que toda dirección desborda en 32 registros (40 - 8). Los desbordamientos totales son $500 \text{ dir} * 32 \text{ reg/dir} = 16.000 \text{ regs}$, que con O_3 se van a gestionar en área serial (consecutiva). Su tamaño será $T_{\text{área_desb}} = 16.000 \text{ reg} * 250 \text{ B/reg} = 4*10^6 \text{ B}$; y expresado en bloques $4*10^6 \text{ B} / 2.048 \text{ B/bq} = 1.954 \text{ bloques}$. El coste del proceso P_3 , en este caso, implica un acceso al área de datos y el recorrido completo del área de desbordamiento: $C(O_3, P_3) = (1 + 1.954) \text{ accesos} = \mathbf{1.955 \text{ accesos}}$.

Para mejorar el rendimiento de ese direccionamiento deberá considerarse un espacio de cubo mayor (procurando que quepan al menos 40 registros por cubo), o cambiar la gestión de desbordamientos. En esta dirección, la cuarta organización propone una gestión encadenada a cubo, de modo que los 32 registros desbordados (de cada dirección) se van a almacenar en $\lceil 32/8 \rceil = 4$ cubos encadenados (en área de desbordamiento, cuyo tamaño es $500 * 4 = 2.000$ cubos). El proceso P_3 implicará recuperar un cubo del área de datos y cuatro encadenados (del área de desbordamiento). Por tanto: $C(O_4, P_3) = 1 \text{ acc} + 4 \text{ acc} = \mathbf{5 \text{ accesos}}$.

Finalmente, si se organizara secuencialmente por K_3 , se tendrían que sumar

los accesos para recuperar un cubo que contenga al menos un registro que encaja con el criterio de búsqueda, con los accesos necesarios para recuperar el resto de registros (búsqueda dicotómica extendida). Como la cardinalidad del dominio es inferior al número de cubos, la primera parte será una búsqueda dicotómica sobre el número de valores del dominio; y como O_5 es no consecutiva, la extensión se calcula por registros: $C(O_5, P_3) = \lceil \log_2 (500 + 1) \rceil \text{ acc} + \lceil (40 + 1)/8 \rceil \text{ acc} = 15 \text{ acc}$.

En resumen, la organización que optimiza el coste de P_3 es O_4 . Las organizaciones O_1 y O_2 tienen la ventaja fundamental de ser organizaciones auxiliares, es decir, no forzarían un cambio de la organización base (mantendrían la organización base anterior, con los correspondientes beneficios sobre otros procesos). En todo caso, de ellas se escogería O_1 , que es la que mejor rendimiento ofrece (debido a la elevada cardinalidad del dominio sobre el que se define K_3).

Apartado C: dado que la organización serial (O_6) ofrece mejor rendimiento que la direccionada (O_7), se puede establecer que $C(O_6, \{P_4, P_5\}) \leq C(O_7, \{P_4, P_5\})$. Habrá que calcular cada una de ellas en función de las frecuencias de los procesos.

En primer lugar, de la organización serial se precisa conocer su tamaño, que será: $T_{\text{archivo}}(O_6) = \lceil 20.000 \text{ reg} * 250 \text{ B/reg} / 2.048 \text{ B/bloque} \rceil = 2.442 \text{ bloques}$. El proceso P_4 se hará con un recorrido a la totalidad y el P_5 implica el acceso a la mediana. Por tanto:

$$C(O_6, \{P_4, P_5\}) = f_1 * C(O_6, P_4) + f_2 * C(O_6, P_5) = f_1 * 2.442 \text{ acc} + f_2 * (2.442+1)/2 \text{ acc}$$

Por otro lado, en la organización O_7 el proceso P_4 también requiere un recorrido a la totalidad (aunque ahora el área de datos tiene 4096 cubos), y el P_5 implica un solo acceso:

$$C(O_7, \{P_4, P_5\}) = f_1 * C(O_7, P_4) + f_2 * C(O_7, P_5) = f_1 * 4.096 \text{ accesos} + f_2 * 1 \text{ acceso}$$

Finalmente, se plantea la relación entre las frecuencias:

$$2.442 * f_1 + 1.221,5 * f_2 \leq 4.096 * f_1 + f_2 \rightarrow 1.220,5 * f_2 \leq 1.854 * f_1 \rightarrow f_2 \leq 1,52 * f_1$$

Y si solo se tienen esos dos procesos ($f_1 + f_2 = 100\%$), se podrán despejar las cotas para esas frecuencias relativas: $f_1 \geq 39,7\%$, $f_2 \leq 60,3\%$.

Apartado D: este apartado plantea una organización serial con índice en árbol B sobre $CI = K_5 (O_6)$. Se necesita un puntero externo capaz de diferenciar 2.442 direcciones, es decir, $T_{ptr_externo} = \lceil (\log_2 2.442) \text{ bits} / 8 \text{ bits/B} \rceil = 2 \text{ bytes}$. Por otro lado, el tamaño del puntero interno depende del número de cubos del índice (máximo previsto). Si no se proporciona como dato, deberá decidirse el diseño de puntero más adecuado (de menor tamaño posible con la suficiente potencia de direccionamiento). En este caso, un índice serial de 20.000 entradas de 17 bytes ocupa 167 bloques, y un árbol B tendrá al menos el doble en el peor caso (por tanto, 1 byte no sería suficiente). Se tomará un puntero interno de tamaño 2 bytes (se deja al lector el ejercicio de calcular el orden y el tamaño del índice tomando solo 1 byte de puntero interno, demostrando así que es insuficiente). A partir de estas premisas, el primer paso es realizar el cálculo del orden:

Se plantea la inecuación: $m * 2 \text{ bytes} + k * (15 + 2) \text{ bytes} \leq 2.048 \text{ bytes}$. Por otro lado, se tiene que $m = k + 1$, y se despeja $19 * m \leq 2.065 \rightarrow m \leq 108,7$. Por tanto, el orden (m) será 108, y la ocupación máxima (k) será 107.

Finalmente, la ocupación mínima es el cincuenta por ciento, y así se tendrá que: $k_{min} = \lfloor 107/2 \rfloor = 53$ y $m_{min} = 54$.

Con estos datos, ya se puede calcular una cota para el tamaño del índice: $T_{índiceB} \leq \lceil \text{entradas} / k_{min} \rceil$, o precisando $T_{índiceB} \leq \lceil 19.999/53 \rceil + 1$ bloques. Si cada bloque son 2 KB, se tendrá que el tamaño máximo del índice es de 756 KB.

Para calcular el coste de P_5 , deberá averiguararse la profundidad del árbol:

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	< 20.000
2	2	2*53	107	< 20.000
3	2*54	108*53	5.831	< 20.000
4	5.832	309.096	314.927	≥ 20.000

El árbol tendrá tres niveles, y el coste máximo de obtener una entrada será, por tanto, 3 accesos, descontando el acceso al nodo raíz que se procurará mantener bloqueado en memoria intermedia. El proceso P_5 se completará con el

acceso al cubo de datos: $C_{\max}(O_6, P_5) = 3$ accesos – 1 acceso + 1 acceso = 3 accesos.

Como el tiempo de acceder a un bloque es 9 ms, el tiempo máximo necesario para resolver el proceso P_5 será igual a 27 ms.

PROBLEMA 4.3: COMPARACIÓN DE ORGANIZACIONES

El almacenaje de una consola portátil de videojuegos tiene un bloque de tamaño 1 KB, y se supone que se dispone de memoria intermedia para los usos básicos. Existe un juego de rol que tiene almacenados cerca de 2.000 personajes, cuya ocupación útil es de 189,5 caracteres, siendo su volumen medio de 211 bytes. El conjunto de procesos a los que se somete el archivo se define como $P = \{P_1, P_2, P_3, P_4\}$, equiprobables (25% cada uno) y con la siguiente descripción:

P_1 : Selección de un registro por clave de identificación (K_1).

P_2 : Selección de un registro por clave de identificación (K_2).

P_3 : Selección de todos los registros (de media 10) por clave de búsqueda no identificativa (K_3 , que presenta 200 valores distintos).

P_4 : Listado de todos los registros.

Considérese que se dispone de un algoritmo de transformación uniforme por $f_1: K_1 \rightarrow N_1$ (con $N_1 = 640$), y otro por $f_3: K_3 \rightarrow N_3$ (con $N_3 = 250$), y que la naturaleza de K_1 es alfanumérica. Con la información disponible, compárense al menos cinco organizaciones base (especificando convenientemente los parámetros) y elíjase la más eficiente en tiempo (menor coste global).

SOLUCIÓN PROPUESTA

Para cubrir el mayor espectro posible de las soluciones aportadas por el lector, aunque solo se piden cinco soluciones se van a analizar algunas más. Dado que el objetivo es comparar organizaciones, se expresarán los costes en número de accesos (el tiempo de acceso sería el resultante de multiplicar esas cantidades por el tiempo de acceso a bloque que, en este caso, no se proporciona, pero no es necesario para comparar organizaciones).

La primera a analizar será la organización **serial consecutiva** (O_1). El primer paso consistirá en averiguar su tamaño (el número de bloques del archivo serial). Se tendrán:

$$T_{\text{archivo}} = 211 \text{ bytes/registro} * 2.000 \text{ registros} = 422.000 \text{ bytes}$$

$$T_{\text{fichero}} = \lceil 422.000 \text{ bytes} / 1.024 \text{ bytes/bq} \rceil = \lceil 412,1 \rceil \text{ bloques} \rightarrow T_{\text{fichero}} = 413 \text{ bloques}$$

Ahora se calculará el coste medio de cada proceso P_i , para después hallar el coste global como la media ponderada de todos ellos (al ser equiprobables, se calculará la media aritmética). Los dos primeros son un acceso a la mediana (por ser K_1 y K_2 claves de identificación); el tercer proceso selectivo es por clave no identificativa y habrá que recorrer todo el archivo; finalmente, el cuarto proceso es una recuperación de la totalidad (también habrá que recorrer todo el archivo).

$$C(O_1, P_1) = C(O_1, P_2) = (413 + 1) / 2 = 207 \text{ accesos}$$

$$C(O_1, P_3) = C(O_1, P_4) = 413 \text{ accesos}$$

$$\mathbf{C(O_1, P) = \sum_{i=1..4} C(O_1, P_i) / 4 = 310 \text{ accesos}}$$

En segundo lugar, podría plantearse una organización **serial no consecutiva** (O_2), que ofrece ventajas sobre operaciones de modificación, gestión de huecos (en borrado e inserciones), y accesos indizados. Como contrapartida, tiene el inconveniente de presentar un área de datos de mayor tamaño (aumenta el número de bloques y, por ende, de accesos en los recorridos seriales). En este caso, no se aprovechará ninguna de esas ventajas, pero sí afectará negativamente al rendimiento el crecimiento del tamaño del archivo. Por lo tanto, no se va a comprobar este rendimiento: se sabe a priori que es peor.

Sí se calculará el rendimiento de la organización **O_3 secuencial consecutiva sobre K_1** , que ofrece alternativas de acceso a través de esta clave. En este caso, se precisa un mecanismo para localizar el desplazamiento del primer registro lógico del bloque (ya que habitualmente los primeros bytes corresponden a un registro “partido” entre dos bloques). Como el tamaño de bloque es 1.024 bytes, se necesitarán 2 bytes por bloque para señalar el comienzo del primer registro lógico ($\lceil (\log_2 1.024)/8 \rceil = 2$). También habrá que recalcular el volumen del archivo, ya que ahora solo se dispondrán de 1.022 bytes/bloque para datos.

$$T_{\text{archivo}} = 211 \text{ bytes/registro} * 2.000 \text{ registros} = 422.000 \text{ bytes}$$

$$T_{\text{fichero}} = \lceil 422.000 \text{ bytes} / 1.022 \text{ bytes/bq} \rceil = \lceil 412,9 \rceil \text{ bq} \rightarrow T_{\text{fichero}} = 413 \text{ bloques}$$

Ahora se procederá como antes para hallar el coste global. En este caso, el primer proceso realizará una búsqueda dicotómica, y los otros tres procesos se resolverán mediante mecanismos seriales.

$$C(O_3, P_1) = \log_2 (413 + 1) = 9 \text{ accesos}$$

$$C(O_3, P_2) = (413 + 1) / 2 = 207 \text{ accesos}$$

$$C(O_3, P_3) = C(O_3, P_4) = 413 \text{ accesos}$$

$$\mathbf{C(O_3, P) = 260,5 \text{ accesos}}$$

También podrá calcularse el rendimiento de la organización **secuencial consecutiva sobre K_3 (O_4)**, con las mismas condiciones que O_3 y dos variantes: en este caso, P_1 se hará de modo serial (equivalente a P_2); y P_3 aplicará la búsqueda dicotómica extendida. En esta búsqueda, un primer paso localizará un registro con el valor dado para K_3 , de entre 200 zonas (200 valores distintos). Después, buscará hacia atrás y hacia delante hasta producir sendos fallos (12 registros en total).

Esos 12 registros ocupan $211 \text{ bytes/reg} * 12 \text{ registros} = 2.532 \text{ bytes}$. Al ser consecutivos, el peor caso consistiría en emplear un acceso para leer un solo byte del primer registro leído (de los 12 totales). El resto de bytes necesarios ocuparán $2.531 \text{ B} / 1.022 \text{ B/bq} = 3 \text{ bloques}$ (que requieren 3 accesos). A estos 4 accesos hay que restarle 1, pues la búsqueda dicotómica termina leyendo uno de ellos.

$$C(O_4, P_1) = C(O_4, P_2) = (413 + 1) / 2 = 207 \text{ accesos}$$

$$C(O_4, P_3) = \log_2 (413 + 1) + 3 = 12 \text{ accesos}$$

$$C(O_4, P_4) = 413 \text{ accesos}$$

C(O₄,P) = 209,75 accesos

Ahora podrían plantearse las organizaciones **secuencial no consecutiva sobre K₁(O₅)** y **secuencial no consecutiva sobre K₃(O₆)**, pero las ventajas que producen (al igual que ocurría en O₂) no son aprovechadas en este caso, y sí afectan los inconvenientes. A lo expuesto en el apartado de la O₂, hay que añadir que esta organización ahorra almacenar el desplazamiento del primer registro de cada bloque (el primer byte del cubo sería el primer byte de un registro), pero al analizar O₄ se pudo comprobar que esas pequeñas marcas no hacían aumentar el tamaño del archivo, y aplicar la no consecutividad sí que lo harían aumentar notablemente. Por lo tanto, no es necesario comparar el rendimiento de estas organizaciones (peor que el de las consecutivas correspondientes ya analizadas).

Por otro lado, se proporciona información suficiente para plantear una organización O₇ **direccionada dispersa sobre K₁** ($N = 640$, $E_c = 1$), con una dispersión cercana a la uniforme. Si el espacio de cubo es un bloque ($E_c = 1$), el tamaño de cada cubo será 4 registros por cubo ($\lfloor 1.024 \text{ B/cubo} / 211 \text{ B/reg} \rfloor = 4$). Se calcula la densidad de ocupación del área de datos $d_{oc} = 2.000/(640*4) = 78,12\%$, y se concluye que no se producirán desbordamientos, por ser la densidad de ocupación menor del 100% y disponer de una dispersión uniforme. En tales términos, P₁ se resolverá en un solo acceso, y el resto de procesos se realizarán como en una organización serial.

$$C(O_7, P_1) = 1 \text{ acceso}$$

$$C(O_7, P_2) = (640 + 1) / 2 = 320,5 \text{ accesos}$$

$$C(O_7, P_3) = C(O_7, P_4) = 640 \text{ accesos}$$

C(O₇,P) = 400,375 accesos

Se tiene información suficiente para plantear una organización **direccionada dispersa sobre K₃** ($N = 250$, $E_c = 1$), que será la O₈. Con el mismo tamaño de cubo, se tendrá una densidad de ocupación $d_{oc} = 2.000/(4*250) =$

200% que revela la necesidad de gestión de desbordamientos. Para esta O_8 , se elegirá una gestión de desbordamientos por área serial.

Dado que cada valor de K_3 lo tienen 10 registros de media, y solo caben 4 registros en el cubo de datos, se tendrán 6 desbordamientos por cada valor. Si se tienen 200 valores de K_3 , el número total de registros desbordados será de $200 * 6 = 1.200$ regs, que ocupan $1.200 \text{ reg} * 211 \text{ B/reg} = 253.200 \text{ B}$. Por lo tanto, el tamaño del área de desbordamiento sería $\lceil 253.200 \text{ B} / 1.024 \text{ B/bq} \rceil = 248$ bloques. Los procesos P_1 y P_2 serían localizaciones seriales a través de clave de identificación, P_4 sería un recorrido a la totalidad, y P_3 implicaría un acceso al área de datos (recupera cuatro registros) y el recorrido de todo el área de desbordamiento (para recuperar el resto).

$$C(O_8, P_1) = C(O_8, P_2) = (250 + 248 + 1) / 2 = 249,5 \text{ accesos}$$

$$C(O_8, P_3) = 1 + 248 = 249 \text{ accesos}$$

$$C(O_8, P_4) = (250 + 248) = 498 \text{ accesos}$$

$$\mathbf{C(O_8, P) = 311,5 \text{ accesos}}$$

Se puede observar en la organización anterior que la potencia de direccionamiento de la clave (200) es inferior al espacio de direccionamiento (250) y, por tanto, existirán cubos vacíos (50). Para evitar su aparición y su efecto sobre el rendimiento, se puede incorporar un directorio virtual (de 250 direcciones virtuales, que consumirá un bloque), garantizando su permanencia en memoria intermedia (para que no perjudique al rendimiento). De este modo, se tendrá una organización **direcciónada dispersa virtual sobre K_3** ($N = 250$, $E_c = 1$), que será identificada como O_9 . Su rendimiento será análogo al de O_8 , pero suprimiendo los accesos a cubos vacíos.

$$C(O_9, P_1) = C(O_9, P_2) = (200 + 248 + 1) / 2 = 224,5 \text{ accesos}$$

$$C(O_9, P_3) = 1 + 248 = 249 \text{ accesos}$$

$$C(O_9, P_4) = (200 + 248) = 448 \text{ accesos}$$

$$\mathbf{C(O_9, P) = 286,5 \text{ accesos}}$$

Se pueden analizar variantes de esta solución alterando la gestión de desbordamientos. No se plantearán gestiones de desbordamiento por saturación,

ya que el directorio virtual ahorra los cubos vacíos, y al ser una dispersión uniforme se tendrá el área de datos saturada. Sí se podrán plantear gestiones encadenadas del área de desbordamiento. En primer lugar, se analizará O_{10} como una organización **direccional dispersa virtual sobre K_3** ($N = 250, E_c = 1$) con gestión de desbordamientos **encadenada a cubo**. Cada dirección (de las 200 que contienen datos) desborda 6 veces, y como cada cubo alberga 4 registros se precisarán 2 cubos de desbordamiento encadenados ($\lceil 6/4 \rceil = 2$) por cada dirección. En total se tendrán $200 * 2 = 400$ cubos de desbordamiento. El proceso P_3 requerirá un acceso al área de datos y otro por cada cubo encadenado, mientras que el resto de los procesos se calculan de modo análogo a la organización anterior.

$$C(O_{10}, P_1) = C(O_{10}, P_2) = (200 + 400 + 1) / 2 = 300,5 \text{ accesos}$$

$$C(O_{10}, P_3) = 1 + 1 + 1 = 3 \text{ accesos}$$

$$C(O_{10}, P_4) = (200 + 400) = 600 \text{ accesos}$$

$$\mathbf{C(O_{10}, P) = 301 \text{ accesos}}$$

Se comprueba que esta organización favorece mucho a P_3 , penalizando excesivamente al resto (si P_3 fuera más frecuente interesaría esta opción, pero no es el caso). Se podrá probar otra opción más equilibrada, llegando a una organización O_{11} **direccional dispersa virtual sobre K_3** ($N = 250, E_c = 1$) con gestión de desbordamientos **encadenada a registro** sobre área independiente. En este caso, cada registro desbordado incluye un puntero de encadenamiento (con partes alta y baja; 3 bytes en total son suficientes). Se tenían 1200 registros desbordados, que ocuparán $1200 \text{ reg} * (211 + 3 \text{ bytes/reg} = 256.800 \text{ bytes}$, que implica un total de $\lceil 256.800 \text{ bytes} / 1.024 \text{ bytes/bq} \rceil = 251 \text{ bloques}$. El proceso P_3 necesitará realizar un acceso al área de datos y otro por cada registro desbordado, mientras que el cálculo del resto de procesos es análogo al caso anterior.

$$C(O_{11}, P_1) = C(O_{11}, P_2) = (200 + 251 + 1) / 2 = 226 \text{ accesos}$$

$$C(O_{11}, P_3) = 1 + 6 = 7 \text{ accesos}$$

$$C(O_{11}, P_4) = (200 + 251) = 451 \text{ accesos}$$

$$\mathbf{C(O_{11}, P) = 227,5 \text{ accesos}}$$

Por último, se va a plantear la posibilidad de aprovechar que la dispersión es tan buena (uniforme) para hacer el espacio de cubo mayor que un bloque. En primer lugar, puede calcularse el espacio de cubo necesario para que no se produzcan desbordamientos. Para ello, cada dirección debería tener al menos $10 * 211 B = 2.110$ bytes, lo que implica contar con tres bloques ($\lceil 2.110 / 1.024 \rceil = 3$). Se propone entonces la organización O_{12} **direccionada dispersa virtual sobre K_3** ($N = 250$, $E_c = 3$). En este caso, solo se tendrán 200 cubos de datos, pero cada uno de ellos precisa tres accesos. De nuevo, P_3 se satisface de modo direccionado (un acceso a cubo, que implica tres accesos a bloque), y el resto de procesos se resolverá mediante el mecanismo serial.

$$C(O_{12}, P_1) = C(O_{12}, P_2) = ((200 + 1) / 2) * 3 = 301,5 \text{ accesos}$$

$$C(O_{12}, P_3) = 1 * 3 = 3 \text{ accesos}$$

$$C(O_{12}, P_4) = 200 * 3 = 600 \text{ accesos}$$

$$\mathbf{C(O_{12}, P) = 301,5 \text{ accesos}}$$

También se puede proponer un espacio de dos bloques por cubo en el direccionamiento virtual disperso sobre K_3 , lo que ofrece un tamaño de cubo grande ($T_c = 2.048 / 211 = 9$) pero no suficiente para albergar todos los registros de cada dirección. Esto obliga a aplicar una gestión de desbordamiento. Se plantea así la organización **direccionada dispersa virtual sobre K_3** ($N = 250$, $E_c = 2$) y gestión de desbordamientos por área serial (O_{13}). Como cada dirección desborda un solo registro (10 regs/dir y $T_c = 9$), se tendrán 200 registros desbordados que van a ocupar $211 * 200 = 42.200$ B, y en total $\lceil 42.200 / 1.024 \rceil = 42$ bq de desbordamiento.

$$C(O_{13}, P_1) = (1.800 * 201 + 200 * (400 + (42 + 1) / 2)) / 2.000 = 223,05 \text{ accesos}$$

$$C(O_{13}, P_2) = C(O_{13}, P_1) = 223,05 \text{ accesos}$$

$$C(O_{13}, P_3) = 1 * 2 + 42 = 44 \text{ accesos}$$

$$C(O_{13}, P_4) = 200 * 2 + 42 = 442 \text{ accesos}$$

$$\mathbf{C(O_{13}, P) = 233,025 \text{ accesos}}$$

Esa gestión de desbordamientos es mejorable con un encadenamiento. Al ser una dispersión uniforme y desbordar solo un registro por cada dirección, el

encadenamiento a cubo no ofrecerá ventajas en P_3 , y precisará un área de desbordamiento mayor que perjudicaría al rendimiento del resto de procesos. Por esto, la última organización que se va a analizar (O_{14}) será una organización **direccional dispersa virtual sobre K_3** ($N = 250$, $E_c = 2$) y gestión de desbordamientos encadenada a registro sobre área independiente. Se tendrán 200 registros desbordados que ocuparán $213 * 200 = 42.600$ bytes, lo que implica utilizar 42 bloques en el área de desbordamiento (igual que en el caso anterior). El único proceso afectado por el cambio en este caso será, por tanto, P_3 . Obsérvese que en este ejemplo se plantea el área de desbordamiento consecutiva (aunque, por simplificar la implementación, lo habitual es considerarla no consecutiva), y que se está despreciando el impacto de la probabilidad de que el registro se encuentre fraccionado entre dos bloques consecutivos.

$$C(O_{14}, P_1) = C(O_{14}, P_2) = 223,05 \text{ accesos}$$

$$C(O_{14}, P_3) = 1 * 2 + 1 = 3 \text{ accesos}$$

$$C(O_{14}, P_4) = 442 \text{ accesos}$$

$$\mathbf{C(O_{14}, P) = 222,775 \text{ accesos}}$$

Para terminar este análisis, deberán compararse los resultados obtenidos y llegar a una conclusión. Como ha podido comprobarse y se resume en la tabla siguiente, para el problema planteado la mejor de las organizaciones analizadas es la **secuencial consecutiva sobre K_3** (O_4), de coste medio global **209,75** accesos.

Organización	Descripción	Coste
$C(O_1,P)$	Serial consecutiva	310,00
$C(O_3,P)$	Secuencial consecutiva sobre K_1	260,50
$C(O_4,P)$	Secuencial consecutiva sobre K_3	209,75
$C(O_7,P)$	Direccionada dispersa sobre K_1 ($N = 640$, $E_c = 1$)	400,38
$C(O_8,P)$	Direccionada dispersa sobre K_3 ($N = 250$, $E_c = 1$)	311,50
$C(O_9,P)$	Direccionada dispersa virtual sobre K_3 ($N = 250$, $E_c = 1$) con gestión de desbordamientos por área serial	286,50
$C(O_{10},P)$	Direccionada dispersa virtual sobre K_3 ($N = 250$, $E_c = 1$) con gestión desbordamientos encadenada a cubo	301,00
$C(O_{11},P)$	Direccionada dispersa virtual sobre K_3 ($N = 250$, $E_c = 1$) con gestión desbordamientos encadenada a registro	227,50
$C(O_{12},P)$	Direccionada dispersa virtual sobre K_3 ($N = 250$, $E_c = 3$)	301,50
$C(O_{13},P)$	Direccionada dispersa virtual sobre K_3 ($N = 250$, $E_c = 2$) con gestión de desbordamientos por área serial	233,03
$C(O_{14},P)$	Direccionada dispersa virtual sobre K_3 ($N = 250$, $E_c = 2$) con gestión de desbordamientos encadenada a registro	222,78

Tabla 4.1. Comparativa de costes de las organizaciones analizadas sobre el conjunto de procesos P

PROBLEMA 4.4: ORGANIZACIÓN INDIZADA I

El almacenaje de una consola tiene bloques de 2 KB y dispone de memoria intermedia para usos básicos (no se permite bloquear ninguna página en memoria intermedia). Se desea desarrollar una aplicación que requerirá un archivo volátil que va a recoger 15.000 registros de media, organizados de forma serial no consecutiva. La ocupación útil de estos registros es 382,4 caracteres, y su volumen medio es 401 bytes. Los cuatro procesos a los que se somete son equiprobables (25% cada uno), y tienen la siguiente descripción:

- P1. Selección de un registro por clave de identificación (clave K_1 , de tamaño fijo e igual a 9 bytes).
- P2. Selección de todos los registros por clave de búsqueda no identificativa (clave K_2 , de tamaño variable con 10 bytes de tamaño medio, y definida sobre un dominio que presenta 250 valores distintos repartidos uniformemente en el archivo).
- P3. Borrado de un registro por K_1 .
- P4. Listado de todos los registros.
- P5. Inserción de un nuevo registro.

De modo independiente para cada clave de búsqueda (K_1 y K_2), se requiere comparar una serie de estructuras auxiliares y elegir la más eficiente en tiempo (menor influencia en el coste global). En el caso de K_2 , por ser esta no identificativa, se compararán índices secundarios cuyas entradas se construyan con listas de punteros. Las estructuras a analizar son los siguientes índices: **serial** no consecutivo, **secuencial** no consecutivo, árbol B, árbol B+, y árbol B*. Especifíquese también el tamaño auxiliar empleado en cada caso.

NOTA: el impacto de los procesos de reorganización local deberá comentarse, pero no se incluirá en el cálculo de los costes.

SOLUCIÓN PROPUESTA

Este problema se disocia en dos completamente distintos: buscar la estructura adecuada para cada índice. Para cada caso, habrá que establecer el subconjunto de procesos cuyo coste se ve afectado por el índice y comparar los costes solo sobre ese subconjunto. Estos índices serán densos (la organización base no es secuencial) y exhaustivos, por lo que ambos se verán afectados por los procesos de actualización. La clave K_1 interviene en las localizaciones para la primera consulta y el borrado, y se actualiza en borrados e inserciones. Su subconjunto de procesos será: $P' = \{P_1, P_3, P_5\}$. Por otro lado, la clave K_2 interviene en la segunda consulta (y procesos de actualización): $P'' = \{P_2, P_3, P_5\}$.

Para referenciar direcciones de datos, se tomará como puntero externo la dirección del cubo de datos. No se incluirá parte baja en los punteros, dado que no se han especificado procesos de acceso invertido en el enunciado. El tamaño de este puntero dependerá del número de cubos de datos. El tamaño del cubo de datos $T_c = \lfloor 2.048/401 \rfloor = 5$, determina el del archivo $T_{\text{archivo}} = \lceil 15.000/5 \rceil = 3.000$ cubos. El puntero externo tendrá, por tanto, 2 bytes ($\lceil (\log_2 3.000)/8 \rceil = 2$).

Índices sobre $CI = K_1$:

El diseño de la entrada va a constar de clave (tamaño fijo, 9 bytes) y puntero externo. El tamaño de la entrada será, en este caso, $T_{\text{entrada}} = 11$ bytes. Las organizaciones que van a analizarse son las cinco siguientes:

O_1 : org. serial no consecutiva, con un índice serial (no consecutivo) sobre K_1 .

O_2 : org. serial no consecutiva, con un índice secuencial (no consecutivo) sobre K_1 .

O_3 : org. serial no consecutiva, con un índice en árbol B sobre K_1 .

O_4 : org. serial no onsecutiva, con un índice en árbol B_* sobre K_1 .

O_5 : org. serial no consecutiva, con un índice en árbol B_+ sobre K_1 .

En primer lugar, se averigua el tamaño del índice O_1 . En cada cubo de este índice cabrían $2.048/11 = 186$ entradas/cubo. Como es un índice denso, tendrá 15.000 entradas (una por registro). En consecuencia, el tamaño de este índice será $T_{\text{índice}}(O_1) = \lceil 15.000 \text{ ent} / 186 \text{ ent/cubo} \rceil = 81$ cubos. La localización a través de K_1 implica un acceso a la mediana, que indica el coste de P_1 (sumándole un acceso para recuperar el cubo de datos). El borrado (P_3) implica, además de la localización, un acceso de escritura al índice (y la lectura y escritura del cubo de datos correspondiente). Finalmente, el proceso de inserción implica un acceso de lectura (último bloque) y otro de escritura (además de los dos accesos al cubo de datos). Con memoria intermedia muy probablemente podría ahorrarse el acceso de lectura en P_5 . Por otro lado, hay que observar que el borrado dejaría huecos, que podrían eliminarse con alguna gestión de huecos (al ser entradas de tamaño fijo, se puede sobrescribir la borrada con la última entrada del índice, por ejemplo). Los costes sobre P' quedan así:

$$C(O_1, P_1) = (81 + 1)/2 \text{ accesos} + 1 \text{ acceso} = 42 \text{ accesos}$$

$$C(O_1, P_3) = ((81 + 1)/2 + 1 + 2) \text{ accesos} = 44 \text{ accesos}$$

$$C(O_1, P_5) = (2 + 2) \text{ accesos} = 4 \text{ accesos}$$

$$C(O_1, P') = (42 + 44 + 4) \text{ accesos} / 3 = \mathbf{30 \text{ accesos}}$$

Para O_2 , el número inicial de cubos del índice es el mismo (81). El cálculo de costes será similar al caso serial, salvo que en este caso la localización de una entrada implica una búsqueda dicotómica sobre el índice. No obstante, existen notables dificultades con esta organización: los borrados producen huecos más difíciles de reutilizar; la inserción puede hacerse en área desordenada, pero esto penalizaría el coste de localización. Una solución plausible sería dejar espacio libre distribuido para inserciones en el índice (dado que el crecimiento del fichero es nulo, por ser equiprobables inserciones y borrados). Esto afectaría al tamaño del archivo (habría más cubos), pero incluso doblando el tamaño del índice solo se incrementaría en un acceso el coste de la localización. También afectaría a la inserción, que precisará una localización y el posterior acceso de escritura. Aunque esta solución no evita la posibilidad de desbordamiento, la hace mucho más improbable. Por tanto, se reformula O_2 añadiendo 50% de espacio libre distribuido a los cubos de índice.

$$C(O_2, P_1) = \lceil \log_2(162 + 1) \rceil + 1 = 9 \text{ accesos}$$

$$C(O_2, P_3) = \lceil \log_2(162 + 1) \rceil + 1 + 2 = 11 \text{ accesos}$$

$$C(O_2, P_5) = \lceil \log_2(162 + 1) \rceil + 1 + 2 = 11 \text{ accesos}$$

$$C(O_2, P') = (9 + 11 + 11) \text{ accesos} / 3 = \mathbf{10,33 \text{ accesos}}$$

En este punto, se van a analizar los árboles de la familia B. En primer lugar, debe observarse que el índice es pequeño (unos 54 cubos manteniendo un 50% de ocupación) y no se prevé que crezca, por lo que el puntero interno que se requiere tendrá 1 byte de tamaño. Así, el orden de los árboles B y B* se plantea con la siguiente inecuación:

$$\begin{aligned} m * 1 + k * 11 &\leq 2.048; m = k + 1 \rightarrow m \leq 2.059/12 = 171,6 \\ \rightarrow m &= 171 \text{ y } k = 170 \end{aligned}$$

Para analizar coste y tamaño del índice por árbol B, se necesita conocer su ocupación mínima, que será $k_{\min} = \lfloor 170/2 \rfloor = 85$, y $m_{\min} = 85 + 1 = 86$. Con estos datos se puede hallar el tamaño máximo del índice: $T_{\text{índice}}(O_3) \leq \lfloor (e - 1)/k_{\min} \rfloor + 1 \Rightarrow T_{\text{índice}}(O_3) \leq 177$ bloques. Y el coste de la localización de una entrada se calcula por la profundidad máxima del árbol:

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	< 15.000
2	2	2*85	171	< 15.000
3	2*86	172*85	14.791	< 15.000
4	172*86	172*85	1.257.320	> 15.000

Se concluye que localizar una entrada (o la hoja del árbol donde debe insertarse) precisa tres accesos de lectura, más uno de escritura en los procesos de actualización. Recuperar el registro requiere sumar un acceso correspondiente a la lectura del cubo de datos (y otro acceso más para los procesos que deben actualizarlo). El cálculo de costes se resume así:

$$C(O_3, P_1) = 3 + 1 = 4 \text{ accesos}$$

$$C(O_3, P_3) = 3 + 1 + 2 = 6 \text{ accesos}$$

$$C(O_3, P_5) = 3 + 1 + 2 = 6 \text{ accesos}$$

$$C(O_3, P') = (4 + 6 + 6) \text{ accesos} / 3 = \mathbf{5,33 \text{ accesos}}$$

Hay que señalar que los procesos de actualización sobre esta organización (al igual que sobre otros índices de la familia B) requieren eventualmente procesos de reorganización local, que suponen otros 2 ó 3 accesos extra (en la partición y plegado, respectivamente), pero cuyo coste no se ha tenido en cuenta por ser relativamente infrecuentes.

Pasando a analizar coste y tamaño del índice por árbol B^* , se parte del orden calculado para el árbol B (que es el mismo). Lo que cambia es su ocupación mínima (k_{\min} y m_{\min}), que ahora se calculan $k_{\min} = \lfloor 2*170/3 \rfloor = 113$, y $m_{\min} = 114$. A partir de estos datos, se halla el tamaño $T_{\text{índice}}(O_4) \leq \lfloor (e - 1)/K_{\min} \rfloor + 1 \rightarrow T_{\text{índice}}(O_4) \leq 133$ bloques. Y la profundidad del árbol:

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	< 15.000
2	2	2*85	171	< 15.000
3	2*114	228*113	25.764	>15.000

Obsérvese que el nivel 1 (raíz) puede contener solo una entrada, lo que implica que el nivel 2, en el peor caso, solo tendrá dos nodos y cada uno de ellos tendrá una ocupación mínima del 50% (como en el árbol B), ya que provienen de la partición de una raíz. El cambio se produce a partir de este segundo nivel, pues sus nodos solo se partitionan si al desbordar su nodo vecino está lleno (en otro caso, rotan). En este caso, estos procesos impiden la formación del tercer nivel (en el peor caso, la profundidad del árbol es de dos niveles). El cálculo de costes será análogo al del árbol B, con la salvedad de que las reorganizaciones locales son ligeramente más frecuentes (habrá menos particiones, pero aparecen procesos de rotación que exigen también tres accesos extra). El impacto de estos procesos, al igual que en el caso del árbol B, no será tenido en cuenta en los costes.

$$C(O_4, P_1) = 2 + 1 = 3 \text{ accesos}$$

$$C(O_4, P_3) = 2 + 1 + 2 = 5 \text{ accesos}$$

$$C(O_4, P_5) = 2 + 1 + 2 = 5 \text{ accesos}$$

$$C(O_4, P') = (3 + 5 + 5) \text{ accesos} / 3 = \mathbf{4,33 \text{ accesos}}$$

En lo referente al árbol B_+ , habrá que calcular la ocupación mínima de las hojas y el número mínimo de descendientes de los nodos no hoja. En las hojas se

tienen k entradas y un puntero interno (de encadenamiento), $k * 11 + 1 \leq 2.048$ 7
 $k = 186$ 7 $k_{\min} = \lfloor (186 + 1) / 2 \rfloor = 93$. Los nodos no hoja contienen un puntero interno más que claves: $m * 1 + (m - 1) * 9 \leq 2.048$ 7 $m = 205$ y en consecuencia $m_{\min} = \lfloor (205 + 1) / 2 \rfloor = 103$. Con estos datos, se podrá averiguar el número mínimo de hojas (peor caso), e iterativamente el número de ascendientes de cada nivel. Al llegar a un punto en que el nivel requiera (0..2) nodos, se concluye que ese nivel tiene un solo nodo, que es la raíz o nivel 1, y se termina el proceso.
nodos (n) = $\lfloor e / k_{\min} \rfloor = \lfloor 15.000 / 93 \rfloor = 161$ nodos hoja nodos ($n - 1$) = $\lfloor \text{nodos}(n) / m_{\min} \rfloor = \lfloor 161 / 103 \rfloor = 1$ nodo nivel $n - 1 \rightarrow$ nivel $n - 1 =$ nivel raíz.

El número de niveles se podrá despejar: $n - 1 = 1 \rightarrow n = 2$ (profundidad del árbol B_+). El tamaño del índice se halla como el sumatorio del número máximo de nodos de cada nivel: $Tíndice(O_5) \leq (161 + 1)$ bloques. El cálculo de costes será similar a los anteriores (difiere en algunos procesos, como los que hacen uso del encadenamiento por un rango de selección, pero no es el caso). Tampoco se considerará el impacto de las reorganizaciones locales.

$$C(O_5, P_1) = 2 + 1 = 3 \text{ accesos}$$

$$C(O_5, P_3) = 2 + 1 + 2 = 5 \text{ accesos}$$

$$C(O_5, P_5) = 2 + 1 + 2 = 5 \text{ accesos}$$

$$C(O_5, P') = (3 + 5 + 5) \text{ accesos} / 3 = \mathbf{4,33 \text{ accesos}}$$

Para este caso conviene aplicar un índice de la familia B, que ofrecerá mejor rendimiento (aunque haya que padecer eventualmente algunos procesos de reorganización local). De estos, el B presenta ligeramente peor rendimiento. Por otro lado, el B^+ aporta variantes funcionales (que no serán aprovechadas en este caso), y tiene la desventaja de que todas las entradas requieren 2 accesos (el B_* tendrá algunas entradas en la raíz, pero su número es tan bajo con respecto al total, que el impacto sobre el coste medio es despreciable). Por su parte, el B_* tardará más tiempo en desarrollar un tercer nivel (hasta tener más de 25.764 entradas), pero este factor tampoco es significativo puesto que se produce la misma cantidad de inserciones que de borrados (el número de registros no crece). En resumen, cualquiera de las dos opciones es buena, si bien la balanza puede inclinarse hacia el B_+ por tener menor frecuencia (en general) en los procesos de reorganización local.

Índices sobre $CI = K_2$:

El diseño de la entrada va a diferir del caso anterior, puesto que además de la clave (con 10 bytes de media + 1 byte de marca de longitud) y una lista de punteros externos (encabezada por la marca de longitud de lista, de 1 byte, y con tantos punteros externos de 2 bytes como indique la marca; de media son $15.000/250 = 60$ punteros). El tamaño de la entrada será la suma de los elementos descritos $T_{\text{entrada}} = 1 + 10 + 1 + 2 * 60 = 132$ bytes. Las listas tienen tamaño variable: crecen en 2 bytes con las inserciones y decrecen otro tanto con los borrados (que son de un solo registro, por realizarse a través de clave de identificación). Sin embargo, el número de entradas se mantiene constante (por ser borrados e inserciones equiprobables), y cada cubo tendrá una pequeña cantidad de espacio libre para evitar desbordamientos. Las organizaciones a analizar son las siguientes:

O_6 : org. serial no consecutiva, con un índice serial (no consecutivo) sobre K_2 .

O_7 : org. serial no consecutiva, con un índice secuencial (no consecutivo) sobre K_2 .

O_8 : org. serial no consecutiva, con un índice en árbol B sobre K_2 .

O_9 : org. serial no consecutiva, con un índice en árbol B_* sobre K_2 .

O_{10} : org. serial no consecutiva, con un índice en árbol B_+ sobre K_2 .

Con estas entradas, cada valor de la CI solo aparecerá una vez. La localización serial no requerirá, por tanto, un recorrido a la totalidad sino que se detendrá al encontrar la entrada buscada. En contrapartida, la inserción requiere localizar la lista con el valor de clave que se va a insertar, y añadir su puntero. En cada cubo de índice caben $2.048/136 = 15$ entradas, lo que deja $2.048 - (132*15) = 68$ bytes de espacio libre en cada cubo para aumentar el tamaño de sus listas (hasta 34 punteros). Como en total existirán 250 entradas (una por valor en el dominio de la CI), el tamaño del índice será $T_{\text{índice}}(O_6) = \lceil 250/15 \rceil = 17$ bq. En el proceso P_2 habrá que tener en cuenta que, de media, se proporcionan 60 resultados.

$$C(O_6, P_2) = (17 + 1) / 2 \text{ accesos} + 60 \text{ accesos} = 69 \text{ accesos}$$

$$C(O_6, P_3) = ((17 + 1)/2 + 1 + 2) \text{ accesos} = 12 \text{ accesos}$$

$$C(O_6, P_5) = ((17 + 1)/2 + 1 + 2) \text{ accesos} = 12 \text{ accesos}$$

$$C(O_6, P'') = (69 + 12 + 12) \text{ accesos} / 3 = \mathbf{31 \text{ accesos}}$$

Si las listas estuvieran organizadas de modo secuencial (O_7) solo se tendrán ventajas, al reducir el coste de la localización por la búsqueda dicotómica:

$$C(O_7, P_2) = \log_2(17 + 1) \text{ accesos} + 60 \text{ accesos} = 65 \text{ accesos}$$

$$C(O_7, P_3) = (\log_2(17 + 1) + 1 + 2) \text{ accesos} = 8 \text{ accesos}$$

$$C(O_7, P_5) = (\log_2(17 + 1) + 1 + 2) \text{ accesos} = 8 \text{ accesos}$$

$$C(O_7, P'') = (65 + 8 + 8) \text{ accesos} / 3 = \mathbf{27 \text{ accesos}}$$

Los cálculos sobre árboles B y B_* comienzan, como anteriormente, averiguando el orden:

$$m * 1 + k * 132 \leq 2.048; m = k + 1 \quad 7 \quad m \leq 2.180/133 = 16,39$$

$$\rightarrow m = 16 \text{ y } k = 15$$

Lo siguiente es calcular su ocupación mínima, $k_{\min} = \lfloor 15/2 \rfloor = 7$, y el número mínimo de descendientes $m_{\min} = 8$. Así, el tamaño máximo del índice queda: $T_{\text{Índice}}(O_8) \leq \lfloor (e - 1) / k_{\min} \rfloor + 1 \quad 7 \quad T_{\text{Índice}}(O_8) \leq 36$ bloques. Finalmente, se calcula la profundidad máxima del árbol B:

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	< 250
2	2	2*7	15	< 250
3	2*8	16*7	127	< 250
4	16*8	128*7	1023	>= 250

Se concluye que localizar una entrada (o la hoja del árbol para su inserción) precisa tres accesos de lectura, más uno de escritura en los procesos de actualización. El proceso de consulta P_2 añadirá los accesos a los cubos de datos (60 registros, en general en cubos distintos), y los de actualización solo la lectura y escritura del cubo afectado (se borra o inserta solo un registro). En el cálculo de costes no se incluirán los que son debidos a reorganizaciones locales (que además serán aún menos frecuentes que en los casos analizados anteriormente).

Estos cálculos se exponen a continuación:

$$C(O_8, P_2) = 3 \text{ accesos} + 60 \text{ accesos} = 63 \text{ accesos}$$

$$C(O_8, P_3) = (3 + 1 + 2) \text{ accesos} = 6 \text{ accesos}$$

$$C(O_8, P_5) = (3 + 1 + 2) \text{ accesos} = 6 \text{ accesos}$$

$$C(O_8, P'') = (63 + 6 + 6) \text{ accesos} / 3 = \mathbf{25 \text{ accesos}}$$

Para realizar el cálculo de costes de O_9 , incluyendo un índice en árbol B_* , no es necesario recalcular el orden (igual que el apartado anterior) pero sí la ocupación mínima y, en consecuencia, el tamaño y profundidad del árbol. En primer lugar, $k_{\min} = \lfloor 15 * 2 / 3 \rfloor = 10$, y por tanto $m_{\min} = 11$. Así, el tamaño máximo del índice: $T_{\text{índice}}(O_9) \leq \lfloor (e - 1) / k_{\min} \rfloor + 1 \rightarrow T_{\text{índice}}(O_9) \leq 25$ bloques. Finalmente, la profundidad máxima del árbol será la siguiente:

Nivel	Nodos	Entradas	Ent. acumuladas
1	1	1	1
2	2	2*10	15
3	2*11	22*10	235
4	22*11	242*10	2.655

Como puede comprobarse, aunque el tamaño del índice es menor (es más denso), la profundidad del árbol será la misma, y por tanto los costes quedarán para O_9 igual que los obtenidos para O_8 . Este índice B_* ofrecerá la ventaja de no aumentar el número de niveles hasta insertar más de 2.655 listas (frente a las 1.024 que ya harían desbordar el árbol B), pero dado que no se informa de la posibilidad de aumentar de 250 valores en el dominio de K_2 (no habrá mas de 250 listas) será preferible optar por O_8 (de momento).

Finalmente, para analizar O_{10} habrá que calcular la ocupación mínima de las hojas. Se tiene que $k * 132 + 1 \leq 2.048 \rightarrow k = 15 \rightarrow k_{\min} = \lfloor (15 + 1) / 2 \rfloor = 8$. Además, se calculará el número mínimo de descendientes de nodo no hoja a partir de punteros internos y claves (sin puntero externo): $m * 1 + (m - 1) * (1 + 10) \leq 2.048 \rightarrow m = 171 \rightarrow m_{\min} = \lfloor (171 + 1) / 2 \rfloor = 86$. A partir de estos datos, se averigua el número mínimo de hojas e iterativamente el número mínimo de ascendientes de cada nivel, hasta llegar a la raíz:

$$\text{nodos (n)} = \lfloor e/k_{\min} \rfloor = \lfloor 250/8 \rfloor = 31 \text{ nodos hoja}$$

$$\text{nodos } (n - 1) = \lfloor \text{nodos}(n) / m_{\min} \rfloor = \lfloor 31 / 86 \rfloor = 0 \rightarrow 1 \text{ nodo de nivel } n - 1$$
$$\rightarrow n - 1 = 1$$

La profundidad del árbol B_+ en este caso es dos ($n = 2$). El tamaño del índice será mayor que O_9 , ya que $T\text{índice}(O_{10}) \leq (31 + 1)$ bloques, pero su rendimiento será algo mejor:

$$C(O_{10}, P_2) = 2 \text{ accesos} + 60 \text{ accesos} = 62 \text{ accesos}$$

$$C(O_{10}, P_3) = (2 + 1 + 2) \text{ accesos} = 5 \text{ accesos}$$

$$C(O_{10}, P_5) = (2 + 1 + 2) \text{ accesos} = 5 \text{ accesos}$$

$$C(O_{10}, P'') = (62 + 5 + 5) \text{ accesos} / 3 = \mathbf{24 \text{ accesos}}$$

En conclusión, sobre la clave K_2 es interesante incorporar un índice en árbol B^+ con listas de punteros (se deja al lector la comprobación numérica de la conveniencia de las listas).

PROBLEMA 4.5: ORGANIZACIÓN INDIZADA II

Sobre un soporte direccionado, cuyo bloque tiene tamaño de 2 KB y 12 ms de tiempo de acceso, se tiene un fichero de 1.000 registros cuya información útil tiene una extensión media de 100 bytes. La organización base de este fichero es serial no consecutiva, con 4 bloques de espacio de cubo ($E_c = 8$ KB). Dado que los cubos carecen de directorio de cubo, la posición de un registro en el cubo se determinará a partir del desplazamiento de su primer byte (*offset*), que requiere 2 bytes en este caso. La densidad ideal del registro es del 90%, y las previsiones de crecimiento del archivo apuntan a que nunca llegará a duplicar su volumen. Se pide resolver los siguientes apartados:

- A) ¿Cuál es su densidad real? ¿Y cuál es el tamaño del fichero?
- B) Se tiene una clave de identificación K_1 de 9 bytes (tamaño fijo), y se desea realizar un índice primario serial consecutivo. Este índice se utilizará solo para recuperación del registro completo (P_1), y nunca para acceso invertido. ¿Cuáles serían su tamaño y el tiempo máximo de acceso a registro aleatorio a través de K_1 ?
- C) En lugar del índice serial del apartado anterior, se propone analizar otro con estructura en árbol B sobre la misma clave de indización. Se tendrá en cuenta que se puede dedicar una página de memoria intermedia para almacenar la raíz del árbol B permanentemente. ¿Cuál es el tamaño máximo (cota superior) del fichero índice? ¿Cuál es el número máximo de accesos (cota superior) para recuperar una entrada de índice dada? Finalmente, ¿cómo queda el tiempo máximo de acceso a registro aleatorio a través de K_1 ?
- D) Por otro lado, se tienen dos procesos (P_2 y P_3) consistentes en obtener los valores que presenta la clave K_2 en los registros en los que otra clave K_3 tiene cierto valor (P_2), y viceversa (P_3 consistirá en obtener los valores de K_3 en los registros en los que K_2 tiene un valor dado). Estos procesos podrán realizarse mediante accesos invertidos sobre sendos índices para las claves K_2 y K_3 , ambas de tamaño variable (15 y 24 bytes de media respectivamente) y no identificativas (definidas sobre dominios cerrados y reducidos, de 5 y 25 valores

distintos respectivamente, y que aparecen con la misma frecuencia). Se deben hacer, para cada caso, los cálculos pertinentes para establecer la conveniencia de utilizar listas invertidas o esquemas de bits en estas claves (esquemas de bits simples y listas invertidas contiguas a la clave, es decir, almacenadas junto con la clave). Suponiendo que se carece de memoria intermedia para bloquear estos índices, ¿cuál es el tiempo de acceso para resolver el proceso P_2 ? ¿Y para el proceso P_3 ?

- E) Averigua cuáles serían los costes de P_2 y P_3 si no se utilizaran accesos invertidos en ellos (sí se realizarán a través de los índices disponibles, o por la organización base).
- F) Se establece que el tamaño máximo de cubo en este archivo es de 90 registros/cubo (conservando los mismos tamaño medio de cubo y tamaño actual de fichero). Se considera la posibilidad de realizar esquemas de bits de puntero implícito. Analizar su conveniencia en cada caso (K_2 y K_3).

SOLUCIÓN PROPUESTA

Apartado A: a partir de la densidad ideal se podrá averiguar el volumen real del registro, y después con este dato se calcularía el tamaño del cubo. Dividiendo el número de registros entre el tamaño del cubo se obtiene el tamaño del fichero, y con éste se halla la densidad real.

$$d_{\text{ideal}} = 100 \text{ bytes} / T_{\text{registro}} = 90\% \rightarrow T_{\text{registro}} = 111,11 \text{ bytes}$$

$$T_{\text{cubo}} = \lfloor 8.192 \text{ bytes/cubo} / 111,11 \text{ bytes/registro} \rfloor = 73 \text{ registros/cubo}$$

$$T_{\text{fichero}} = \lceil 1.000 \text{ registros} / 73 \text{ registros/cubo} \rceil = 14 \text{ cubos} \rightarrow T_{\text{fichero}} = 112 \text{ KB}$$

$$d_{\text{real}} = 100 * 1.000 \text{ bytes} / 112 * 1.024 \text{ bytes} \rightarrow d_{\text{real}} = 87,19 \%$$

Apartado B: dado que el archivo tiene solo 14 cubos, con un byte es suficiente para el puntero interno. La entrada tendrá, por tanto $9 + 1 = 10$ bytes de tamaño. Contendrá 1.000 entradas, almacenadas serialmente en bloques de 2 KB, luego su tamaño será de 5 bloques (**10 KB**), y el tiempo máximo para el proceso P_1 será el necesario para recorrer todo el índice más 4 accesos para leer el cubo de datos ($E_c = 4$) donde esté almacenado el registro buscado:

$$T_{\text{índice}} = \lceil 1.000 \text{ entradas} * 10 \text{ B/entrada} / 2.048 \text{ B/bloque} \rceil = 5 \text{ bloques}$$

$$C(O_1, P_1) = 5 \text{ acc} + 4 \text{ acc} = 9 \text{ acc} \rightarrow t_{\text{máx}}(P_1) = 9 \text{ acc} * 12 \text{ ms/acc} = \mathbf{108 \text{ ms}}$$

Apartado C: este índice tendrá entradas y punteros internos (estos últimos de 1 byte de tamaño, porque el índice será presumiblemente pequeño a la vista del tamaño del índice serial). En cada nodo del índice (de 1 bloque) deben caber m punteros internos y k claves, por tanto:

$$1 * m + 10 * k \leq 2.048 \rightarrow m \leq 187,1 \rightarrow m = 187 \text{ y } m = k + 1 \rightarrow k = 186$$

$$k_{\min} = \lfloor 186/2 \rfloor = 93 \rightarrow m_{\min} = 94, \text{ y } T_{\text{índice}} \leq \lfloor (1.000 - 1) / 93 \rfloor + 1 = \mathbf{11 \text{ bq}}$$

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	< 1.000
2	2	2*93	187	< 1.000
3	2*94	188*93	17.671	> 1.000

Como la profundidad del árbol es de dos niveles, y la raíz estará permanentemente bloqueada en memoria intermedia, el coste máximo para recuperar una entrada de índice será **1 acceso**. Para hallar el coste de acceso a registro, se suman los accesos para recuperar un cubo $C(O_1, P_1) = 1 + 4 = 5$ accesos. Finalmente, el tiempo máximo de acceso a registro aleatorio se obtiene a partir de la siguiente expresión: $t_{\max}(P_1) = 5 \text{ acc} * 12 \text{ ms/acc} = \mathbf{60 \text{ ms}}$.

Apartado D: los índices invertidos necesitan que el puntero externo identifique únicamente cada registro, por lo que se añade al puntero relativo una parte baja que localiza al registro dentro de su cubo (en este caso, el desplazamiento de 2 bytes). El puntero externo que utilizarán estos índices tendrá, por tanto, un tamaño de 3 bytes.

Para K_2 (cuyo dominio tiene cardinalidad 5), cada esquema de bits tendría 5 bits (1 byte) más tres bytes de puntero externo. El índice tendría en total 1.000 esquemas (uno por registro), y por tanto ocuparía 4.000 bytes (dos bloques). En cambio, para K_3 cada esquema de bits tendría 25 bits (4 bytes) más los 3 bytes del puntero externo, y el total del índice (los 1.000 esquemas) ocuparía 7.000 bytes (4 bloques).

En lo referente a las listas invertidas, su longitud dependerá de lo que se repita cada valor: K_2 presenta 5 valores repartidos uniformemente (cada valor aparece en $1.000/5 = 200$ registros), y K_3 admite 25 valores (cada uno de ellos aparecerá en $1.000/25 = 40$ registros). Para K_2 , las listas se compondrán de marca de longitud de la clave (1 byte), valor de la clave (de media 15 bytes), marca de longitud de la lista (1 byte), y lista de punteros ($200 * 3$ bytes). Análogamente, las listas sobre K_3 tendrán 1 byte de marca de longitud de la clave, 24 bytes para la clave, otro para la marca de longitud de la lista, y 120 bytes para la lista de punteros ($40 * 3$ bytes). En total, cada una de las 5 listas del índice sobre K_2 ocuparía 617 bytes, lo que implica que en cada bloque caben $\lfloor 2.048/617 \rfloor = 3$ listas, y que se necesitan 2 bloques para almacenar las 5 listas. Por otro lado, cada una de las 25 listas sobre K_3 totaliza un tamaño medio de 146 bytes, por lo que caben $\lfloor 2.048/146 \rfloor = 14$ listas en cada bloque, y en solo dos bloques cabrán las 25 listas.

Los esquemas de bits se recorren serialmente en ambas direcciones del acceso invertido, por lo que su tamaño es crucial para su conveniencia. La inserción de nuevos esquemas es óptima (serial), pero este proceso no está definido en este caso por lo que no será tenida en cuenta esta característica.

Como solo se van a valorar los procesos P_2 y P_3 , las **listas invertidas** serán la mejor opción **sobre K_3** porque solo ocupan 2 bloques (que precisan 2 accesos en el peor caso, y $(14 * 1 + 11 * 2) / 25 = 1,44$ accesos de media) frente a los 4 bloques de los esquemas de bits). El caso de K_2 es más complicado, pues ambas estructuras ocupan 2 bloques. Para la segunda parte del acceso invertido, ambas tendrán un comportamiento similar (recorrido a la totalidad, en general). Sin embargo, para la primera parte los esquemas de bits precisan un recorrido a la totalidad y no así las listas invertidas: 3 de las listas se localizan en 1 acceso, y el resto en 2 accesos (1,4 accesos de media). Por lo que, también **para K_2** , se escogería la opción de las **listas invertidas**.

Como estos índices ocupan poco, se podría procurar su permanencia en memoria intermedia (P_2 y P_3 tendrían coste cero en accesos a soporte, en tal caso). Sin embargo, el problema descarta explícitamente esta posibilidad, por lo que el coste medio quedaría:

$$C(O, P_2) = 1,4 \text{ accesos} + 2 \text{ accesos} = \mathbf{3,4 \text{ accesos}}$$

$$C(O, P_3) = 1,44 \text{ accesos} + 2 \text{ accesos} = \mathbf{3,44 \text{ accesos}}$$

Apartado E: si no se hicieran por acceso invertido, los procesos P_2 y P_3 podrían resolverse con una búsqueda serial (pues la organización base es serial). Al estar ambos basados en una selección sobre clave no identificativa, precisarían un acceso serial a la totalidad que implica recuperar 14 cubos (de $E_c = 4$), como se calculó en el apartado (a). Por tanto, el tiempo necesario para realizar estos procesos sería: $t(P_2) = t(P_3) = 14 * 4 * 12 = \mathbf{672 \text{ ms.}}$

Dado que existen índices, también se podrían realizar mediante acceso indizado. No obstante, esta posibilidad es aquí aún peor que la serial, pues en ambos casos el número de resultados es superior al número de cubos (200 y 40 resultados respectivamente, frente a 14 cubos), por lo que en el peor caso (y muy frecuentemente) se van a tener que recuperar todos los cubos de datos (como en el proceso serial) y además se habrán empleado algunos accesos (1,4 y 1,44 respectivamente) en consultar los índices.

Apartado F: los esquemas de bits de puntero implícito evitan almacenar el puntero externo situando el esquema de bits en el índice en la misma posición relativa que ocupa el registro al que hacen referencia en el fichero de datos. Para que esto sea posible, cada cubo debe tener un tamaño máximo establecido, y cada registro (ocurra o no) tendrá su esquema de bits reservado. Además, a cada

esquema de bits se le añade un bit que refleja si el registro de esa posición ocurre o no (por ejemplo, 0 si no existe, y 1 en otro caso).

En este caso, para K_2 los esquemas tendrían 6 bits (1 byte) y para K_3 necesitarían 26 bits (4 bytes). Estos índices ya no tendrán 1.000 entradas, sino 90 entradas por cubo (tamaño máximo del cubo). Como hay 14 cubos, se necesitarán $14 * 90 = 1.260$ entradas. En el caso de la $CI = K_2$, el índice completo ocupará un solo bloque (1.260 bytes), mientras que para la $CI = K_3$ ocupará 3 bloques ($\lceil 1.260 \text{ ent} * 4 \text{ bytes/ent} / 2.048 \text{ bytes/bloque} \rceil = 3$ bloques).

Las ventajas que puede ofrecer esta variante del esquema de bits son su menor tamaño, y que la segunda parte del acceso invertido ya no es serial, sino que se accede directamente a los esquemas de bits involucrados (puesto que se conoce su dirección). Sin embargo, esto último no será una ventaja en este caso, dado que el número medio de resultados que proporcionan los procesos P_2 y P_3 es mucho mayor que el número de bloques del índice. En concreto, el proceso P_2 , después de consultar el índice por K_2 tendría que recuperar 200 esquemas de bits, lo que implicaría recuperar los tres bloques del índice para K_3 en la mayor parte de los casos. Este análisis se suma al hecho de que el índice por listas invertidas para K_3 es de menor tamaño (2 bloques) y número medio de accesos para la primera parte del acceso invertido (1,44 accesos), y conduce a la conclusión de que para K_3 sigue interesando aquella opción. Sin embargo, el reducido tamaño del índice por esquemas de bits para K_2 (1 bloque) y el ahorro de accesos para consultarlos (en ambas partes del acceso invertido) llevan a concluir que se obtiene cierta mejoría con esta opción.

Resultado:

- para $K_2 \rightarrow$ esquemas de bits de puntero implícito
- para $K_3 \rightarrow$ listas invertidas

PROBLEMA 4.6: DISEÑO Y COMPARACIÓN DE ORGANIZACIONES

Una empresa que se dedica al alquiler de videojuegos desea informatizar la información relativa a sus productos con el fin de realizar una gestión y manejo de la información más eficiente. La información que desea almacenar se recoge en la siguiente descripción:

- (1) De cada videojuego se precisa saber la marca (máximo 12 caracteres pero de media solo 6,2), el género (máximo 15 caracteres y de media 7), y la edad mínima recomendable (dos dígitos numéricos como máximo y de media 1,6).
- (2) Además, se desea almacenar el nombre del videojuego (único, máximo 60 caracteres y de media 18,8), el número de unidades de ese videojuego (de media 5, pero puede haber como máximo 10 ejemplares) y el modelo de consola en la que funciona (9 caracteres de máximo y 5 de media).
- (3) Junto a lo anterior, es importante saber si el juego necesita algún accesorio para poder jugar (hecho que solo ocurre el 5% de las veces), y cuáles (cuando se necesitan, se recogen de media 2,3 accesorios y como máximo 5). Cada accesorio se identificará por un tipo (micrófono, cámara, acelerómetro, volante, pistola, espada, fotosensor, pulsador, alfombra, guitarra, o batería) con 7,5 caracteres de media y la cantidad de ese tipo de accesorio que es necesaria (de media 1,2, aunque como máximo podrá ser necesario tener 2).
- (4) Otra característica interesante es el número de veces que ha sido alquilado el videojuego (máximo 5.000 veces, y de media 134). De los alquileres no se requiere recoger información (solo saber cuántas veces se ha alquilado).
- (5) De los procesos que afectan a este fichero, solo se desea optimizar el rendimiento de dos de ellos, por ser críticos, y que son equiprobables. A saber:
 - Selección por el nombre del videojuego (K_1 identificativo).

- Selección por género del videojuego (K_2 con 50 registros de media por valor, y 52 registros por valor como máximo).

Observando la descripción anterior, se pide:

- A) Realizar el diseño lógico que recoja toda la información.
- B) Plantear el diseño físico-lógico pésimo (registro de tamaño fijo), y calcular su densidad ideal.
- C) Realizar un diseño físico-lógico optimizado, y calcular su densidad ideal.
- D) Se requiere almacenar 5.000 registros que siguen el diseño anterior. Se tiene una dispersión uniforme sobre $N = 40$ para el campo que contiene el nombre del videojuego (K_1). Finalmente se plantea una organización direccional dispersa (con $E_c = 2$ bloques, y $T_b = 2$ KB). Resta decidir cómo debe ser la gestión de los registros desbordados, de modo que se minimice el coste de los procesos críticos, barajándose las siguientes opciones:
 - Gestión de desbordamiento serial no consecutiva.
 - Gestión de desbordamientos con encadenamiento a cubo.
 - Gestión de desbordamientos con encadenamiento a registro.
 - Gestión de desbordamientos con saturación.
- E) Con el fin de mejorar el coste en accesos se desea implementar un índice con lista de punteros sobre el género del videojuego. Las estructuras propuestas para ese índice son el árbol B y el árbol B₊. Analizar cuál sería la mejor opción atendiendo al coste. Discutir la conveniencia de cambiar la organización base (seleccionada en el apartado anterior) y si esto podría disminuir el coste medio.

SOLUCIÓN PROPUESTA

Apartado A: el diseño lógico que se extrae de los supuestos semánticos explícitos aportados en el enunciado (puntos a, b, c y d), es el siguiente:

Marca	C(12),
Género	C(15),
EdadMínima	N(2),
<u>Nombre</u>	C(60),
NumUnidades	N(2),
Consola	C(9),
Accesorio (
Tipo	C(12),
Cantidad	N(1)
) ^{*5} ,	
NumVecesAlquilado	N(4)

En la notación adoptada, los tipos de campo N y C se refieren a su naturaleza numérica o alfanumérica, respectivamente, marcando entre paréntesis su tamaño (máximo). Observar que el elemento de datos ‘Accesorio’ es un vector compuesto de dos campos, y además es grupo repetitivo (de cero a cinco veces).

Apartado B: el diseño físico-lógico pésimo se construye con campos de longitud fija e igual al tamaño máximo, de manera que cualquier registro posible quepa en el espacio asignado. Así, el diseño físico-lógico pésimo es el siguiente:

Marca	b(12),
Género	b(15),
EdadMínima	b(2),
<u>Nombre</u>	b(60),
NumUnidades	b(2),
Consola	b(9),
Accesorio (
Tipo	b(12),
Cantidad	b(1)
) ⁵ ,	
NumVecesAlquilado	b(4)

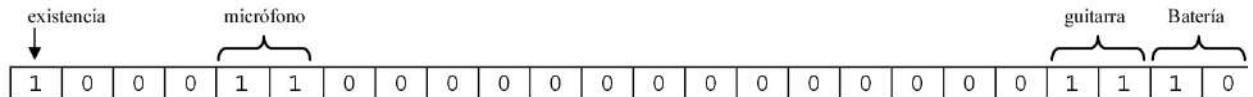
Una vez obtenido este diseño, se podrá calcular su densidad ideal como la relación entre información útil y volumen real. La información útil del registro son 45,58 B ($6,2 + 7 + 1,6 + 18,8 + 2 + 5 + 0,05 * (2,3 * (7,5 + 1)) + 4$). Y su volumen real siguiendo este diseño son 169 B ($12 + 15 + 2 + 60 + 2 + 9 + 5 * (12 + 1) + 4$). Finalmente, la densidad ideal de ese registro es $d_i = 45,58/169 = 26,97\%$.

Apartado C: para mejorar la densidad ideal del registro, es preciso reducir el espacio que se utiliza, especialmente el necesario para garantizar su correcta interpretación. Así, se eliminarán los caracteres de relleno mediante la introducción de campos de control (marcas) que indicarán la longitud del campo que venga a continuación, o la ocurrencia y repetición de elementos opcionales y repetitivos.

Además, también es conveniente codificar los campos donde sea posible, reduciendo la cantidad de bytes necesarios. De este modo, los campos numéricos podrán codificarse en base 256 (en lugar de caracteres en base decimal), y el tipo de accesorio en un byte enumerado. Este elemento de datos podría quedar así:

NumAccesorio	b (1) ,
Accesorio (
Tipo	b (1) ,
Cantidad	b (1)
) NumAccesorio ,	

En este diseño, la información útil del elemento son $2,3 * 2 = 4,6$ B, y la real 5,6 B, al añadirle la marca de reiteración. Sin embargo, si en lugar de un byte enumerado se incluye un mapa de 11 bits (uno por cada valor en el dominio), se contempla la repetición de valores (ahorrando el grupo repetitivo y su marca). Por otro lado, la cantidad solo admite los valores 1 y 2, por lo que podría codificarse en el mismo mapa con otros 11 bits. El resultado sería un mapa de 3 bytes (22 bits) que son al tiempo útiles y reales (se reduce el volumen y se aumenta la densidad). Dado que sobran bits, se puede reservar uno de ellos para marcar la existencia (si el primer bit es un 1, ocurren los tres bytes; si es un 0, los otros dos bytes no se almacenan), ahorrando en media 1,9 B. Adoptando el orden de valores definido en el enunciado, y siendo el primer bit el tipo y el segundo la cantidad, un ejemplo de codificación para registrar “dos micrófonos, dos guitarras y una batería” sería:



El diseño físico-lógico queda como se muestra debajo, siendo su ocupación útil de 41,15 bytes ($6,2 + 7 + 1 + 18,8 + 1 + 5 + 0,05 * 3 + 2$), su volumen real de 46,1 bytes ($1 + 6,2 + 1 + 7 + 1 + 1 + 18,8 + 1 + 1 + 5 + (1+0,05 * 2) + 2$), y su densidad ideal **89,26%**.

LongMarca	b(1),
Marca	b(LongMarca),
LongGénero	b(1),
Género	b(LongGénero),
EdadMínima	b(1),
LongNombre	b(1),
Nombre	b(LongNombre),
NumUnidades	b(1),
LongConsola	b(1),
Consola	b(LongConsola),
Accesorios (
bitmap1	b(1)
bitmap2	b(1) bitmap1[7]=1
bitmap3	b(1) bitmap1[7]=1
)	
NumVecesAlquilado	b(2)

Apartado D: comenzamos este apartado analizando la gestión de desbordamientos serial no consecutiva (O_1). Para calcular el coste de cada proceso es necesario averiguar cuántos registros desbordan.

El número de registros lógicos que caben de media en cada cubo (tamaño de cubo) es $T_c = \lfloor (2.048 \text{ B/bloque} * 2 \text{ bloques/cubo}) / 46,1 \text{ B/reg} \rfloor = 88 \text{ reg/cubo}$. Por ser una dispersión uniforme, cada cubo recibe el mismo número de registros, concretamente $5.000 \text{ reg} / 40 \text{ direcciones} = 125 \text{ reg/cubo}$. Por tanto, en cada dirección desbordan $125 - 88 = 37 \text{ reg}$. Si desbordan $37 \text{ reg/cubo} * 40 \text{ cubos}$, en total se tendrán 1.480 registros desbordados, que suponen el 29,6% del total. Estos registros almacenados en área serial no consecutiva ocupan $\lceil 1.480 / 88 \rceil = 17 \text{ cubos}$.

Llegados a este punto, se pueden hallar los costes de los procesos descritos:

$$C(O_1, P_1) = (1 + (0,296 * (17 + 1)/2)) * E_c = 7,33 \text{ accesos}$$

$$C(O_1, P_2) = (40 + 17) * E_c = 57 * 2 = 114 \text{ accesos}$$

$$C(O_1, P) = 0,5 * C(O_1, P_1) + 0,5 * C(O_1, P_2) = 0,5 * 7,33 + 0,5 * 114 = \mathbf{60,67 \text{ accesos}}$$

El proceso P_1 tiene un coste elevado para tratarse de un direccionamiento. Esto es por tener una tasa de desbordamiento elevada y un coste muy alto para localizar esos registros desbordados. Si, en cambio, la gestión de desbordamientos hubiera contado con encadenamiento a cubo (O_2), este proceso se vería muy beneficiado. En primer lugar, hay que tener en cuenta que se necesita incorporar al cubo nueva información de control (puntero de encadenamiento), que indica cuál de los cubos de desbordamiento es el asociado a ese cubo de datos (por haber menos de 256 cubos de desbordamiento, con un byte será suficiente). En este caso, este puntero no afecta al tamaño de cubo $T_c = \lfloor (4.096 \text{ B} - 1 \text{ B}) / 46,1 \text{ B/reg} \rfloor = 88 \text{ reg/cubo}$. De este modo, cada cubo desbordado tendrá un cubo de desbordamiento (ya que los registros desbordados en cada dirección caben en un cubo 37 reg. < 88 reg/cubo). En resumen, se tendrían 40 cubos de desbordamiento y los siguientes costes:

$$C(O_2, P_1) = (1 + 0,296 * 1) * E_c = 2,59 \text{ accesos}$$

$$C(O_2, P_2) = (40 + 40) * E_c = 160 \text{ accesos}$$

$$C(O_2, P) = 0,5 * C(O_2, P_1) + 0,5 * C(O_2, P_2) = 0,5 * 2,59 + 0,5 * 160 = \mathbf{81,3 \text{ accesos}}$$

Se observa que la introducción de esta gestión no es eficiente, sin duda por tener un proceso que precisa un recorrido a la totalidad (P_2) y un área de desbordamiento demasiado grande. El tamaño de esta área podría reducirse con una gestión de desbordamientos con encadenamiento a registro (O_3). En este caso, el cubo tiene un puntero identificativo al primer registro desbordado (puntero que indica en qué cubo está, con un byte, y en qué posición está almacenado, con 2 bytes). Este puntero tampoco afecta al tamaño de cubo de datos. Sin embargo, con esta organización, los registros desbordados deberán ir acompañados de otro puntero de encadenamiento, que incrementa su tamaño en tres bytes. Así, el tamaño de cubo de desbordamiento es $T_c = \lfloor 4.096 / 49,1 \text{ B/reg} \rfloor = 83 \text{ reg/cubo}$, y el número de cubos de desbordamiento será $\lceil 1.480 / 83 \rceil = 18$ cubos, claramente menor. Sin embargo, al ser la longitud de la cadena (37 registros) superior al número de cubos de desbordamiento, deberá considerarse como peor caso que los registros desbordados están distribuidos entre los 18

cubos, y por tanto recorrer el encadenamiento equivale a recorrer todo el área de desbordamiento. El cálculo de costes es análogo al realizado para O_1 , pero con 18 cubos de desbordamiento en este caso.

Como última alternativa, se propone la gestión de desbordamientos con saturación (O_4). Esta opción, sin embargo, no tiene sentido ya que el área de datos está saturada (densidad de ocupación 100%) y esta gestión de desbordamientos no conseguiría almacenar ningún registro. Por tanto, entre todas las opciones planteadas, la de **mayor eficiencia en coste** es la O_1 .

Finalmente, fuera de los parámetros del enunciado, se puede analizar la conveniencia de aumentar el espacio de cubo (O_5). En este caso, con $E_c = 3$ bloques se tendría $T_c = \lfloor 2.048 * 3 / 46,1 \text{ B/reg} \rfloor = 133 \text{ reg/cubo}$, que es superior a las necesidades de almacenamiento para cada dirección (125 registros) por lo que no habría ningún desbordamiento. Sin embargo, como se comprueba en el siguiente cálculo de costes, el resultado no es mejor que el obtenido con O_1 :

$$C(O_5, P_1) = 1 * E_c = 3 \text{ accesos}$$

$$C(O_5, P_2) = 40 * E_c = 120 \text{ accesos}$$

$$C(O_5, P) = 0,5 * C(O_2, P_1) + 0,5 * C(O_2, P_2) = 0,5 * 3 + 0,5 * 120 = \mathbf{61,5 \text{ accesos}}$$

Apartado E: en este apartado se mejorará la eficiencia del proceso P_2 por la introducción de un índice. Respecto a la pregunta acerca de si sería buena idea cambiar de organización base, con el índice el coste de aquel proceso se hará independiente de la organización base, aunque no del espacio de cubo. Por tanto, interesa adoptar la organización base que optimice el otro proceso (P_1) sin que utilice cubos de datos grandes (que perjudican el coste del acceso indizado). Esta organización base no es otra que O_2 , con 2,59 accesos medios para P_1 y $E_c = 2$ (el más pequeño de los planteados). Resta decidir qué estructura es la más adecuada para ese índice, y calcular los costes globales de la organización definitiva (O_6).

El uso del índice es localizar cubos que contienen registros seleccionados, y para ello solo es necesario identificar un cubo entre 80, por lo que 1 byte es suficiente para actuar de puntero externo. El diseño físico lógico de la entrada tendrá clave (marca de longitud + cadena) y apuntamiento. Este último consiste en un grupo repetitivo que contiene un puntero y ocurre un máximo de 52 veces

y en media 50, y que por tanto se puede implementar con una marca de reiteración de 1 byte. En resumen, el tamaño de la entrada será $T_e = 1 + 7 + 1 + 50 * 1 = 59$ B. Dado que hay 5.000 registros, que la longitud media de la lista es de 50 punteros, y que la clave de indización no es multivaluada (cada registro tiene un solo valor y por tanto aparece solo en una lista de punteros), se tendrán $5.000/50=100$ entradas.

El puntero interno, de menor o igual tamaño que el externo, también será de 1 byte. Cada página de índice contendrá un nodo. El tamaño de página interesa que sea pequeño, y por eso se escogerá un bloque. En la estructura árbol B, encontramos que cada nodo contiene m punteros a descendiente (internos) y k entradas, siendo los descendientes uno más que las entradas. Formalmente, $m * 1 + k * 59 \leq 2.048$ y $m = k + 1$, de donde $k = 34$ y $m = 35$. Las cotas inferiores $k_{\min} = 34/2 = 17$ y $m_{\min} = 18$ son los datos necesarios para hallar la profundidad máxima (en el peor caso), que es de $n = 2$ niveles tal y como se muestra en la tabla a continuación:

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	≤ 100
2	2	$2*17=34$	35	≤ 100
3	$2*18=36$	$36*17=612$	647	>100

Para analizar el índice con estructura de árbol B^+ , se calculan por separado ocupación y orden. La ocupación de las hojas $k * 59 + 1 \leq 2.048 \rightarrow k = 34$ y $k_{\min} = (k + 1) / 2 = 17$. Por el contrario, el orden (número máximo de descendientes) define a los nodos no hoja, en cuyas entradas no aparecen punteros externos (solo valores de indización): $m * 1 + (m - 1) * (1 + 7) \leq 2.048 \rightarrow m = 228$ y $m_{\min} = 114$. En este caso, el número de niveles se calcula por divisiones sucesivas (*bottom-up*):

$$\text{nodos } (n) = \lfloor e / k_{\min} \rfloor = \lfloor 100/17 \rfloor = 5 \text{ nodos hoja}$$

$$\text{nodos } (n - 1) = \lfloor \text{nodos}(n) / m_{\min} \rfloor = \lfloor 5/114 \rfloor \in (0, 2) \rightarrow \text{nodo raíz } (1 \text{ nodo})$$

Por lo tanto, este índice en árbol B^+ consta de 6 nodos y su profundidad también es 2 niveles (como en el árbol B), si bien su funcionalidad y capacidad son mucho mayores (el árbol B podría llegar a 3 niveles con 647 entradas

mientras que el árbol B⁺ no crecería hasta albergar 3876 entradas). Como ambos producen el mismo coste, se procederá a adoptar una de estas dos soluciones (cualquiera) para concluir el ejercicio con el pertinente cálculo de costes sobre O_6 (O_2 con índice). Se asume que la raíz del árbol está en memoria intermedia, y que en la distribución de resultados se da el peor caso (50 resultados en 50 cubos distintos).

$$C(O_6, P_1) = C(O_2, P_1) = 2,59 \text{ accesos}$$

$$C(O_6, P_2) = 1 + 50 = 51 \text{ accesos}$$

$$C(O_6, P) = 0,5 * C(O_6, P_1) + 0,5 * C(O_6, P_2) = 26,8 \text{ accesos}$$

PROBLEMA 4.7: ACCESO MULTICLAVE

Un museo de arte desea informatizar la información referente a los cuadros y esculturas que obran en su poder (en total 100.000 registros con un volumen real de 180 bytes y una información útil de 150 bytes). De entre toda la información almacenada por cada obra de arte, merece la pena destacar los siguientes campos:

- K_1 : **Nombre de la obra**: único, con 20 caracteres de media y 30 de máximo.
- K_2 : **Autor**: presenta de media 10 registros por valor (16 como máximo). Está compuesto por 50 caracteres como máximo y 23 de media.
- K_3 : **Fecha de creación de la obra** (día/mes/año): único, 4 B (longitud fija).

Los principales procesos que se realizan sobre la información almacenada son los que se indican a continuación ($f(P) = \{40\%, 20\%, 10\%, 20\%, 5\%, 5\%\}$):

- Consulta selectiva por K_1 .
- Consulta selectiva por K_2 .
- Consulta por selección de un rango de K_3 (de media devuelve 200 registros).
- Consulta selectiva por K_3 .
- Modificación de todas las claves seleccionando por K_2 .
- Borrado de un registro seleccionando por K_3 .

Observando la información anterior, se pide analizar los siguientes puntos:

- A) Se tiene un soporte de acceso directo con tamaño de bloque 2 KB. También se cuenta con un diseño de cubo con $E_c = 2$ bloques y espacio libre distribuido para modificaciones de 10% (PCTFREE = 10%), que es suficiente para absorber todas las modificaciones que se realicen en el cubo, y en cada cubo se reservan 2 B para información

de control.

Comparar las siguientes organizaciones base en función del coste global de los procesos descritos:

O_1 : Organización serial no consecutiva.

O_2 : Organización secuencial no consecutiva con clave de ordenación K_1 .

O_3 : Organización direccionada dispersa sobre $CD = K_1$ y $N = 8.000$ cubos; tasa de desbordamiento del 4%, gestión de desbordamientos en área independiente serial no consecutiva.

B) Se decide incorporar índices para mejorar el coste de algunos de los procesos frecuentes. Los índices que se desea estudiar se aplicarán sobre las claves K_2 (con lista de punteros) y K_3 , y seguirán una estructura en árbol B o árbol B⁺. ¿Cuál es la combinación más eficiente (organización base y auxiliares) en función del coste global?

NOTA: la página de índice no sigue el mismo diseño que los cubos de datos.

C) Se observa que el tercer proceso tiene unas características peculiares: la condición se establece sobre K_3 , y el proceso devuelve el valor de K_2 para ese registro (el resto de campos no son relevantes). Se opta por implementar este proceso mediante un acceso invertido, para lo que se precisa contar con sendos índices invertidos sobre estas claves.

¿Qué combinación de índices de los anteriormente mencionados sería bueno tener en cuenta para cada clave en función de los costes globales? ¿Sería interesante cambiar la organización base?

SOLUCIÓN PROPUESTA

Apartado A: se comenzará con la organización serial no consecutiva (O_1).

Para calcular los accesos que requiere cada proceso en esta organización, debe calcularse el número de cubos de datos que conlleva, y para ello el tamaño de cubo. En este cálculo hay que tener en cuenta que parte del espacio de cubo se usa en información de control, y que del espacio utilizado para almacenar registros debe reservarse el 10% como espacio libre distribuido para modificaciones. Así, se tiene $T_c = \lfloor (2.048 \text{ B/bloque} * 2 \text{ bloques/cubo} - 2\text{B}) * 0,9 / 180 \text{ B/reg} \rfloor = 20 \text{ reg/cubo}$.

A partir de este dato y de la cantidad total de registros a almacenar, se calcula el tamaño de O_1 como $T_{\text{archivo}} = \lceil 100.000 \text{ reg} / 20 \text{ reg/cubo} \rceil = 5.000$ cubos. En función de este dato, se calculan los costes para la organización serial. Los procesos P_1 , P_4 y P_6 consisten en una localización por clave identificativa, mientras que los otros tres requieren un recorrido a la totalidad. Además, el proceso P_5 añade un acceso de escritura por cada registro modificado (10 de media), siempre en el cubo donde se localiza gracias al espacio libre distribuido. Por último, el proceso P_6 añade un acceso de escritura para eliminar el registro seleccionado. Matemáticamente:

$$C(O_1, P_1) = C(O_1, P_4) = ((1 + 5.000)/2) * E_c = 5001 \text{ accesos}$$

$$C(O_1, P_2) = C(O_1, P_3) = 5.000 * E_c = 10.000 \text{ accesos}$$

$$C(O_1, P_5) = (5.000 + 10) * E_c = 10.020 \text{ accesos}$$

$$C(O_1, P_6) = ((1 + 5.000)/2) + 1) * E_c = 5.003 \text{ accesos}$$

$$\begin{aligned} C(O_1, P) &= 0,4 * C(O_1, P_1) + 0,2 * C(O_1, P_2) + 0,1 * C(O_1, P_3) + 0,2 * \\ &C(O_1, P_4) + 0,05 * C(O_1, P_5) + 0,05 * C(O_1, P_6) = 0,4 * 5.001 + 0,2 * 10.000 + \\ &0,1 * 10.000 + 0,2 * 5.001 + 0,05 * 10.020 + 0,05 * 5.003 = \mathbf{6.751,75 \text{ accesos}} \end{aligned}$$

En lo que respecta a O_2 , organización secuencial con clave de ordenación K_1 no consecutiva, el número inicial de cubos será el mismo de O_1 : $T_{\text{archivo}} = 5.000$ cubos. Los costes serán similares, con dos salvedades: en el proceso P_1 , la localización por K_1 ya no será serial sino por búsqueda dicotómica (binaria); y en

segundo lugar, al modificarse el contenido de la clave de ordenación durante P_5 , ya no podrá almacenarse en la misma posición, sino que el efecto de la modificación será el de borrado de los registros afectados y reinserción de los mismos (en un área desordenada serial). Por tanto, se trata de una organización que degenera al ir incrementándose el volumen del área serial desordenada, cuestión que afecta al coste de todos los procesos. Como aproximación, se calculan los costes iniciales:

$$C(O_2, P_1) = \lceil \log_2 (5.000 + 1) \rceil * E_c = 26 \text{ accesos}$$

$$C(O_2, P_2) = C(O_1, P_2) = 10.000 \text{ accesos}$$

$$C(O_2, P_3) = C(O_1, P_3) = 10.000 \text{ accesos}$$

$$C(O_2, P_4) = C(O_1, P_4) = 5.001 \text{ accesos}$$

$$C(O_2, P_5) = (5.000 + 10 * (1 + 1)) * E_c = 10.040 \text{ accesos}$$

$$C(O_2, P_6) = C(O_1, P_6) = 5.003 \text{ accesos}$$

$$C(O_2, P) = 0,4 * C(O_2, P_1) + 0,2 * C(O_2, P_2) + 0,1 * C(O_2, P_3) + 0,2 * C(O_2, P_4) + 0,05 * C(O_2, P_5) + 0,05 * C(O_2, P_6) = \mathbf{4.758,75 \text{ accesos}}$$

Por último, es el turno de analizar la organización direccional dispersa con clave de direccionamiento K_1 y área de desbordamiento serial no consecutiva (O_3). Como en otras ocasiones, se debe calcular el número de registros desbordados, y a partir de ahí el volumen del área desbordada. La tasa de desbordamiento es del 4%, lo que significa que el número de registros desbordados será $100.000 * 0,04 = 4.000$ reg. Estos registros se guardarán en el área de desbordamiento, cuyo tamaño de cubo es idéntico al de las dos organizaciones anteriores (20 reg/cubo), y que permite calcular el tamaño del área serial $T_{desbord} = \lceil 4.000 \text{ reg}/20 \text{ reg/cubo} \rceil = 200$ cubos. Todos los costes se verán afectados, al cambiar el tamaño del fichero. Pero además el proceso P_1 se ve beneficiado por el acceso direccional, y en el proceso P_5 hay que tener en cuenta que al modificarse el valor de la CD, en general, se borrará el registro de su localización reinsertándose en otro cubo. Se tendrán:

$$C(O_3, P_1) = (1 + 0,04 * ((1+200)/2)) * E_c = 10,04 \text{ accesos}$$

$$C(O_3, P_2) = C(O_3, P_3) = (8.000 + 200) * E_c = 16.400 \text{ accesos}$$

$$C(O_3, P_4) = ((1 + 8.200)/2) * E_c = 8.201 \text{ accesos}$$

$$C(O_3, P_5) = (8.200 + 10 * (1 + 1 + 0,04 * 1)) * E_c = 16.440,8 \text{ accesos}$$

$$C(O_3, P_6) = ((8.200 + 1)/2 + 1) * E_c = 8.203 \text{ accesos}$$

$$C(O_3, P) = \sum_i f(i) * C(O_3, P_i) = \mathbf{7.796,41 \text{ accesos}}$$

En resumen, y atendiendo a estos resultados, la mejor organización es O_2 .

Apartado B: para la inclusión de índices, es necesario pensar en el diseño de las entradas. En primer lugar debe observarse que los tamaños de fichero en todas las organizaciones oscilan desde 5.000 a 8.200 cubos, espacios de direccionamiento que en todos los casos pueden cubrirse con 2 bytes, que será el tamaño del puntero externo. En el caso de la clave de indización K_3 , la entrada tendrá 4 bytes fijos para la clave y 2 más para el puntero (en total 6 bytes). En el caso de K_2 en cambio es necesario diseñar entradas con listas de punteros, por ser un índice secundario. Así, para la clave serán necesarios 24 bytes (1 de marca de longitud + 23 bytes de ocupación media), y para la lista de punteros se precisa un byte de marca de reiteración y 10 punteros de media (20 B). En total, la entrada de K_2 requiere 45 B.

Analizaremos primero el caso del índice secundario K_2 . Este contiene 10_4 entradas, ya que las listas de punteros son disjuntas (no hay multivaluación en K_2) cada lista contiene 10 punteros, y son un total de 10_5 registros. En un árbol B, cada nodo (almacenado en una página de 2 KB) contendrá k entradas y otros tantos punteros a descendiente más uno ($m = k + 1$). Teniendo en cuenta que un índice serial consecutivo ocuparía 220 bloques, está claro que el número de nodos de un árbol B será superior a 256 y, por tanto, el puntero interno necesario tendrá 2 B. Así, se tiene $2 * m + 45 * k \leq 2.048 \rightarrow k = 43$ y $m = 44 \rightarrow k_{\min} = 21$ y $m_{\min} = 22$.

Con estos datos se puede proceder a calcular la profundidad del árbol:

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	≤ 10.000
2	2	$2*21$	43	≤ 10.000
3	$2*22$	$44*21$	967	≤ 10.000
4	$44*22$	$968*21$	21.295	> 10.000

En el peor caso, este árbol tendrá **tres niveles de profundidad**.

Para conocer las características del índice estructurado según un B_+ , han de calcularse la ocupación de las hojas y el orden de los nodos no hoja. En las hojas se tienen $k * 45 + 2 \leq 2.048 \rightarrow k = 45$ entradas $7 k_{\min} = \lfloor (45 + 1) / 2 \rfloor = 23$ entradas. Por otro lado, en nodos no hoja hay $m * 2 + (m - 1) * 24 \leq 2.048 \rightarrow m = 79$ descendientes como máximo, y como mínimo $m_{\min} = \lfloor (79 + 1)/2 \rfloor = 40$. El siguiente paso para averiguar la profundidad del árbol es calcular el número máximo de hojas, e iterativamente el número máximo de ascendientes por nivel hasta llegar a la raíz.

$$\text{nodos } (n) = \lfloor e/k_{\min} \rfloor = \lfloor 10.000/23 \rfloor = 434 \text{ nodos hoja}$$

$$\text{nodos } (n - 1) = \lfloor \text{nodos}(n)/m_{\min} \rfloor = \lfloor 434/40 \rfloor = 10 \text{ nodos en el nivel } n - 1$$

$$\text{nodos } (n - 2) = \lfloor \text{nodos}(n-1)/m_{\min} \rfloor = \lfloor 10/40 \rfloor \in (0,2) \rightarrow \text{raíz (nivel 1)}$$

Por tanto, la profundidad del árbol B^+ será de **tres niveles**, igual que el B .

El caso del índice primario (K_3) es más sencillo. Su orden, calculado según lo siguiente $2 * m + 6 * k \leq 2.048$ $7 k = 255$, $m = 256$, $k_{\min} = 127$ y $m_{\min} = 128$. Sus 10^5 entradas, una por registro, se distribuirán en el peor caso en tres niveles:

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	≤ 100.000
2	2	$2*127$	255	≤ 100.000
3	$2*128$	$256*127$	32.767	≤ 100.000
4	$256*128$	$32.768*127$	$4.194.303$	> 100.000

Si este índice se hubiera hecho con estructura B_+ , la ocupación de las hojas sería $k * 6 + 2 \leq 2.048 \rightarrow k = 341$ entradas $\rightarrow k_{\min} = \lfloor (341 + 1) / 2 \rfloor = 171$ entradas. En los nodos no hoja habrá $m * 2 + (m - 1) * 4 \leq 2.048 \rightarrow m = 342$ punteros a descendiente $\rightarrow m_{\min} = \lfloor (342 + 1) / 2 \rfloor = 171$ descendientes como mínimo. Con estos datos, y mediante divisiones sucesivas, se podrá calcular la profundidad:

$$\text{nodos } (n) = \lfloor e / k_{\min} \rfloor = \lfloor 100.000/171 \rfloor = 584 \text{ nodos hoja}$$

$$\text{nodos } (n - 1) = \lfloor \text{nodos}(n) / m_{\min} \rfloor = \lfloor 584/171 \rfloor = 3 \text{ nodos en el nivel } n - 1$$

$$\text{nodos } (n - 2) = \lfloor \text{nodos}(n - 1) / m_{\min} \rfloor = \lfloor 3/171 \rfloor \in (0,2) \rightarrow \text{raíz (nivel 1)}$$

Por lo tanto, la profundidad del árbol B_+ también es **tres niveles**. Dado que todos los índices tienen la misma profundidad, no es relevante cuál se aplique. Sin embargo, para K_3 se preferirá el árbol B^+ por existir un proceso mixto (rango) en P_3 , para el que resultará muy beneficioso el encadenamiento de las hojas (con 2 accesos se encuentra la primera entrada, y con dos más se garantizan 200 coincidencias ya que el número mínimo de entradas en una hoja es 171).

Una vez se conocen las características de todos los índices, se procede a calcular los costes de las organizaciones actualizadas con la inclusión de índices. Se notarán con el mismo símbolo de la organización base pero añadiendo un apóstrofo para significar que es una organización distinta (que incluye esos índices). En todos los cálculos se va a considerar que el nodo raíz de un árbol está siempre en memoria. Notar que todos los procesos añaden el número de accesos a cubos de datos (multiplicado por E_c) y que los procesos P_5 y P_6 deben incluir accesos para actualizar el índice (en el caso de P_6 1 acceso de escritura para borrar; y en el caso de P_5 , al modificarse la clave de indización de los diez registros, deberá eliminarse la entrada e insertar 10 nuevas entradas (como nuevas o incorporando un puntero a otras tantas listas existentes). El coste de este proceso sumará los accesos necesarios para: leer la entrada del índice K_2 (2 accesos); leer los 10 cubos de datos apuntados, y reescribir el registro modificado en la misma posición dado que cabe gracias al espacio libre distribuido (10 + 10 accesos a cubo, dos accesos a bloque por cubo); borrar la entrada de índice obsoleta en K_2 (1 acceso); reinserir las 10 entradas en K_2 (10 veces 2 accesos para localizar la posición y otro de escritura para añadir la entrada). Matemáticamente:

$$C(O'_1, P_1) = C(O_1, P_1) = 5.001 \text{ accesos}$$

$$C(O'_1, P_2) = 2 + 10 * E_c = 22 \text{ accesos}$$

$$C(O'_1, P_3) = (2 + 2) + 200 * E_c = 404 \text{ accesos}$$

$$C(O'_1, P_4) = 2 + 1 * E_c = 4 \text{ accesos}$$

$$C(O'_1, P_5) = 2 + 10 * (1 + 1) * E_c + 1 + 10 * (2 + 1) = 73 \text{ accesos}$$

$$C(O'_1, P_6) = 2 + (1 + 1) * E_c + 1 = 7 \text{ accesos}$$

$$C(O_1', P) = 2.050 \text{ accesos}$$

En las otras dos organizaciones, dado que las claves privilegiadas no afectan a los índices, los resultados que se tendrán son prácticamente los mismos, a excepción del proceso P_1 , para el que el coste será el ya calculado para la organización base, y del proceso P_5 . En efecto, en la organización secuencial, ya que los registros cambian de ubicación, en lugar de escribir el cubo para modificar el registro hay borrarlo y luego insertarlo en un área desordenada (un acceso a cubo más por cada registro). En la direccionada, tras borrar el registro hay que leer el cubo de su nueva dirección (1 acceso) e insertarlo allí si cabe (1 acceso), y si no cabe se inserta en el área de desbordamiento (1 acceso); en total, en ambos casos, se añaden dos accesos a cubo de datos por cada registro. Finalmente, dado que se ha modificado la localización de los registros, hay que actualizar los punteros de las 10 entradas en K_1 (10 veces, dos accesos para localizar la entrada y uno más de escritura para actualizarla). Para resumir, no se repetirá la descripción de estos costes parciales, y sí directamente los costes globales de ambas organizaciones.

$$C(O_2', P_5) = 2 + 10 * (1+1+1) * E_c + 1 + 10 * (2+1) + 10 * (2+1) = 123 \text{ acc}$$

$$C(O_2', P) = 0,4 * 26 + 0,2 * 22 + 0,1 * 404 + 0,2 * 4 + 0,05 * 123 + 0,05 * 7$$

$$C(O_2', P) = 62,5 \text{ accesos}$$

$$C(O_3', P_5) = 2 + 10 * (1+1+1+1) * E_c + 1 + 10 * (2+1) + 10 * (2+1)$$

$$C(O_3', P_5) = 143 \text{ accesos}$$

$$C(O_3', P) = 0,4 * 10,04 + 0,2 * 22 + 0,1 * 404 + 0,2 * 4 + 0,05 * 143 + 0,05 * 7$$

$$C(O_3', P) = 56,77 \text{ accesos}$$

Y por lo tanto, la **mejor organización** atendiendo a los costes será la O_3' .

Apartado C: puesto que la descripción de P_4 ha cambiado y apunta a un acceso invertido, será necesario plantear estructuras auxiliares invertidas para analizar su conveniencia. El diseño es común al apartado anterior salvo porque los punteros externos son identificativos, para lo que se deberá incorporar una parte baja que indique la posición del registro dentro del cubo. En todas las organizaciones base de este problema el cubo tiene 4 KB, y así la posición se codifica en 2 bytes (tamaño puntero externo 2 + 2 = 4 bytes). El diseño es

similar, pero el tamaño bien distinto, ya que la entrada de K_3 pasa de 6 bytes a 8 bytes, y la entrada de K_2 pasa de 45 bytes a $(24 + 1 + 10 * 4) = 65$ bytes.

También debe repetirse el cálculo del orden y, por ende, el de la profundidad de cada árbol. Comenzando por el índice secundario K_2 , para la estructura en árbol B se tiene $2 * m + 65 * k \leq 2.048$ $k = 30$ y $m = 31$, y los mínimos serán $k_{\min} = 15$ y $m_{\min} = 16$.

Con estos datos procedemos a calcular la profundidad del árbol B:

Nivel	Nodos	Entradas	Ent. acumuladas	
1	1	1	1	≤ 10.000
2	2	$2 * 15 = 30$	31	≤ 10.000
3	$2 * 16 = 32$	$32 * 15 = 480$	511	≤ 10.000
4	$32 * 16 = 512$	$512 * 15 = 7.680$	8.191	≤ 10.000
5	$512 * 16 = 8.192$	$8.192 * 15 = 122.880$	131.071	> 10.000

En el peor caso, este árbol B tendrá **cuatro niveles** de profundidad.

Respecto al B_+ para esta clave, se tienen $k * 65 + 2 \leq 2.048 \rightarrow k = 31$ entradas $\rightarrow k_{\min} = \lfloor (31 + 1) / 2 \rfloor = 16$ entradas. Por otro lado, los nodos no hoja tendrán las mismas características ($m = 79$ y $m_{\min} = 40$) que el índice no invertido, ya que los nodos no hoja carecen de punteros externos. Por tanto, la profundidad:

$$\text{nodos } (n) = \lfloor e / k_{\min} \rfloor = \lfloor 10.000 / 16 \rfloor = 625 \text{ nodos hoja}$$

$$\text{nodos } (n - 1) = \lfloor \text{nodos}(n) / m_{\min} \rfloor = \lfloor 625 / 40 \rfloor = 15 \text{ nodos en el nivel } n - 1$$

$$\text{nodos } (n - 2) = \lfloor \text{nodos}(n - 1) / m_{\min} \rfloor = \lfloor 15 / 40 \rfloor \in (0,2) \rightarrow \text{raíz (nivel 1)}$$

Por tanto, la profundidad del árbol B^+ será de **3 niveles**, igual que antes y a diferencia del B, lo que justifica su uso. Pero además, al ser este índice utilizado en la segunda fase del acceso invertido (proyección), el encadenamiento de las hojas supone una ventaja para el acceso invertido. En suma, la estructura elegida es B^+ .

Para el índice por clave K_3 , el orden $2 * m + 8 * k \leq 2.048 \rightarrow k = 204$ y además $m = 205$, y consiguientemente $k_{\min} = 102$ y $m_{\min} = 103$. Este árbol

mantiene sus tres niveles de profundidad, según se sigue:

Nivel	Nodos	Entradas	Ent. acumuladas
1	1	1	1
2	2	$2 * 102 = 204$	205
3	$2 * 103 = 206$	$206 * 102 = 21.012$	21.217
4	$\cancel{206 * 103}$ $= 21.218$	$\cancel{21.218 * 102}$ $= 2.164.236$	≥ 100.000

A la misma conclusión se llegará sobre la estructura B_+ , la ocupación de las hojas es $k * 8 + 2 \leq 2.048 \rightarrow k = 255$ entradas $\rightarrow k_{\min} = \lfloor (255 + 1) / 2 \rfloor = 128$ entradas. En nodos no hoja se mantienen $m = 342$ y $m_{\min} = 171$. Y con todo esto:

$$\text{nodos}(n) = \lfloor e / k_{\min} \rfloor = \lfloor 100.000 / 128 \rfloor = 781 \text{ nodos hoja}$$

$$\text{nodos}(n - 1) = \lfloor \text{nodos}(n) / m_{\min} \rfloor = \lfloor 781 / 171 \rfloor = 4 \text{ nodos en el nivel } n - 1$$

$$\text{nodos}(n - 2) = \lfloor \text{nodos}(n - 1) / m_{\min} \rfloor = \lfloor 4 / 171 \rfloor \in (0,2) \rightarrow \text{raíz (nivel 1)}$$

Por tanto, se mantiene la decisión de optar por la estructura en árbol B^+ para el índice por K_3 , que es más conveniente para selecciones en un rango como ocurre en el proceso P_3 . Así, ambos índices comparten estructura y, dado que tienen la misma profundidad que los del apartado anterior, se mantienen los costes asociados a los procesos. La excepción, claro está, es el proceso P_3 cuya definición ha cambiado y que tratará de resolverse mediante acceso invertido.

Este acceso invertido tendrá dos fases: *selección*, recuperando un conjunto de punteros (200, de media) sobre el índice K_3 ; y proyección, durante el que se accederá al encadenamiento de las hojas del índice sobre K_2 para traducir esos punteros en su valor de clave. La selección requiere la búsqueda de la cota inferior en el árbol de tres niveles (2 accesos) y el recorrido del rango mediante el encadenamiento de las hojas (2 accesos más en el peor caso). Por otro lado, la proyección debe recorrer todas las hojas del índice K_2 (625 hojas, 625 accesos). En total, el coste de este proceso parece peor que el obtenido en el apartado anterior, con acceso indizado en lugar de acceso invertido.

$$C(O_3'', P_3) = (2 + 2) + 625 = 629 \text{ accesos}$$

(mayor que $C(O_3', P_3) = 404$ accesos).

Sin embargo, hay que resaltar que esos 629 son accesos a bloques de índice, y que serán los mismos bloques en todas las ejecuciones que se hagan de ese proceso, independientemente del rango definido para la selección. Es cierto que en la organización anterior se tenían 404 accesos, pero 400 de estos accesos (el 99%) son a cubo de datos, y es poco probable encontrar este tipo de bloques en memoria intermedia, ya que cambian de una consulta a otra. Por tanto, la tasa de acierto en memoria intermedia para el acceso invertido se presume elevada y no así la del acceso indizado. Esto puede suponer que la mayoría de accesos sean lógicos, y el coste efectivo (número de accesos físicos) sea muy reducido. Es habitual encontrar los bloques de índice en memoria intermedia (ya que son muy accedidos), e incluso el administrador puede bloquear en memoria intermedia una estructura. Así, manteniendo los índices en memoria intermedia, el acceso invertido tendría un coste efectivo de cero accesos físicos, mientras que el indizado tendría probablemente una cifra cercana a los 400 accesos físicos.

En resumen, el acceso indizado presenta menor coste $C(O_3', P_3)$, pero se debería optar por el acceso invertido ya que éste ofrece mejor coste efectivo (menos accesos físicos).

Capítulo 5

ALMACENAMIENTO Y DISEÑO FÍSICO EN ORACLE

A continuación aplicaremos los conceptos vistos en el capítulo anterior para estudiar las estructuras de almacenamiento de un SGBD. En particular utilizaremos ORACLE, un popular Sistema Gestor de Bases de Datos Relacionales con una arquitectura Cliente/Servidor.

En la terminología de Oracle, una Base de Datos es un conjunto de infomaciones que se almacenan en archivos y a los que se accede a partir de una instancia o ejemplar de la base de datos. La instancia es el conjunto de estructuras en memoria y de procesos que gestionan la base de datos y a partir de la cual las aplicaciones pueden acceder a estos.

Para lograr la independencia de los datos frente a los procesos que acceden a ellos, Oracle hace uso de la tradicional separación entre las unidades de almacenamiento lógico y físico. De esta forma, una base de datos en Oracle está formada por una o varias unidades de almacenamiento lógico, conocidas como espacios de tablas (*tablespace*). Cada espacio de tablas (Figura 5.1) está compuesto de archivos de datos, que son las estructuras físicas propiamente dichas. Este nivel de abstracción permite que Oracle pueda delegar la gestión de los ficheros a diferentes sistemas operativos e incluso llevar a cabo una gestión propia.

5.1 ESPACIOS DE TABLAS

Oracle permite la creación de diferentes tipos de espacios de tablas con propósitos diferentes:

- **SYSTEM**, es el espacio de tablas de sistema donde se almacena el diccionario de datos así como disparadores y otros procedimientos almacenados una vez compilados. Es creado automáticamente por Oracle en el momento de crear la base de datos.
- **DATA**, se trata de los espacios de tablas adecuados para el

almacenamiento de datos de usuario como tablas e índices. Cada usuario tiene asignado un espacio de tablas por defecto en el que se almacenan sus objetos. Aunque es posible almacenar datos en *SYSTEM* no es una práctica recomendable. Es posible, y generalmente deseable, tener diferentes espacios de tablas para un mismo esquema si los objetos son dispares. De esta forma se puede tener un espacio de tablas para los índices y otro para las tablas con tamaños de bloque diferentes.

- **TEMP**, se trata de un espacio de tablas que almacena información temporal en disco, como resultados de operaciones de ordenación o agrupación que no es posible almacenar en memoria. Cada usuario tiene un espacio de tablas temporal asignado por defecto.
- **UNDO**, es el espacio de tablas usado para almacenar la información de deshacer necesaria para el soporte transaccional. A partir de la versión Oracle 9i, es posible que el SGBD gestione de forma automática el espacio necesario mediante el parámetro *UNDO_MANAGEMENT = AUTO*. Con la opción manual, son los segmentos de *rollback* o de retroceso los encargados de almacenar esta información.

Los espacios de tablas se dividen en unidades, llamadas segmentos, en las que se almacena un objeto específico de la base de datos como una tabla o un índice. Cada segmento está a su vez compuesto de extensiones, que se corresponde con el espacio reservado para el almacenamiento de una estructura. Cada extensión se compone de un conjunto de bloques de datos contiguos. El bloque de datos es la unidad mínima de transferencia de información para una base de datos Oracle, y su tamaño debe ser múltiplo del tamaño de bloque físico definido por el sistema operativo.

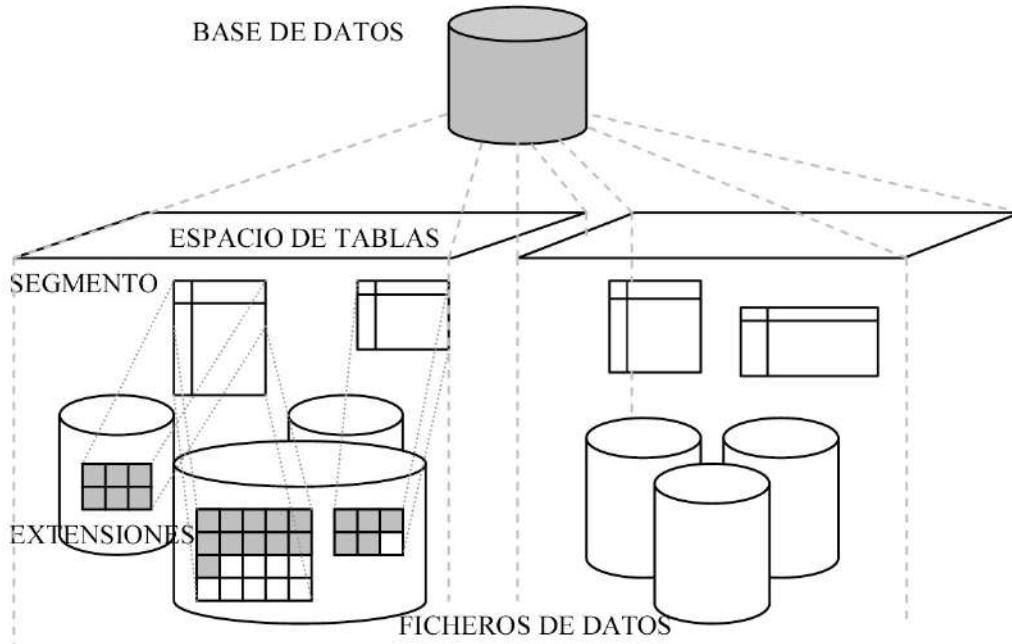


Figura 5.1. Estructuras de almacenamiento en Oracle

Los espacios de tablas pueden encontrarse en dos estados diferentes, en línea y fuera de línea. Cuando el espacio de tablas está en línea la base de datos tiene acceso a los datos. Cuando el espacio de tablas está fuera de línea los datos no se encuentran disponibles para la base de datos, sin embargo se pueden realizar operaciones de mantenimiento, resguardo o traslado del espacio de tablas.

5.1.1 Segmentos y extensiones

Cada objeto de la base de datos se almacena en su propio segmento determinado que inicialmente está compuesto de una extensión pero que puede ir añadiendo extensiones de manera dinámica a medida que se necesita más espacio. A la hora de crear un espacio de tablas, podemos definir el tamaño por defecto de las extensiones de ese espacio de tablas, pero también es posible definir las características específicas para un objeto, mediante el uso de la cláusula *STORAGE*. Esta cláusula permite el control detallado del espacio que se asigna al segmento inicial así como el tamaño y número de las extensiones subsiguientes que se realizarán cuando se agote el espacio. La gestión dinámica de las extensiones se puede hacer de forma global, las tablas de extensiones se almacenan en el diccionario de datos, o local, el espacio de tablas gestiona la asignación.

A continuación se muestra la sentencia utilizada para crear un espacio de tablas al que asigna inicialmente un fichero de 5 MB que puede crecer hasta 10 MB en incrementos de 1 MB. La gestión de las extensiones es local y automática.

```
CREATE TABLESPACE TABLAS
LOGGING
DATAFILE 'C:\oracle\oradata\orcl\tablas.dbf' SIZE 5M
AUTOEXTEND ON NEXT 1M MAXSIZE 10M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO
```

5.1.2 Bloques

Los bloques determinan la unidad de acceso básico a los datos para una base de datos Oracle. El tamaño por defecto se define al crear la base de datos mediante el parámetro de inicialización *DB_BLOCK_SIZE* y no se puede modificar tras crear la base de datos, es decir, requiere crearla de nuevo. Sin embargo, se pueden crear espacios de tablas con tamaños diferentes si en la memoria gestionada por Oracle (SGA) se definen subcachés adecuadas para cada uno de los nuevos tamaños de bloque. El tamaño del bloque de Oracle debe ser un múltiplo del tamaño del bloque utilizado por el sistema operativo y puede ser 2 K, 4 K, 8 K, 16 K o 32 K. Tamaños razonables son 4 K u 8 K aunque determinadas aplicaciones pueden obtener mejoras de rendimiento si este tamaño se ajusta adecuadamente. Por ejemplo, si es necesario almacenar filas muy grandes, objetos binarios grandes (BLOBs) o para bloques de índices puede ser interesante un tamaño de bloque mayor. Por otro lado, esto puede afectar al rendimiento si disminuye la concurrencia.

La Figura 5.2 muestra la estructura de un bloque Oracle, que incluye cabeceras de información de control y espacio destinado al almacenamiento. La cabecera tiene un tamaño mínimo fijo, pero por ejemplo, para el caso de las tablas implementa un directorio de filas almacenadas dentro del bloque cuyo tamaño depende del número de filas.

Es posible controlar el porcentaje de utilización del espacio de los bloques mediante el uso de los parámetros *PCTFREE* y *PCTUSED*. Estos parámetros sirven para implementar una política de gestión del espacio distribuido que facilita sobre todo que la actualización de una fila no provoque un desbordamiento del bloque que conllevaría una migración del almacenamiento

físico de la fila y una cascada de actualizaciones. El parámetro *PCTFREE* determina el porcentaje de espacio libre que se desea dentro de los bloques. Una vez rebasado este límite el bloque se considera lleno y no se realizan más inserciones. Sin embargo, queda espacio libre para posibles actualizaciones. Si la ocupación del área de datos desciende por debajo del porcentaje definido por *PCTUSED*, el bloque entra en la lista de bloques disponibles dentro del segmento para un objeto determinado.

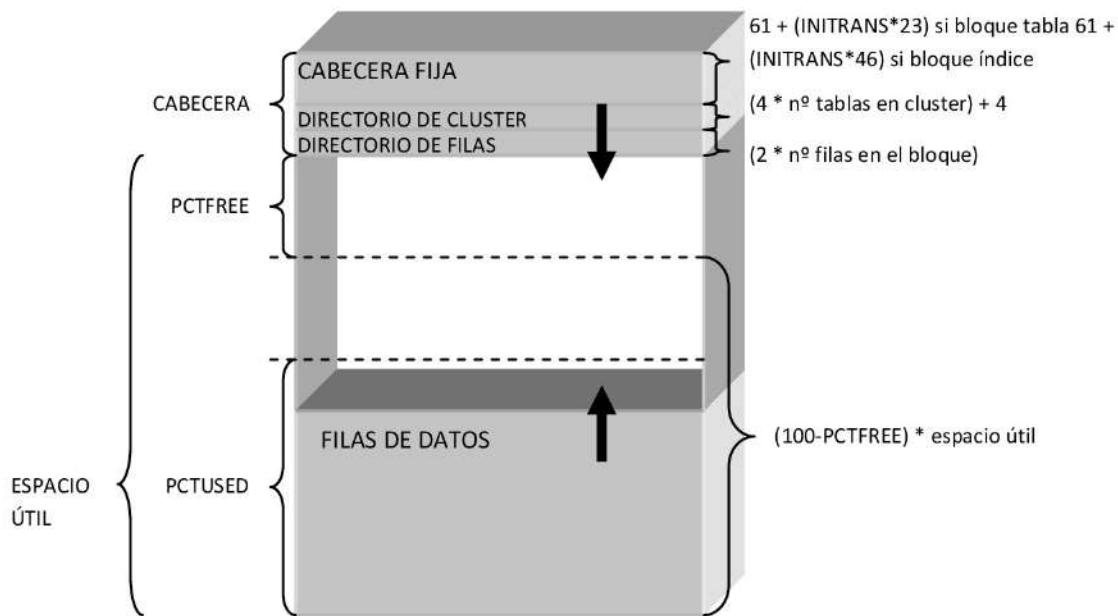


Figura 5.2. Esquema del bloque de datos Oracle

5.1.3 Tablas

Las tablas de Oracle se almacenan como el resto de objetos en los bloques, almacenando las tablas en montículo, es decir, la ubicación de la fila no depende de ninguno de los valores de la fila. Cada fila se identifica a partir de ese momento por un valor *ROWID* que es la posición física de la fila. Esta organización se podría considerar serial si no fuera por la excepción de que no se inserta siempre al final, sino en los bloques con espacio libre y si no existe ninguno es cuando se realiza una extensión del segmento de tabla. La estructura de un *ROWID* es OOOOOOFFBBBBBBRRR donde:

- OOOOOO es el número de un objeto de la base de datos, y sirve para identificar a los objetos que usan el mismo segmento.
- FFF es el número del archivo de datos.

- BBBB es el número del bloque de datos. El número del bloque es relativo al fichero de datos y no al espacio de tablas.
- RRR es el número de la fila dentro del directorio del bloque.

Cada fila se almacena dentro de un bloque, excepto si su tamaño excede el del bloque o tiene más de 256 columnas, ocasiones en las que las filas se encadenan entre varios bloques. Cada fila se compone de una cabecera y un espacio destinado a los datos. La cabecera ocupa al menos 3 bytes pero puede requerir información adicional dependiendo del tipo de tabla o para las filas encadenadas. Las columnas de datos usan 1 byte para indicar la longitud si la longitud de los datos es menor a 250 bytes y 3 en caso de que superen este límite. La Figura 5.3 muestra la estructura de una fila Oracle.

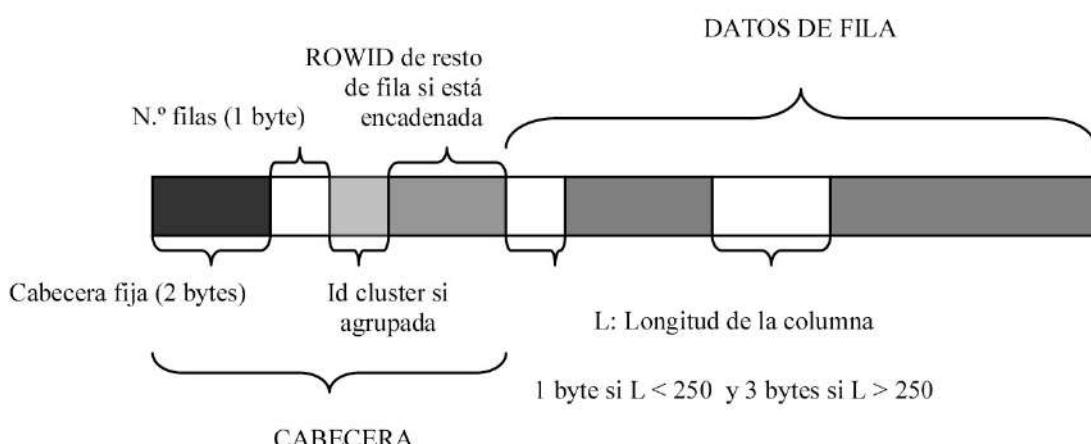


Figura 5.3. Formato de fila Oracle

La siguiente sentencia crea una tabla con dos columnas y especifica el espacio inicial del segmento y el resto de parámetros relativos a las extensiones a través de la cláusula *STORAGE*.

```
CREATE TABLE ejemplo (cod NUMBER(2), nombre VARCHAR2(14) )
STORAGE (
    INITIAL 100K NEXT 50K
    MINEXTENTS 1 MAXEXTENTS 50
    PCTINCREASE 5) ;
```

5.2 ESTRUCTURAS AUXILIARES

Oracle proporciona varios tipos de índices así como la posibilidad de añadir índices especializados llamados índices de dominio. El índice por defecto es un índice en árbol B, aunque en realidad se corresponde con lo que a nivel teórico y en el capítulo anterior se describía como arbol B+. Oracle también proporciona índices en mapa de bits, índices de listas invertidas y árboles R para aplicaciones de almacenes de datos, búsqueda textual y bases de datos espaciales.

5.2.1 Índices en árbol B+

Oracle genera automáticamente un índice en arbol B+ para las claves primarias y alternativas que se definen en una tabla. Los índices de las claves primarias se usan para facilitar la comprobación de la unicidad de las claves primarias y la existencia de claves ajenas.

Es importante reseñar algunas características particulares que permiten:

Indexar una o varias columnas y especificar el orden de las columnas. Sin embargo, para que las consultas hagan uso de los índices multicolumna, éstas deben usar las columnas indexadas en primer lugar. La estructura de la entrada de las hojas del índice es :

<col1><col2>...<colN><rowid>

Crear índices primarios (*UNIQUE*) y secundarios. Si algunas de las columnas que se indizan contienen valores *NULL*, cada uno de estos valores se considera un valor diferente. Es necesario tener en cuenta que las filas cuyas entradas se componen únicamente de valores nulos no se indizan. Esto implica que los índices no se usan para las consultas con condiciones de selección *IS NULL* o *IS NOT NULL*.

Permite crear índices cuyas entradas sean una función calculada a partir de una columna, siempre que esta función sea determinista. Un uso común de estos índices es la normalización de los valores de una columna a mayúsculas para su búsqueda.

Los índices secundarios almacenan a nivel hoja el valor de la entrada y el *ROWID* que sirve como apuntador al resto de la fila. Sin embargo, haciendo uso

de la opción *COMPRESS* al crear el índice, los nodos hoja del índice almacenan un prefijo que incluye las columnas comunes y un sufijo que representa las que son únicas. En el caso de los índices multicolumna, el prefijo es un subconjunto de las columnas que se han indizado. En los niveles superiores, no hoja, los índices almacenan tan solo el prefijo necesario para discriminar entre los valores del nodo inferior de la derecha y de la izquierda.

5.2.2 Índices en mapa de bits

Oracle también proporciona índices en mapa de bits que resultan interesantes para columnas con un número de valores diferentes moderado. Los índices utilizan tantas columnas como valores diferentes existan en las columnas que se indizan. También se indizan cada una de las filas para evitar tener que almacenar los punteros *ROWID* al preservar los índices el mismo orden que en la tabla de datos. Los índices pueden tener una gran cantidad de bits a cero por lo que adicionalmente se almacenan comprimidos mediante una técnica que reduce un gran número de ceros consecutivos a una codificación en únicamente un código de escape y el valor entero de esos ceros. Todas las operaciones se pueden realizar sobre la representación comprimida. Por último, como los índices en mapa de bits utilizan la misma estructura de almacenamiento en bloques, la estructura de la entrada cuando ésta supera el tamaño de un bloque es:

```
<col><id_filainicial><id_filafinal><mapa_de_bits_comprimido>
```

5.2.3 Tablas organizadas como índices

Las tablas organizadas como índices son tablas que en vez de estar organizadas como un montículo lo están como un índice en árbol B+. Este tipo de tablas necesitan una clave primaria y en el nivel hoja en vez de tener el *ROWID* hacia la fila, tienen el resto de valores de la fila. Sin embargo, puesto que las filas están sujetas a la reorganización dinámica propia de los índices, los índices secundarios sobre estas estructuras presentan peculiaridades. En este caso el apuntador no es un *ROWID* físico, sino lógico, lo que puede provocar que sean necesarios varios accesos para recuperar una fila a través de un índice secundario.

Este tipo de tablas está especialmente indicado para relaciones con filas pequeñas, en especial si el valor clave es proporcionalmente grande. En este caso

supone un ahorro de espacio y frecuentemente un ahorro en el número de accesos a disco. En cambio, si la tabla requiere el acceso por claves secundarias y sufre inserciones y borrados frecuentes, puede dejar de compensar.

5.2.4 Agrupaciones y agrupaciones asociativas

Una agrupación es otra alternativa para el almacenamiento de los datos de tabla. La agrupación almacena los datos de dos tablas diferentes combinados por el valor de una clave en el mismo bloque de datos. Esto es una característica interesante para tablas que frecuentemente se consultan a través de una operación de combinación. Por ejemplo, las tablas de departamento y empleados podrían agruparse si las consultas más frecuentes siempre se realizan sobre los subconjuntos de empleados de un departamento. Otra situación interesante cuando la relación es maestro-detalle, como en el caso de pedido y productos de un pedido. En este caso es interesante que ambos estén almacenados en el mismo bloque.

Como contrapartida, todas las consultas que impliquen solo una de las tablas van a sufrir una pérdida de eficiencia al tener que leer la agrupación de dos tablas.

La agrupación asociativa sigue la misma estructura pero además la posición física del grupo de filas viene determinada por el valor de la clave mediante un direccionamiento disperso. Esta organización está indicada para consultas que realizan una selección mediante un valor concreto. Sin embargo, el espacio requerido para una agrupación asociativa puede ser demasiado grande, en particular para tablas de tamaño considerable, puesto que la organización de la tabla implica tener bloques vacíos si la función de direccionamiento no transforma ninguna clave.

5.2.5 Partición de tablas

Oracle permite el particionamiento horizontal de tablas y de índices, característica que resulta de especial utilidad para base de datos muy grandes, bases de datos distribuidas y facilita la administración y transporte de datos entre entornos transaccionales y de almacen de datos.

La partición de tablas se puede realizar por rangos de valores, por listas de valores o mediante una función asociativa. Además es posible combinar varios

de estos criterios como por ejemplo la división por rangos y la asociativa. Sobre las particiones es posible definir índices globales, para todas las tablas, o locales, para una partición.

5.3 DISEÑO FÍSICO

El diseño físico de una base de datos implica el dimensionamiento y elección de las estructuras adecuadas para la materialización física de un esquema, pero este proceso debe tener en cuenta muchos otros aspectos, como las aplicaciones, consultas y frecuencias que se van a realizar sobre esa base de datos. El diseño físico debe ser especializado para un gestor, sistema operativo y plataforma hardware concreta y debe considerar aspectos claves para el rendimiento como la distribución de unidades de procesamiento y almacenamiento. Otro aspecto que es necesario considerar en el diseño físico es el diseño de mecanismos de seguridad de la base de datos. Es clave además de asignar permisos y roles a los usuarios, determinar cuáles son los requisitos de recuperación de la base de datos y cómo se van a cumplir.

El diseño físico toma un punto de vista preventivo sobre la gestión de una base de datos, anticipándose a las necesidades que su operación puede generar. Es clave que el administrador, que estará implicado en el proceso de diseño físico, realice pruebas y prototipos significativos para validar las decisiones tomadas. Por otro lado será necesario un enfoque complementario durante la operación de base de datos que asegure el buen funcionamiento y rendimiento de la base de datos, monitorizando y ajustando parámetros.

En ambos casos las herramientas de monitorización y ajuste de los diferentes aspectos de la base de datos juegan un papel especial. Oracle proporciona un conjunto de herramientas integrado en Oracle Enterprise Manager para la administración de bases de datos e instancias. Entre otras funcionalidades incluye herramientas para la obtención de estadísticas y el análisis y optimización de consultas. Por ejemplo, vistas del diccionario de datos como *DBA_TABLES*, *DBA_INDEXES* o *DBA_SEGMENTS* proporcionan información sobre los parámetros y características físicas de la base de datos. Los paquetes *DBMS_STAT* de Oracle permiten la elaboración de estadísticas que mediante el uso de *EXPLAIN PLAN* o de herramientas como SQL Tuning Advisor pueden utilizarse en el proceso de optimización de planes de ejecución para las consultas.

5.3.1 Selección de índices

A continuación se presentan algunos heurísticos para la elección de índices, especialmente secundarios ya que Oracle se encarga de crear índices para las claves primarias y alternativas.

No conviene indizar tablas pequeñas, pues el número de accesos a la tabla completa no será mucho mayor que por el índice pero se incurre en un coste de mantenimiento.

Si se accede frecuentemente a través de una clave ajena o se utiliza frecuentemente la combinación con la tabla que define la clave como primaria se debe considerar un índice por la clave ajena. Este índice puede hacer que el coste de realizar una combinación usando solo los índices sea ventajoso.

Se debe considerar realizar índices por las columnas que se usan frecuentemente en las condiciones *WHERE*, *ORDER BY* o *GROUP BY* u otras operaciones que implican ordenación como *UNION* o *DISTINCT*. La utilidad de estos índices depende de la cantidad de filas o tuplas que debamos mostrar en el resultado.

En ocasiones hay consultas que se pueden realizar accediendo solo a un índice, en particular, para los casos en los que el índice se compone de varias columnas.

No conviene indexar columnas que se modifiquen frecuentemente o que tengan una longitud muy grande.

En el caso de que la columna tenga pocos valores se debe valorar el uso de un índice en mapa de bits.

PROBLEMA 5.1: ESTIMACIÓN DEL TAMAÑO DE UNA BASE DE DATOS

En el Problema 1.1 del Capítulo 1 se propuso un esquema para la base de datos de la Sociedad Pública de Alquiler que ahora se desea implementar en Oracle. El esquema lógico usando el modelo relacional se presenta en la Tabla 5.1. Teniendo en cuenta las características de almacenamiento de Oracle se pide realizar una primera estimación del tamaño de la base de datos inicial. Para ello tendremos en cuenta las siguientes simplificaciones:

- Todas las tablas se almacenan en un mismo espacio de tablas definido con un tamaño de bloque 4 K.
- El parámetro *PCTFREE* se fija inicialmente a 0, es decir se tratará de ocupar todo el espacio disponible en el bloque.
- Para los índices utilizaremos *PCTFREE* = 33.
- Consideramos que todos los bloques tienen una cabecera de 86 bytes.
- La cabecera de fila usa 3 bytes.
- Las filas de longitud inferior a 250 bytes usan 1 byte de marca de longitud y las de longitud superior a 250 bytes usan 3 bytes.

La Tabla 5.1 resume las características de los principales dominios.

La Tabla 5.2 presenta las estimaciones de número de filas para cada una de las tablas de la base de datos.

NOMBRE DE DOMINIO	CARÁCTER	TAMAÑO MÁXIMO	TAMAÑO MEDIO
APELLIDOS	VAR	50	20,3
CALLE	VAR	50	35,2
CIF	FIJO	9	9
CODIGO_AGENCIA	FIJO	8	8
CODIGO_ALQUILER	FIJO	10	10
CODIGO_VIVIENDA	FIJO	10	10
CP	FIJO	5	5
DESCRIPCION_VIVIENDA	VAR	2048	315
FECHA	FIJO	8	8
IMPORTE	VAR	5	3,2
NIF	FIJO	8	8
NOMBRE	VAR	20	9,5
NOMBRE_AGENCIA	VAR	30	7,2
NUMERO	VAR	3	1,3
PISO	VAR	2	1,1
POBLACIÓN	VAR	20	10,3
PREFERENCIAS	VAR	2048	250
TELEFONO	FIJO	9	9
TIPO	ENUM	12	8,4

Tabla 5.1. Dominios y estimación de tamaños en bytes (B)

TABLA	NÚMERO DE FILAS
VIVIENDA	200.000
PROPIETARIO	60.000
AGENCIA	2.500
ALQUILER	1.000.000
INQUILINO	600.000
RENOVACION	30.000

Tabla 5.2. Estimación de número de filas

DISCUSIÓN DEL ENUNCIADO

Para estimar el tamaño de la base de datos debemos calcular tanto el tamaño de las tablas como las de los índices que Oracle crea automáticamente, en especial los índices para las claves primarias. Se realizan los cálculos para la tabla VIVIENDA.

Tamaño de la tabla VIVIENDA:

La tabla VIVIENDA se almacena en el espacio de tablas en un segmento propio. El espacio de tablas determina el tamaño de bloque (T_{bloque}), en nuestro caso 4 K. Podemos estimar el volumen de la tabla (V_{tabla}) mediante la siguiente fórmula:

$$\left[V_{tabla} = N_{bloques} * T_{bloque} \quad \text{donde } N_{bloques} \text{ es el número de bloques} \right]$$

El número de bloques es el producto del número de filas y el factor de bloqueo medio FB . Para determinar el tamaño medio de las filas debemos asignar tipos adecuados a cada una de las columnas de la tabla y recoger estimaciones del tamaño para los valores.

$$\left. \begin{aligned} N_{bloques} &= \overline{FB} * N_{filas} \\ \overline{FB} &= \left\lfloor \frac{EU_{bloque}}{\overline{T}_{fila}} \right\rfloor \\ \text{donde } EU_{bloque} &\text{ es el espacio útil dentro del bloque} \end{aligned} \right\}$$

El cálculo del tamaño de una fila requiere tener datos y cabecera de fila. Para el cálculo del tamaño medio de una columna debemos tener en cuenta si se trata de una columna de tamaño fijo (por ejemplo, *CHAR*) o de tamaño variable (*VARCHAR2*) ya que su almacenamiento es distinto. La documentación de Oracle nos dice que las marcas ocupan un byte para los campos de longitud menor a 250 y 3 bytes para los de longitud superior.

Por último, será necesario tener en cuenta que el espacio de un bloque se reparte entre cabecera y espacio de datos. Un segundo factor son los parámetros para la gestión del espacio libre distribuido (*PCTFREE/PCTUSED*) que limitan el espacio útil.

$$\left. \begin{array}{l} T_{bloque} = C_{bloque} + D_{bloque} = C_{bloque} + EL_{bloque} + EU_{bloque} \\ \text{donde } C_{bloque} \text{ es el tamaño de la cabecera de bloque} \\ \\ EL_{bloque} = \frac{PCTFREE * D_{bloque}}{100} \\ \\ EU_{bloque} = \frac{(100 - PCTFREE) * D_{bloque}}{100} \end{array} \right\}$$

Para la tabla VIVIENDA, los tipos de cada uno de los campos junto con la estimación de la longitud media de la fila se presentan en la Tabla 5.3.

COLUMNA	TIPO	MARCA (Bytes)	DATOS (Bytes)	
CÓDIGO_VIVIENDA	CHAR(10)	0	10	
CALLE	VARCHAR2(50)	1	35,2	
NÚMERO	VARCHAR2(3)	1	1,3	
PISO	VARCHAR2(2)	1	1,1	
CP	CHAR(5)	0	5	
POBLACIÓN	VARCHAR2(20)	1	10,3	
TIPO	VARCHAR2(12)	1	8,4	
DESCRIPCIÓN	VARCHAR2(2048)	3	315	
NIF_PROPIETARIO	CHAR(8)	0	8	
CIF_AGENCIA	CHAR(9)	0	9	TOTAL
	FILA MEDIA	8	403,3	411,3

Tabla 5.3. Tabla VIVIENDA

$$\overline{T_{fila}}(VIVIENDA) = 3 + 411,3 = 414,3B$$

$$EU_{bloque} = 4096 - 86 = 4010 B$$

$$\overline{FB} = \left\lfloor \frac{EU_{bloque}}{\overline{T_{fila}}} \right\rfloor = \left\lfloor \frac{4010B}{414,3B} \right\rfloor = \left\lfloor 9,68 \right\rfloor = 9 \text{ filas por bloque.}$$

Puesto que tendremos 9 filas por bloque, podemos calcular el número de bloques que necesitaremos para 200.000 filas. El volumen necesario para los datos de la tabla será de 86,80 MB.

$$N_{bloque} = \left\lceil \frac{200.000 \text{ filas}}{9 \frac{\text{filas}}{\text{bloque}}} \right\rceil = 22.223 \text{ bloques.}$$

$$V_{tabla}(VIVIENDA) = 22.223 * 4096 B = 86,80 MB$$

Tamaño del índice primario PK_VIVIENDA:

Oracle genera índices automáticamente para las claves primarias y alternativas que se definen en una tabla. Estos índices se usan tanto en la combinación como al comprobar las restricciones de clave primaria, alternativa y ajena. El tipo de índice por defecto que se genera es del tipo árbol B+. En el caso de las claves primarias este índice, como es lógico, no tiene claves repetidas (*UNIQUE*).

Las entradas de un índice en arbol B+ único están siempre representadas en las hojas. La entrada se compone del valor que se indiza y un puntero al resto de la fila, para lo que se utiliza el *ROWID*. Puesto que el objeto es siempre el mismo para este tipo de índices suponemos que el *ROWID* solo se compone de estos tres componentes: *FICHERO*, *BLOQUE* y *FILA* y por lo tanto ocupa 12 bytes.

$$\left[T_{entrada_hoja} = T_{clave} + T_{rowid_ext} = 10B + 12B = 22B \right]$$

Para los punteros internos al árbol vamos a suponer que la *FILA* no es un valor necesario y por lo tanto el puntero es de tan solo 9 bytes. Oracle almacena solo un prefijo separador en los nodos no hoja. Puesto que el tamaño de este prefijo depende de los valores y su distribución usaremos para la estimación el tamaño de la entrada y por lo tanto obtendremos una cota superior para el número de bloques en los niveles internos y altura del árbol.

$$\left[T_{\text{entrada_no_hoja_max}} = T_{\text{clave}} + T_{\text{rowid_int}} = 10B + 9B = 19B \right]$$

Para calcular el tamaño de los índices vamos a tomar que el espacio libre distribuido de los bloques es 1/3 del total (*PCTFREE* = 33) de forma análoga a los desarrollos teóricos, aunque este parámetro puede ser configurado por el administrador también para los índices. De esta forma el espacio útil de los bloques de índices:

$$EU_{\text{bloque}} = \frac{(100 - \text{PCTFREE}) * D_{\text{bloque}}}{100} = \frac{(100 - 33) * 4.010}{100} = 2.686,7$$

Para calcular el número de bloques hoja, restaremos en primer lugar el espacio ocupado por los punteros internos que enlazan estos para el recorrido secuencial. El número de bloques se estima de forma semejante a las tablas.

$$EU_{\text{bloque_nodo_hoja}} = 2.686,7 - 2 * T_{\text{rowid_int}} = 2.668,7B$$

$$\overline{FB}_{\text{nodo_hoja}} = \left\lfloor \frac{EU_{\text{bloque_nodo_hoja}}}{T_{\text{entrada_hoja}}} \right\rfloor = \left\lfloor \frac{2.668,7B}{22B} \right\rfloor = 121 \text{ entradas}$$

$$N_{\text{nodo_hoja}} = \left\lfloor \frac{200.000 \text{ entradas}}{121 \frac{\text{entradas}}{\text{bloque}}} \right\rfloor = 1.653$$

Para el segundo nivel tendremos por tanto $N_{\text{bloque}} - 1$ 1.652 entradas separadoras que almacenar. En este caso no tenemos punteros internos para el recorrido secuencial.

$$\overline{FB_{nodo_no_hoja}} = \left\lfloor \frac{EU_{bloque_nodo_hoja}}{T_{entrada_hoja}} \right\rfloor = \left\lfloor \frac{2.686,7B}{19B} \right\rfloor = 140$$

$$N_{nodo_no_hoja}^2 = \left\lceil \frac{1.652 \text{ entradas}}{\frac{140 \text{ entradas}}{\text{bloque}}} \right\rceil = 12$$

Por lo tanto necesitamos como máximo 12 bloques para almacenar el segundo nivel del árbol y otro bloque para el tercer nivel

$$V_{indice} = N_{bloques} * T_{bloque} = (1.653 + 140 + 1) * 4.096 B = 6,51 MB$$

El volumen total de la tabla incluye ambas estructuras y es $V = 86,81 MB + 6,51 MB = 93,52 MB$. La Tabla 5.4 presenta las mismas estimaciones para todas las tablas de la base de datos.

TABLA	Tamaño (MB)	Índice PK (MB)	Tamaño Total (MB)
VIVIENDA	86,81	6,51	93,32
PROPIETARIO	3,01	1,78	4,79
AGENCIA	0,22	0,09	0,31
ALQUILER	63,01	32,52	95,52
INQUILINO	195,31	17,74	227,83
RENOVACION	0,73	0,98	1,71
	349,09	59,62	423,49

Tabla 5.4. Volumen de tablas

De esta forma estimamos que la base de datos tiene un volumen de 423,49 MB.

PROBLEMA 5.2: ESTIMACIÓN DEL VOLUMEN DE UNA TABLA

Tras la inserción de una nueva vivienda en la base de datos de la Sociedad Pública de Alquiler se ha comprobado que la descripción se suele modificar en el periodo de 1 mes. Se ha estimado que el 20% de las filas de *VIVIENDA* se ven afectadas y que en esos casos la longitud del campo *DESCRIPCIÓN* se dobla. Se pide calcular el espacio libre distribuido (*PCTFREE*) con el que se ha de crear la tabla para evitar filas encadenadas para la inserción inicial. ¿Cuál será el volumen de la tabla en este caso? ¿Y el del índice asociado a la clave primaria?

DISCUSIÓN DEL ENUNCIADO

Para el 20% de las filas de vivienda el tamaño aumenta 315 B de media debido a la modificación. Deberemos dejar sin ocupar un espacio equivalente de forma que las filas se puedan acomodar tras la inserción. Es imposible garantizar que no habrá filas encadenadas, pues tanto el tamaño como el porcentaje se trata de promedios, pero sí que de esta forma reducimos la probabilidad. La siguiente inecuación representa el caso medio, donde p representa el porcentaje de filas que se modifican. Usando el tamaño medio de la fila VIVIENDA calculado en el Problema 5.1.

$$EU_{bloque} > \left(\overline{T}_{fila}^{inicial} * (1 - p) + \overline{T}_{fila}^{1mes} p \right) * \overline{FB}$$

$$4.010 > (414,3 * 0,8 + (414,3 + 315) * 0,2) * \overline{FB}$$

$$\overline{FB} < 8,40$$

Teniendo cuenta el crecimiento podríamos almacenar hasta 8,40 filas de media en cada bloque. Llegado este punto tenemos dos alternativas:

Podemos estimar el espacio medio libre que debería dejar dentro del bloque para calcular el espacio libre medio necesario. Puesto que $PCTFREE$ solo toma valores enteros tomaremos $PCTFREE = 14$ que garantiza que en el caso medio no se produce encadenamiento:

$$\overline{EL}_{bloque} = 8,40 * 0,2 * 315B = 529,2B$$

$$EL_{bloque} = \frac{PCTFREE * D_{bloque}}{100} = \frac{PCTFREE * 4010}{100}$$

$$PCTFREE = \frac{100 * 529,2B}{4010B} = 13,19$$

Podemos realizar una estimación algo más conservadora y reservar espacio para al menos 2 filas que aumentan el tamaño medio. En este caso el parámetro

$PCTFREE = 16$.

$$EL_{bloque} = 2 * 315B = 630B$$

$$PCTFREE = \frac{100 * 630B}{4.010B} = 15,71$$

Tomando $PCTFREE = 14$, el espacio inicial disponible para datos dentro de un bloque se reduce a 3.448,6 B y el factor de bloqueo para los registros iniciales es 8,32, es decir de 8 filas por bloque. Usaremos 25.000 bloques y el volumen de la tabla se estima que asciende a 97,65 MB.

A pesar de que la tabla crece, la información a indexar es la misma, ya que descripción no forma parte de la clave, por lo que el tamaño del índice no debería variar.

PROBLEMA 5.3: CREACIÓN DE ÍNDICES

Deseamos crear un índice para el campo *TIPO* de la tabla *VIVIENDA* para acelerar la consulta “Importe medio de los alquileres de viviendas de un tipo determinado”. Justifica si el índice debe ser un árbol B+ o un índice en mapa de bits. Estima cuántos valores diferentes podría tener el dominio *TIPO* para que la opción óptima sea mantener un mapa de bits para esta consulta. Para simplificar los cálculos asume que el campo tiene una longitud fija de 12 bytes, que la distribución de los valores de *TIPO* para la tabla es uniforme y que los índices no usan compresión.

La consulta propuesta se traduce en la siguiente sentencia SQL:

```
SELECT AVG(IMPORTE)
FROM VIVIENDA JOIN ALQUILER ON COD_VIVIENDA
WHERE TIPO = 'X'
```

DISCUSIÓN DEL ENUNCIADO

Esta consulta se puede expresar mediante Álgebra Relacional de numerosas formas, aunque la más directa es la siguiente:

$$AVG\left(\Pi_{IMPORTE}\left(\sigma_{TIP0='X'}\left(VIVIENDA \underset{COD_VIVIENDA}{*} \Pi_{ALQUILER}\right)\right)\right)$$

Podemos avanzar que el optimizador de consultas realizará diferentes operaciones como el tratar de proyectar y seleccionar cuanto antes para reducir el volumen de los subresultados. Las siguientes expresiones son equivalentes, pero proporcionarán siempre un plan lógico más óptimo.

$$AVG\left(\sigma_{TIP0='X'}\left(VIVIENDA \underset{COD_VIVIENDA}{*} \Pi_{ALQUILER}\right)\right)$$

$$AVG\left(\sigma_{TIP0='X'}\left(\Pi_{TIPO, COD_VIVIENDA} VIVIENDA\right)_{COD_VIVIENDA} * \Pi_{ALQUILER}\right)$$

La operación que se ve afectada por la elección del índice es la selección mediante tipo que trabaja sobre la proyección de la tabla VIVIENDA. Para estimar cuál de los índices beneficiaría más a esta consulta realizamos una estimación del volumen de los índices así como del número de accesos necesarios para esta operación.

Arbol B+:

La columna TIPO no es única, por lo que la estructura de las entradas de las hojas del arbol será:

<valor_entrada><longitud_punteros><rowid>...<rowid>

$$\left[T_{entrada} = T_{clave} + \lceil \log_2 L_{lista} \rceil + L_{lista} * T_{ROWID_externo} \right]$$

Puesto que la distribución de valores se considera uniforme la longitud de la lista o el número de filas que toman ese valor se puede estimar en función del

número de valores n como:

$$\left[L_{lista} = \frac{N_{filas}}{n} \right]$$

Para un número grande de L_{lista} o pequeño de n la entrada podría necesitar varios bloques de datos. El número de bloques hoja para una clave se puede estimar como:

$$N_{bloques_hoja} = \frac{T_{entrada}}{EU_{bloque}} = \frac{12 + \lceil \log_2 L_{lista} \rceil + L_{lista} * 12}{4010} \approx \frac{(L_{lista} + 1) * 12}{4010}$$

$$N_{bloques_hoja} \approx \frac{(200.000 + n) * 12}{4010n}$$

Puesto que n se supone relativamente pequeño podemos asumir sin más cálculos que el árbol solo tendrá un nivel adicional. Para el cálculo del número de accesos podemos estimar que se realiza un acceso al nivel superior y tantos como sean necesarios para leer los bloques de la hoja. El caso peor (y más común) exige al menos acceder al final de la entrada anterior y el principio de la siguiente.

$$N_{accesos} = N_{accesos_hoja} + N_{accesos_no_hoja} = \frac{(200.000 + n) * 12}{4.010 * n} + 2 + 1$$

Mapa de bits:

La estructura del mapa de bits de Oracle indica con un bit la presencia o ausencia de fila. Esta estructura se repite para cada bloque del índice.

<valor_entrada><rowid_inicio><rowid_fin><mapa_de_bits>

$$T_{\text{entrada}} = T_{\text{clave}} + 2 * T_{\text{ROWID}} + T_{\text{MAPA_BITS}} \approx T_{\text{MAPA_BITS}}$$

$$T_{\text{MAPA_BITS}} = \frac{N_{\text{filas}} * n}{8\text{bits}} = \frac{200.000 * n}{8} = 25.000n \text{ B}$$

$$N_{\text{bloques}} = \frac{T_{\text{MAPA_BITS}}}{EU_{\text{bloque}}} = \frac{2.5000 * n}{4.010} = 6,23n$$

$$N_{\text{accesos}} = 6,23n$$

A continuación debemos comparar el modelo de coste de acceso para ambas estructuras:

$$N_{\text{accesos}}^{\text{mapa_bits}} < N_{\text{accesos}}^{B+}$$

$$6,23n < \frac{(200.000 + n) * 12}{4010 * n} + 2 + 1$$

Resolviendo la ecuación de segundo grado obtenemos un valor $n < 9,56$ lo que nos indica que la opción de un mapa de bits es la más adecuada si *TIPO* tiene pocos valores, en particular 9 o menos. Con las características de compresión de Oracle éste sería el índice más adecuado incluso para valores algo superiores.

PROBLEMA 5.4: SELECCIÓN DE ÍNDICES EN UNA BASE DE DATOS

Sobre la base de la Sociedad Pública de Alquiler se han considerado las siguientes transacciones como las más frecuentes:

- A. Insertar los detalles de un nuevo inmueble, junto con los de su propietario y agencia si fuese necesario.
- B. Insertar los detalles de un nuevo contrato y un nuevo inquilino.
- C. Insertar los detalles de un nuevo contrato que es renovación del anterior.
- D. Modificar los detalles de una vivienda para ampliar la descripción.
- E. Mostrar viviendas libres en una zona (CP) y rango de fechas determinados.
- F. Mostrar viviendas libres ordenadas por el importe mensual del último alquiler.
- G. Mostrar alquileres que acaban en menos de 2 meses de la fecha actual.
- H. Mostrar viviendas de un propietario identificado por Nombre y Apellidos.
- I. Mostrar histórico de contratos de alquiler para un inquilino identificado por Nombre y Apellidos.
- J. Mostrar histórico de contratos para una vivienda identificada por su Dirección.

Además, será necesario tener en cuenta las siguientes restricciones de integridad añadidas al modelo ER:

1. Se debe asegurar que no existen solapamientos entre los periodos de alquiler de una vivienda.
2. Se debe asegurar que un inquilino no tiene dos alquileres al mismo tiempo.

3. Un alquiler no debe ser renovación de sí mismo.

Se pide discutir qué índices adicionales serían necesarios crear para realizar estas transacciones.

Para analizar el uso de tablas por cada una de las transacciones vamos a realizar un mapa de transacciones y analizaremos qué tablas se ven implicadas y para qué operaciones inserción (I), localización (L), modificación (M) y borrado (B). La Tabla 5.5 es un resumen de este mapa de transacciones. Se han omitido las columnas que no tienen sentido para alguna de las transacciones para simplificar la representación. Además, aparecen en gris las operaciones que se verían ralentizadas por el uso de índices.

TABLA	A I	B L	C I	D L	E M	F L	G L	H L	I L	J L
VIVIENDA	X				X X	X X		X		X
PROPIETARIO	X							X		
AGENCIA	X									
ALQUILER		X X	X X			X X X			X X	
INQUILINO		X	X						X	
RENOVACIÓN			X X							

Tabla 5.5. Mapa de transacciones y tablas

DISCUSIÓN DEL ENUNCIADO

Las transacciones de inserción *B* y *C* tienen que consultar otras tablas (*ALQUILER*, *INQUILINO* y *RENOVACIÓN*) debido a las restricciones definidas. Como se extrae de la Tabla 5.5 las tablas más consultadas son precisamente *ALQUILER* y *VIVIENDA*, así que nos centraremos primero en ellas.

La tabla *ALQUILER* se usa en las transacciones *B* y *C* mediante las restricciones que hacen uso de las columnas *FECHA_INICIO* y *FECHA_FIN*. La transacción *E* tambien usa estas fechas mediante la especificación de un rango. Del mismo modo, la consulta *G* debe hacer comparaciones con el campo *FECHA_FIN*. Deberíamos considerar el uso de índices para los atributos *FECHA_INICIO* y *FECHA_FIN*, incluso combinados, ya que las consultas de rango hacen uso de ambas columnas. Puesto que *G* usa tan solo la *FECHA_FIN* se debe valorar o bien usar esta columna como primer atributo combinado o la posibilidad de usar un nuevo índice para esta columna.

Por otro lado, la consulta *F* hace uso de la *FECHA_FIRMA* para determinar el último contrato. Esta misma columna se usa en las consultas *I* y *J* para ordenar. Se debería considerar la posibilidad de crear un índice para *FECHA_FIRMA*. Además, puesto que la tabla está sujeta a inserciones frecuentes puede no ser adecuado, en especial si los listados ordenados de *I* y *J* contienen pocas filas.

La tabla *VIVIENDA* se usa en varias consultas (*E* y *F*) junto a la tabla *ALQUILER*. Como es la clave de *COD_VIVIENDA* la que se usa para hacer la combinación podría resultar interesante indizar esta columna en la tabla *ALQUILER*. El campo *CP* se usa en las consultas *E* y *J* (implícitamente a través de la dirección) por lo que es un buen candidato para otro índice. *PROPIETARIO* e *INQUILINO* se consultan mediante *NOMBRE* y *APELLIDOS*, dependiendo del tamaño de tabla e índice tenemos otros dos índices candidatos.

Finalmente, la tabla *INQUILINO* y *ALQUILER* se combinan frecuentemente y un último índice candidato es el DNI del inquilino en *ALQUILER*.

Capítulo 6

CASOS PRÁCTICOS: DISEÑO E IMPLEMENTACIÓN EN EL SGBD ORACLE

Como último capítulo, se van exponer dos casos prácticos para el diseño y desarrollo de una base de datos. Para el diseño se seguirá la metodología vista a lo largo del libro (Capítulos 1 y 2), y se llevarán a cabo todas las fases del modelado: esquema Entidad-Interrelación (E/R) y Relacional antes de pasar a la implementación en el Sistema Gestor de Base de Datos (SGBD). El capítulo presenta los dos casos prácticos y cada uno está estructurado en tres bloques. En el primero se verá el diseño conceptual a partir de los supuestos semánticos de un enunciado dado de acuerdo al modelo E/R. En el bloque dos pasaremos al diseño lógico transformando el esquema E/R a un diagrama relacional que junto con sus restricciones semánticas darán lugar al esquema relacional. A partir de éste pasaremos al tercer bloque, donde se introduce el diseño lógico específico proporcionando la implementación en el SGBD ORACLE en el material adicional del libro.

6.1 CASO PRÁCTICO 1: GESTIÓN FONDOS DE UN MUSEO

Se desea realizar una base de datos para un museo o pinacoteca. Este museo tiene sus obras o cuadros distribuidos entre un fondo permanente y una exposición temporal. El fondo permanente está expuesto en tres plantas y cada planta tiene varias salas organizadas según el orden cronológico de las obras que alberga. Un estilo pictórico se caracteriza por un nombre y el periodo que abarca (fecha de inicio y fecha de fin) y puede comprender varias salas.

Las plantas se identifican por su número de piso y hay que almacenar el número de salas que comprende. De las salas hay que guardar su código (que será un código con el número de planta a la que pertenece y el número de sala), obras que alberga y número. De los cuadros interesa almacenar el pintor autor del cuadro, características técnicas del cuadro (óleo, temple, acrílico, esmalte), periodicidad recomendable para revisión por el departamento de restauración, fecha y título. De cada pintor se almacenará su nombre real si se conociera,

artístico, nacionalidad, estilos a los que ha pertenecido y las fechas de nacimiento y fallecimiento.

Del buen estado de los cuadros se encargan los departamentos de conservación y restauración. Los conservadores y restauradores se identifican por un código y un nombre.

Los conservadores son los responsables del perfecto estado de los cuadros, de tal forma que hay un conservador encargado de las obras de un mismo estilo pictórico. Así, por ejemplo, habrá un conservador encargado de las obras de estilo renacentista, otro de las obras impresionistas, etc. Un conservador solo puede ser responsable de un estilo pictórico.

Los restauradores son los que se encargan de restaurar los cuadros; se hacen revisiones periódicas dependiendo del tipo de cuadro que sea. Los restauradores están especializados por pintores, es decir, un restaurador será el encargado de restaurar los cuadros de uno o varios pintores determinados. Así, habrá un restaurador para el Greco, otro de Goya, otro de Sysley y de Degas, etc. Hay que tener guardado en la base de datos el periodo de restauración de un cuadro (fecha de inicio y fecha de fin) y qué restaurador se encarga del trabajo. Un restaurador puede estar trabajando en más de una obra al mismo tiempo.

Los cuadros del fondo permanente pueden estar prestados a otros museos o a la exposición temporal del mismo museo. Hay que guardar la información de a quién se realiza el préstamo, cuáles son los datos del museo (nombre, dirección y teléfonos). Habrá que controlar dónde se encuentran en cada momento los cuadros propiedad del museo, si están en las salas del fondo permanente, exposición temporal, prestados o restaurándose.

La exposición temporal del museo se caracteriza por un título y sus fechas. Hay que almacenar el conservador encargado de la exposición, cuáles y el número de cuadros que alberga. La exposición temporal estará compuesta por obras del fondo permanente del museo, más las obras prestadas por otras instituciones o colecionistas particulares.

Un restaurador podrá estar trabajando en un máximo de tres obras y solo se debe permitir que tenga en cola de espera para restauración un cuadro.

Cuando haya que restaurar un cuadro hay que asignar ese trabajo a un restaurador que le pertenezca y habrá que establecer una fecha próxima prevista para inicio de restauración; si el restaurador indicado estuviera libre la fecha sería la del día, empezando ese mismo día el trabajo de restauración. Si, en

cambio, estuviera ocupado, es decir, está restaurando tres obras en ese momento, la fecha de inicio de la restauración de la obra sería la primera fecha de finalización de restauración de alguna de las obras en las que está trabajando.

6.1.1 Diseño conceptual: Esquema E/R

En este bloque se va seguir la dinámica que se ha visto en anteriores capítulos: se desglosará el enunciado en grupos de supuestos, se analizarán y se realizarán subdiagramas E/R hasta llegar a un diagrama E/R final de todo el enunciado, que unido con los supuestos semánticos no contemplados en el diagrama darán como resultado el esquema E/R.

Para dar un enfoque de caso práctico a este capítulo, y que el lector tome el rol de diseñador de una forma más activa, se han enunciado los supuestos tomados por el diseñador y los supuestos incompletos o ambiguos en el enunciado de los que necesita más información del dominio. Para completar y aclarar estos supuestos adicionales detectados y otras necesidades de identificación de requisitos que surjan, se ha simulado la interacción que el diseñador de una base de datos tiene con usuarios y expertos del dominio en cuestión. Así, se ha realizado una iteración más en el diseño con esos nuevos supuestos, obteniendo un incremento de semántica en los subdiagramas E/R.

Pasamos a hacer el esquema E/R:

DISCUSIÓN DEL ENUNCIADO

PARTE 1

Se desea realizar una base de datos para un museo o pinacoteca. Este museo tiene sus obras o cuadros distribuidos entre un fondo permanente y una exposición temporal. El fondo permanente está expuesto en tres plantas, y cada planta tiene varias salas organizadas según el orden cronológico de las obras que alberga.

Un estilo pictórico se caracteriza por un nombre y el periodo que abarca (fecha de inicio y fecha de fin) y puede comprender varias salas.

Las plantas se identifican por su número de piso y hay que almacenar el número de salas que comprende. De las salas, hay que guardar su código (que será un código con el número de planta a la que pertenece y el número de sala), obras que alberga y número. De los cuadros, interesa almacenar el pintor autor del cuadro, características técnicas del cuadro (óleo, temple, acrílico, esmalte), periodicidad recomendable para revisión por el departamento de restauración, fecha y título. De cada pintor se almacenará su nombre real si se conociera, artístico, nacionalidad, estilos a los que ha pertenecido y las fechas de nacimiento y fallecimiento.

Se identifican como entidades: *ESTILO_PICTÓRICO*, *PLANTA*, *SALA*, *CUADRO* y *PINTOR*. Veamos cuáles son los atributos de cada una de ellas.

La entidad *ESTILO_PICTÓRICO* tiene como atributos el *Nombre* identificador principal (AIP) y el *Periodo*, que será un atributo compuesto que indica el periodo de tiempo de un estilo con una *Fecha_ini* y una *Fecha_fin*, este último atributo opcional para contemplar en la base de datos estilos pictóricos contemporáneos.

La entidad *PLANTA* tiene como atributo a *Número AIP* y a *Num_salas* que será un atributo derivado (D1), y su valor, semántica no reflejada en el diagrama E/R, será el calculo del número de salas que tiene cada planta, que se podrá deducir a partir de la interrelación **Contiene** entre *PLANTA* y *SALA*.

Como atributos de la entidad *SALA* están: *Cod_sala* AIP que será un código con el número de planta al que pertenece y el número de sala, por esta fuerte

dependencia se considerará la relación en identificación **Contiene** entre *PLANTA* (entidad fuerte) y *SALA* (entidad débil). Además, *SALA* tiene el atributo derivado (D2) *Num_cuadros*, que será calculado a partir del número de cuadros que tiene cada sala y se podrá deducir a partir de la interrelación **Expone**, entre *SALA* y *CUADRO*. Esta semántica no está reflejada en el diagrama. Cómo diseñadores nos cuestionamos si hay que controlar que hay un orden cronológico de las salas.

La entidad *CUADRO* tiene el atributo *Técnica* que tomará los valores: óleo, temple, acrílico o esmalte, y los atributos *Per_restauración*, *Fecha* y *Titulo*, pero queda sin determinar por qué se identifica un cuadro. La entidad *PINTOR* tiene a *Nombre* como atributo opcional y, además, los atributos: *Nom_artístico*, *Nacionalidad* y un atributo *Periodo* compuesto de los atributos *Fecha_nac*, que almacenará la fecha de nacimiento, y el atributo *Fecha_fall*, que guardará la fecha de fallecimiento. Este último atributo será opcional, ya que puede ser que el pintor sea contemporáneo y no haya fallecido aún. En los supuestos no se indica por qué se identificará un pintor.

Como interrelaciones tenemos la ya nombrada **Expone** entre las entidades *SALA* y *CUADRO* con cardinalidad N:M. Teniendo en cuenta que los cuadros del museo se exponen entre el fondo permanente y la exposición temporal, puede que haya cuadros que no estén asignados en ninguna sala del fondo permanente. Por esta razón habrá que preguntarse si puede haber salas vacías de cuadros, o una sala tiene una capacidad máxima de cuadros.

La interrelación **Pinta** asocia las entidades *CUADRO* y *PINTOR*, y según la semántica de los supuestos se sabe solo parte de la cardinalidad. Se dice que un pintor puede haber pintado varias obras, pero no se determina si puede haber cuadros anónimos, y si un cuadro puede ser pintado por más de un pintor.

Por último, están las entidades *ESTILO_PICTÓRICO* y *CUADRO* asociadas por la interrelación **Es_de**, donde la cardinalidad no está del todo explicitada, ya que los supuestos indican que puede haber muchos cuadros de un estilo determinado. Pero, ¿un cuadro es de un solo estilo o puede haber cuadros que sean fusión de varios estilos?

Como observación, según los supuestos del enunciado, se pide guardar el estilo pictórico de las salas y pintores, pero sería redundante asociar la entidad *ESTILO_PICTÓRICO* con *SALA* y *ESTILO_PICTÓRICO* con *PINTOR*, ya que esa información se puede derivar de la interrelación **Es_de**.

Así, como diseñador nos encontramos con las siguientes consideraciones de supuestos que tomar, que debemos consultar con los usuarios y/o expertos del

dominio:

1. ¿Hay que controlar que se da un orden cronológico en las salas?
2. ¿Cuál es el identificador de CUADRO y PINTOR?
3. ¿Puede ser un cuadro anónimo y pintado por más de un pintor?
4. ¿Puede un cuadro pertenecer a más de un estilo pictórico?
5. ¿Puede haber salas sin cuadros o vacías?, ¿hay un máximo de cuadros a exponer en una sala?
6. ¿Se desea almacenar un histórico de períodos de fechas de exposición de un cuadro en sala?

La Figura 6.1 muestra el diagrama E/R con los supuestos comentados, pero quedan indeterminados los elementos anteriores denotados con “?”.

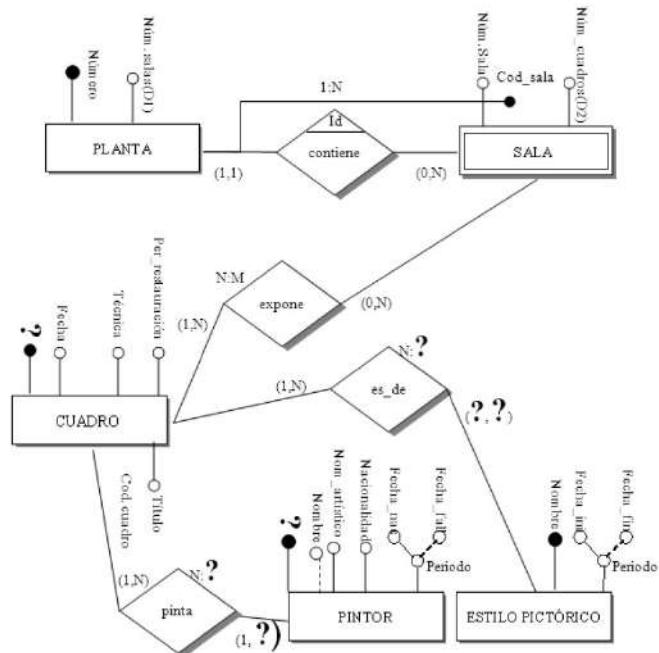


Figura 6.1. Subdiagrama E/R incompleto correspondiente a la parte 1 del enunciado

Tras consultar con el museo añadimos los siguientes supuestos complementarios al enunciado:

1. No hay que controlar que se da un orden cronológico en las salas.
2. Un pintor se identificará por un código y, además, no hay dos pintores

con un nombre artístico igual. Un cuadro se identificará por el pintor autor y por el título del cuadro, ya que un pintor nunca pondrá el mismo título a más de un cuadro suyo.

3. No hay obras anónimas y un cuadro es pintado por un solo artista o pintor.
4. Un cuadro pertenece a un solo estilo pictórico.
5. No puede haber salas vacías y el máximo de cuadros depende de la sala.
6. Hay que almacenar el histórico de fechas y lugar de exposición de los cuadros del museo en el fondo permanente.

Después de estos supuestos adicionales tenemos que *CUADRO* es una entidad débil en identificación y así se debe reflejar en la interrelación **Pinta**. La entidad *PINTOR* tendrá como AIP el atributo *Cod_pintor* y como atributo identificador alternativo (AIA) *Nom_artístico*.

La interrelación **Es_de** será 1:N y, por último, representamos en el diagrama la cardinalidad de la interrelación **Expone**, reflejando que como mínimo las salas tienen que tener un cuadro, y se añadirá a *SALA* un atributo *Max_cuadros*, donde se almacenará el número de cuadros máximo que se puede exponer en sala. Para almacenar el periodo de tiempo que están expuestos los cuadros en las salas se considerará un atributo *Periodo*, compuesto de *Fecha_ini* y *Fecha_fin*. Como se pide almacenar un histórico, y un cuadro tendrá varios periodos de exposición en una sala, este atributo será además mutivalorado.

Así, el diagrama E/R quedará como muestra la Figura 6.2:

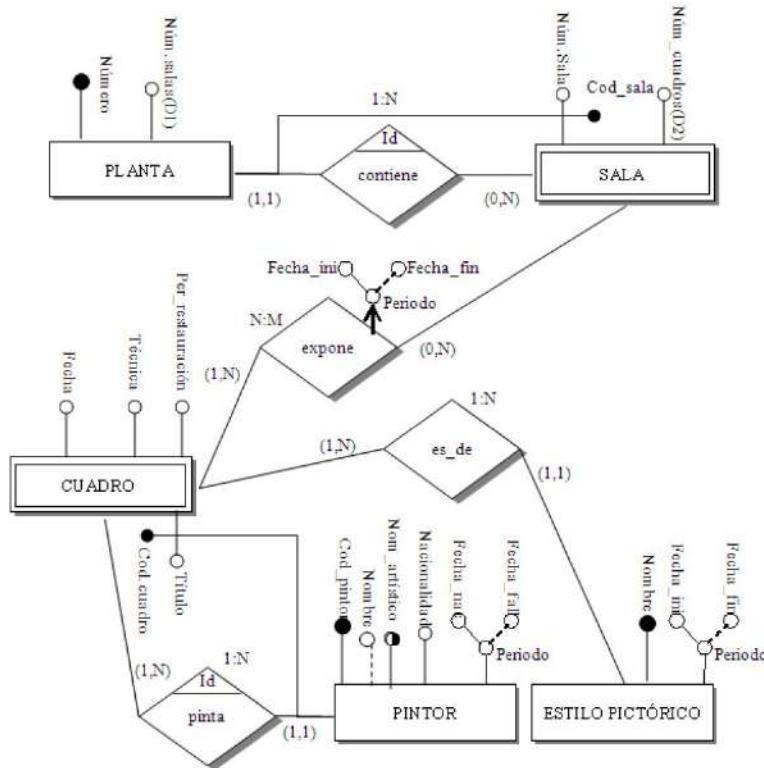


Figura 6.2. Subdiagrama E/R correspondiente a la parte 1 del enunciado

En este subdiagrama E/R hay semántica que no se refleja dando lugar a los siguientes supuestos semánticos complementarios al diagrama:

- SS1. La *Fecha_nac* < *Fecha_fall* en *PINTORES*, *Fecha_ini* < *Fecha_fin* en todas las fechas del atributo compuesto *Periodo* en este subdiagrama y siguientes.
- SS2. El valor calculado del atributo derivado (D1) *Num_salas* de la entidad *PLANTA*.
- SS3. El valor calculado del atributo derivado (D2) *Num_cuadros* de la entidad *SALA*.
- SS4. No se refleja que un cuadro solo puede estar en una sala en un momento dado.
- SS5. Que el número de cuadros de una sala no sobrepase al máximo fijado en cada una de las salas.
- SS6. Comprobaciones de fecha como que *Fecha* de entidad *CUADRO* debe ser mayor que la *Fecha_nac* y menor que *Fecha_fall* de pintor.

- SS7. El atributo *Técnica* de la entidad *CUADROS* tomará valores en el dominio *TÉCNICA CUADRO* = {óleo, temple, acrílico, esmalte}.

PARTE 2

Del buen estado de los cuadros se encargan los departamentos de conservación y restauración. Los conservadores y restauradores se identifican por un código y un nombre.

Los conservadores son los responsables del perfecto estado de los cuadros, de tal forma que hay un conservador encargado de las obras de un mismo estilo pictórico. Así, por ejemplo, habrá un conservador encargado de las obras de estilo renacentista, otro de las obras impresionistas, etc. Un conservador solo puede ser responsable de un estilo pictórico.

Los restauradores son los que se encargan de restaurar los cuadros; se hacen revisiones periódicas dependiendo del tipo de cuadro que sea. Los restauradores están especializados por pintores, es decir, un restaurador será el encargado de restaurar los cuadros de uno o varios pintores determinados. Así, habrá un restaurador para el Greco, otro de Goya, otro de Sysley y de Derain, etc. Hay que tener guardado en la base de datos el periodo de restauración de un cuadro (fecha de inicio y fecha de fin) y qué restaurador se encarga del trabajo. Un restaurador puede estar trabajando en más de una obra al mismo tiempo.

Se detectan las entidades *CONSERVADOR* y *RESTAURADOR*. Tanto conservadores como restauradores son personas que trabajan en el museo con la misma información a almacenar (código y nombre), sin embargo se interrelacionan de distinta manera, por ello, se creará una generalización donde la entidad *PERSONAL* es el supertipo de la jerarquía y las entidades *CONSERVADOR* y *RESTAURADOR* son los subtipos de la misma. Los atributos *Código AIP* y *Nombre* estarán en el supertipo. El atributo discriminante será *Tipo_Profesión*, que tomará valores en el dominio *TIPO_PROFESION* = {restaurador, conservador}.

Como las únicas personas, personal del museo, que aparecen en la semántica del enunciado son los restauradores y conservadores, consideraremos esta jerarquía como total. Quedaría por determinar la exclusividad o solapamiento, ya que los supuestos no dicen nada al respecto, y habría que saber también si un conservador puede restaurar y/o un restaurador puede ejercer de

conservador.

Para guardar la información de qué estilo pictórico es responsable un conservador, está la interrelación **Responsable** entre las entidades *CONSERVADOR* y *ESTILO_PICTORICO*. La cardinalidad de esta interrelación no queda del todo determinada, ya que en los supuestos se dice que un conservador es responsable de un único estilo, pero queda por saber si varios conservadores pueden ser responsables de un mismo estilo y si todo estilo debe tener un conservador asociado.

Para guardar la información de qué pintores son de los que puede restaurar obras cada restaurador, las entidades *RESTAURADOR* y *PINTOR* están asociadas por la interrelación **Puede_rest**. Según los supuestos, la cardinalidad está sin determinar del todo, porque se dice que un restaurador puede restaurar cuadros de uno a varios pintores, pero queda por saber cuestiones como si los cuadros de un mismo pintor pueden ser restaurados por distintos restauradores y si todo pintor tiene asignado un restaurador.

Para almacenar la información de qué restaurador realizó el último trabajo de cada uno de los cuadros, se tendrá la interrelación **Restaura** entre las entidades *RESTAURADOR* y *CUADRO*, como hay que almacenar el periodo de restauración del cuadro, la interrelación **Restaura** tendrá un atributo *Periodo* compuesto de *Fecha_ini* y *Fecha_fin*. Quedaría por determinar si se quiere un histórico de los trabajos de restauración realizados. En la interrelación **Restaura** la cardinalidad tampoco está definida del todo; en los supuestos se dice que un restaurador puede estar restaurando más de una obra, pero no se dice si un cuadro puede estar siendo restaurado por más de un restaurador en un mismo periodo de tiempo.

Entonces, nos encontramos con las siguientes cuestiones a dar respuesta:

1. ¿Puede un conservador restaurar?, y, ¿puede ejercer de conservador un restaurador?
2. ¿Puede haber más de un conservador responsable de un mismo estilo pictórico?, y, ¿todo estilo de cuadros del museo tiene un conservador como responsable?
3. ¿Puede haber varios restauradores especializados en un mismo pintor?, ¿puede haber varios restauradores restaurando al mismo tiempo un mismo cuadro?, y, ¿todo pintor autor de obras del museo tiene un restaurador asignado?

4. ¿Desea el cliente un histórico de quién y cuándo se realizaron los trabajos de restauración de los cuadros?

La Figura 6.3 muestra el diagrama E/R con los supuestos comentados y quedan indeterminados elementos denotados con “?”.

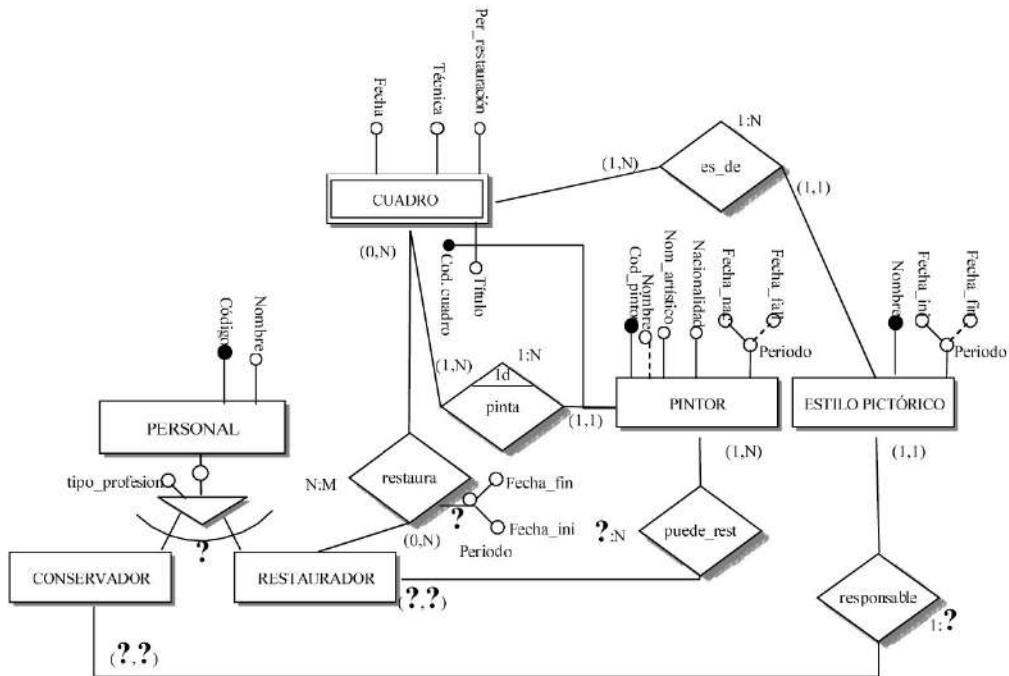


Figura 6.3. Subdiagrama E/R incompleto correspondiente a la parte 2 del enunciado

Tras dar respuesta a las cuestiones detectadas hay que considerar los siguientes supuestos complementarios al enunciado:

1. Un conservador no restaura cuadros, ni un restaurador puede ejercer de conservador.
2. Puede haber varios conservadores responsables de un mismo estilo pictórico, pero un estilo tiene que tener al menos un conservador como responsable.
3. Un mismo pintor puede estar asignado a más de un restaurador y un cuadro puede ser restaurado en el mismo periodo de tiempo por más de un restaurador. Todo pintor autor de cuadros del museo tiene que tener al menos un restaurador asociado.
4. Sí se quiere un histórico de quién y cuándo se realizaron los trabajos de restauración de los cuadros.

Después de estos supuestos adicionales a los iniciales del enunciado tenemos que la jerarquía será exclusiva. La cardinalidad de la interrelación **Responsable** es 1:N y la de **Puede_rest** N:M. Como hay que almacenar el histórico, el atributo *Periodo* de la interrelación **Restaura** lo consideraremos multivaluado para guardar los períodos de tiempo de restauración de un cuadro por un restaurador.

Así, después de considerar estos supuestos complementarios al enunciado, el diagrama E/R quedará como muestra la Figura 6.4:

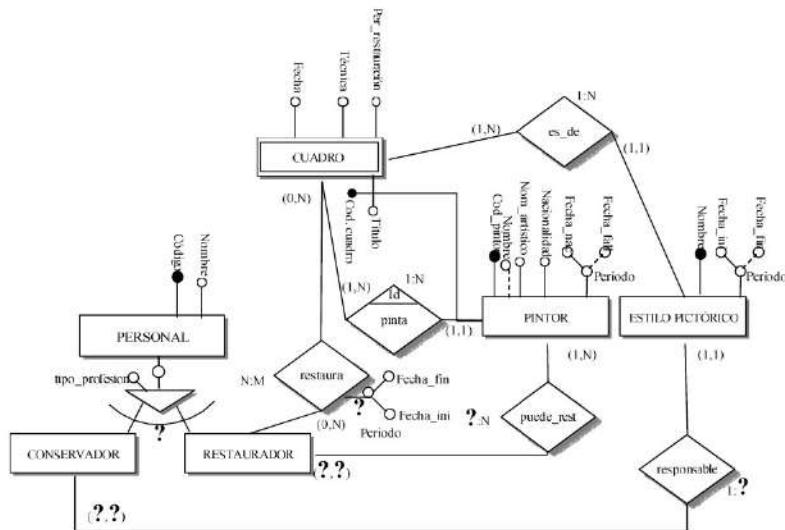


Figura 6.4. Subdiagrama E/R correspondiente a la parte 2 del enunciado Diagrama E/R

Al añadir a los supuestos semánticos complementarios al diagrama E/R (SS1 al SS7) están los siguientes supuestos semánticos no reflejados en el subdiagrama E/R (Figura 6.4) correspondiente a la parte 2:

- SS8. Un restaurador solo restaura cuadros de pintores que tenga asignados.
- SS9. El atributo *Tipo_profesión* de la entidad *PERSONAL* toma valores en el dominio *TIPO_PROFESIÓN* = {restaurador, conservador}.
- SS10. Un cuadro solo puede estar en un sitio: expuesto en sala o restaurándose.

PARTE 3

Los cuadros del fondo permanente pueden estar prestados a otros museos o a la exposición temporal del mismo museo. Hay que guardar la información de a quién se realiza el préstamo, cuáles son los datos del museo (nombre, dirección y teléfonos). Habrá que controlar dónde se encuentran en cada momento los cuadros propiedad del museo, si están en las salas del fondo permanente, exposición temporal, prestados o restaurándose.

La exposición temporal del museo se caracteriza por un título y sus fechas. Hay que almacenar el conservador encargado de la exposición, cuáles y el número de cuadros que alberga. La exposición temporal estará compuesta por obras del fondo permanente del museo, más las obras prestadas por otras instituciones o colecionistas particulares.

Como entidades encontramos *COLECCIÓN* y *EXPOSICIÓN_TEMPORAL*. La primera representa otros museos, instituciones y colecciones privadas de obras de arte y tiene como atributos a *NOMBRE*, *DIRECCIÓN* y el atributo multivalorado *Teléfonos*, pero no se dice cómo identificar esta entidad. La entidad *EXPOSICIÓN_TEMPORAL* tiene los atributos *Título* y el atributo compuesto *Periodo*, donde se guardará la fecha de inauguración y clausura de la exposición y, al igual que en *COLECCIÓN*, en los supuestos no se dice cómo se identifica una exposición. Como solo hay una exposición en el museo al mismo tiempo, no es necesario asociar la entidad *EXPOSICIÓN_TEMPORAL* al museo.

Según los supuestos semánticos, en nuestra base de datos hay que almacenar, además de los cuadros propios del museo, cuadros que nos prestan otras colecciones. Al distinguir entre obras propias del museo y ajena, se va a considerar una generalización para la entidad *CUADRO*, pasando a tener una jerarquía con supertipo *CUADRO* y con subtipos *PROPIO* y *AJENO*. El atributo discriminante será *Tipo_cuadro*, que tomará valores en el dominio *TIPO CUADRO* = {*propio*, *ajeno*}. Un cuadro será propio o ajeno, no pudiendo ser de los dos tipos al mismo tiempo, considerando así la jerarquía como total y exclusiva. Esta generalización tiene sentido porque los subtipos se interrelacionan de manera distinta; por ejemplo, los préstamos de otras colecciones (cuadros ajenos) solo se exponen en la exposición temporal, y en las salas del fondo permanente solo se exponen cuadros propios.

Al considerar la jerarquía en *CUADRO*, cambia el diagrama E/R que teníamos hasta ahora, la relación **Expone**, asociará la entidad *SALA* con *PROPIO*, ya que solo los cuadros propiedad del museo son los que se exponen

en las salas del fondo permanente.

Para guardar la información referente a estos prestamos de cuadros ajenos por otras colecciones a la exposición temporal, consideramos la interrelación ternaria **Nos_prestan** que asocia las entidades *EXPOSICION_TEMPORAL*, *COLECCION* y *AJENOS*. Se supone que una colección le presta al menos un cuadro a la exposición temporal. Pero para completar la cardinalidad de esta interrelación faltaría saber si un cuadro pertenece a una única colección o podría variar en el tiempo. Para controlar en qué periodo de tiempo están estas obras en préstamo, faltaría saber si ese periodo coincide con las fechas de la exposición temporal, o bien habría que considerar otro periodo distinto de préstamo. En ese caso se pondría un atributo *Periodo* en la interrelación para indicarlo.

Para modelar con qué más están asociados los cuadros ajenos faltaría saber: qué información de los cuadros no propiedad del museo se quiere almacenar en la base de datos, si los restauradores son responsables de pintores que no tengan cuadros propiedad del museo, si restauran cuadros de otros museos, si los conservadores son responsables de estilos pictóricos de obras que no sean propias del museo, etc.

Para almacenar la información de los cuadros que prestamos a museos, instituciones y colecciones privadas externas, la entidad *COLECCION* y *PROPIO* estarán interrelacionadas por **Prestamo_ext**. La cardinalidad será 1:N, teniendo en cuenta que un cuadro del museo puede o no estar prestado. Quedaría por determinar si se quiere guardar un archivo histórico de los periodos de préstamo a otras instituciones realizadas.

Existen préstamos de carácter interno de obras del museo (cuadros propios) a su exposición de manera temporal. Por ello consideramos la interrelación **Prestamo_int** que asocia las entidades *PROPIO* y *EXPOSICION_TEMPORAL*. En las cardinalidades hay que tener en cuenta que puede que una exposición temporal esté compuesta solo de cuadros ajenos. Para almacenar fechas de préstamo al igual que en **Nos_prestan** habría que saber si el periodo coincide con las fechas de la exposición temporal.

En el enunciado se indica que se desea almacenar el número de cuadros de los que consta la exposición, por ello se considerará en la entidad *EXPOSICION_TEMPORAL* el atributo *Num_cuadros* que será derivado (D3), y su valor, semántica no reflejada en el diagrama E/R, será el cálculo del número de cuadros de una exposición a partir de las interrelaciones **Prestamo_int** y **Nos_prestan**.

La *EXPOSICION_TEMPORAL* estará asociada a la entidad *CONSERVADOR* por la interrelación **Dirige** para almacenar el conservador encargado de la exposición. Como diseñadores sabemos por el enunciado que un conservador es responsable de un estilo pictórico, pero dependiendo de los estilos pictóricos de las obras que se exponen en una exposición temporal determinada, nos puede surgir una cuestión: ¿hay algún requisito que controlar para que un conservador sea director de una exposición temporal?

Así, nos encontramos con las siguientes cuestiones:

1. ¿Cómo se identifica una colección?
2. ¿Cómo se identifica una exposición temporal del museo?
3. ¿Se quiere guardar un archivo histórico de los préstamos realizados: de los que hace el museo a colecciones externas y/o a su exposición y de los cuadros que se prestan al museo?
4. ¿El periodo de préstamo interno (de cuadros propios del museo a la exposición temporal) y el externo (de cuadros ajenos) coincide con las fechas de la exposición?
5. ¿Hay algún requisito que controlar en relación al estilo pictórico para que un conservador pueda ser director de una exposición temporal?
6. ¿Qué información de los cuadros prestados se quiere almacenar en la base de datos?
7. ¿Realizan los restauradores restauraciones de cuadros de otras colecciones?
8. ¿Un cuadro tiene un único dueño (museo, institución o coleccionista), responsable del préstamo o puede variar en el tiempo?

La Figura 6.5 muestra el diagrama E/R con los supuestos comentados y quedan indeterminados elementos denotados con “?”.

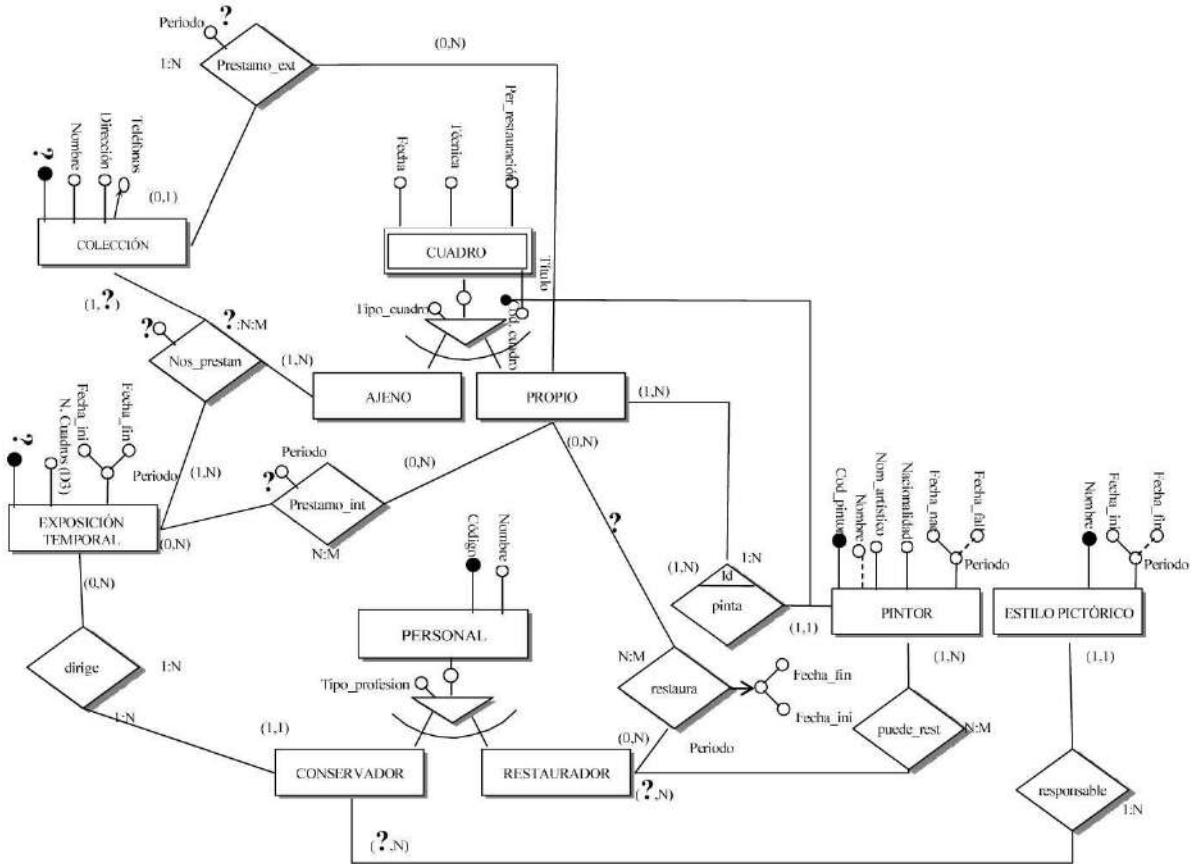


Figura 6.5. Subdiagrama E/R incompleto correspondiente a la parte 3 del enunciado

Como respuesta a las cuestiones anteriores, se consideran los siguientes supuestos semánticos complementarios al enunciado:

1. Una colección se va identificar por un código.
2. Una exposición temporal en un museo se identifica por el título.
3. No se quiere tener un archivo histórico y fechas de los préstamos del museo a otras colecciones, pero sí del resto.
4. El periodo de préstamo de un cuadro a la exposición temporal coincide con las fechas de la exposición.
5. No se requiere ningún requisito a los conservadores para dirigir una exposición temporal.
6. Respecto a los cuadros ajenos, se quiere guardar en la base de datos el título, pintor y estilo pictórico del cuadro y colección a la que

pertenece. Los conservadores no se responsabilizarán de estilos pictóricos de cuadros que no sean del museo, así como los restauradores no están encargados de pintores de cuadros que no sean los del museo.

7. Los restauradores solo restauran cuadros propios del museo.

8. Un cuadro tiene un único dueño y no cambia de propiedad.

Según estos supuestos se puede ver en la Figura 6.6 que *COLECCION* y *EXPO_TEMPORAL* tienen AIP. Como no se quiere tener un histórico de los préstamos de la interrelación **Prestamo_ext**, no será necesario contemplar un atributo “Periodo” en la interrelación. Respecto al histórico que sí que hay que almacenar de préstamos internos **Prestamo_int** y los que nos prestan **Nos_prestan** al coincidir el periodo de préstamo con las fechas de la exposición, ya estaría contemplado.

Según el supuesto semántico complementario al enunciado 7 anterior, habrá que reconsiderar cardinalidades mínimas que había en las interrelaciones **Puede_rest** y **Responsable**, ya que puede que haya en la base de datos almacenados pintores (de cuadros ajenos) que no estén asignados a ningún restaurador, igual que habrá estilos pictóricos (de cuadros ajenos) asociados a ningún conservador.

Como los restauradores solo hacen trabajos de cuadros propios, cambia el diagrama E/R que se tenía en la interrelación **Restaura**, ya que asociará *RESTAURADOR* con el subtipo *PROPIO* y el atributo *Per_restauración* pasará a ser información solo del subtipo *PROPIO*.

Y, por último, según el supuesto semántico complementario al enunciado 8 anterior la interrelación ternaria **Nos_prestan** tiene cardinalidad 1:N:M.

Así, después de considerar estos supuestos complementarios al enunciado, el diagrama E/R quedará como muestra la Figura 6.6:

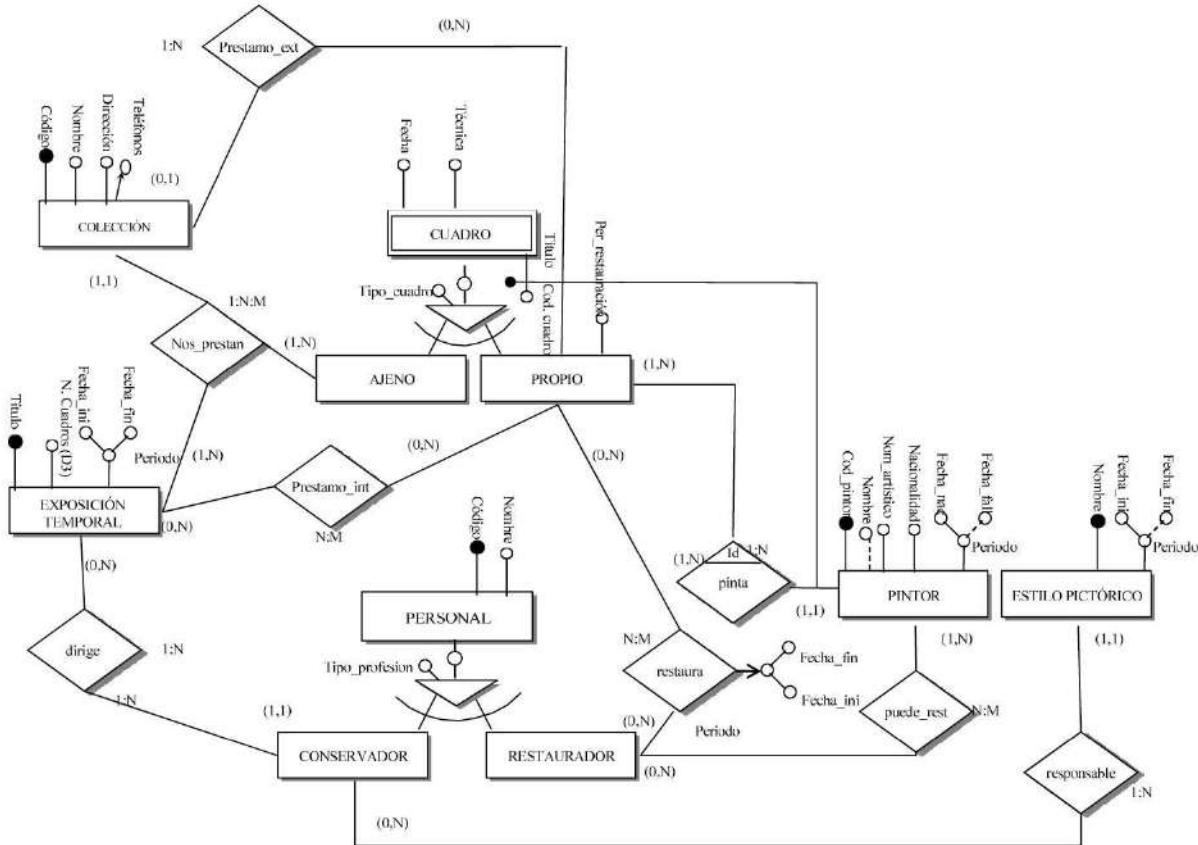


Figura 6.6. Subdiagrama E/R correspondiente a la parte 3 del enunciado

En el subdiagrama E/R (Figura 6.6) correspondiente a la parte 3, no se reflejan los siguientes supuestos complementarios al diagrama E/R que se añadirán a los ya anteriores (SS1 al SS10):

Modificamos el SS10 y pasa a ser:

- SS10. Un cuadro propio solo puede estar expuesto en sala, restaurándose, prestado a una colección externa o en la exposición temporal.
- SS11. El valor calculado del atributo derivado (D3) *Num_cuadros* de la entidad *EXPOSICION_TEMPORAL*.
- SS12. El estilo pictórico del que es responsable un conservador es el estilo que tenga al menos un cuadro propio del museo.
- SS13. Los pintores de los que está encargado un restaurador serán pintores que sean autores de al menos un cuadro propio del museo.
- SS14. Las fechas de la exposición temporal no están solapadas.

PARTE 4

Un restaurador podrá estar trabajando en un máximo de tres obras y solo se debe permitir que tenga en cola de espera para restauración un cuadro.

Cuando haya que restaurar un cuadro, hay que asignar ese trabajo a un restaurador que le pertenezca, y habrá que establecer una fecha próxima prevista para inicio de restauración; si el restaurador indicado estuviera libre, la fecha sería la del día, empezando ese mismo día el trabajo de restauración. Si, en cambio, estuviera ocupado, es decir, está restaurando tres obras en ese momento, la fecha de inicio de la restauración de la obra sería la primera fecha de finalización de restauración de alguna de las obras en las que está trabajando.

Como un restaurador solo puede tener asignado tres cuadros en un momento dado cambia la cardinalidad máxima en la interrelación **Restaura**.

Para poder almacenar la información necesaria y llevar a cabo la gestión de próxima restauración de un cuadro, tal y como indica el enunciado, las entidades **RESTAURADOR** y **PROPIO** estarán asociadas por la interrelación **Rest_próxima**. En la interrelación estará el atributo derivado (D4) *Fecha_prox*, cuyo valor, semántica no reflejada en el diagrama, será calculado a partir de la interrelación **Restaura**. Así, cuando haya que asignar un nuevo trabajo de restauración a un restaurador, habrá que comprobar que no tenga en cola de espera ningún trabajo de restauración. Además, si ese restaurador está trabajando en menos de 3 restauraciones, el atributo *Fecha_prox* tomará la fecha del día de la operación y, si por el contrario, está trabajando en tres trabajos, *Fecha_prox* será la fecha más cercana de entre las *Fecha_fin* de finalización de los tres trabajos de restauración en los que está implicado.

La propuesta de solución final será el siguiente diagrama E/R (Figura 6.7). Como supuestos semánticos no contemplados en el diagrama E/R, añadir:

- SS15. El valor calculado del atributo derivado (D4) *Fecha_prox*.

Como propuesta de solución se obtiene el esquema E/R, compuesto del diagrama E/R representado en la Figura 6.7, junto con los supuestos SS1 al SS15 complementarios al diagrama.

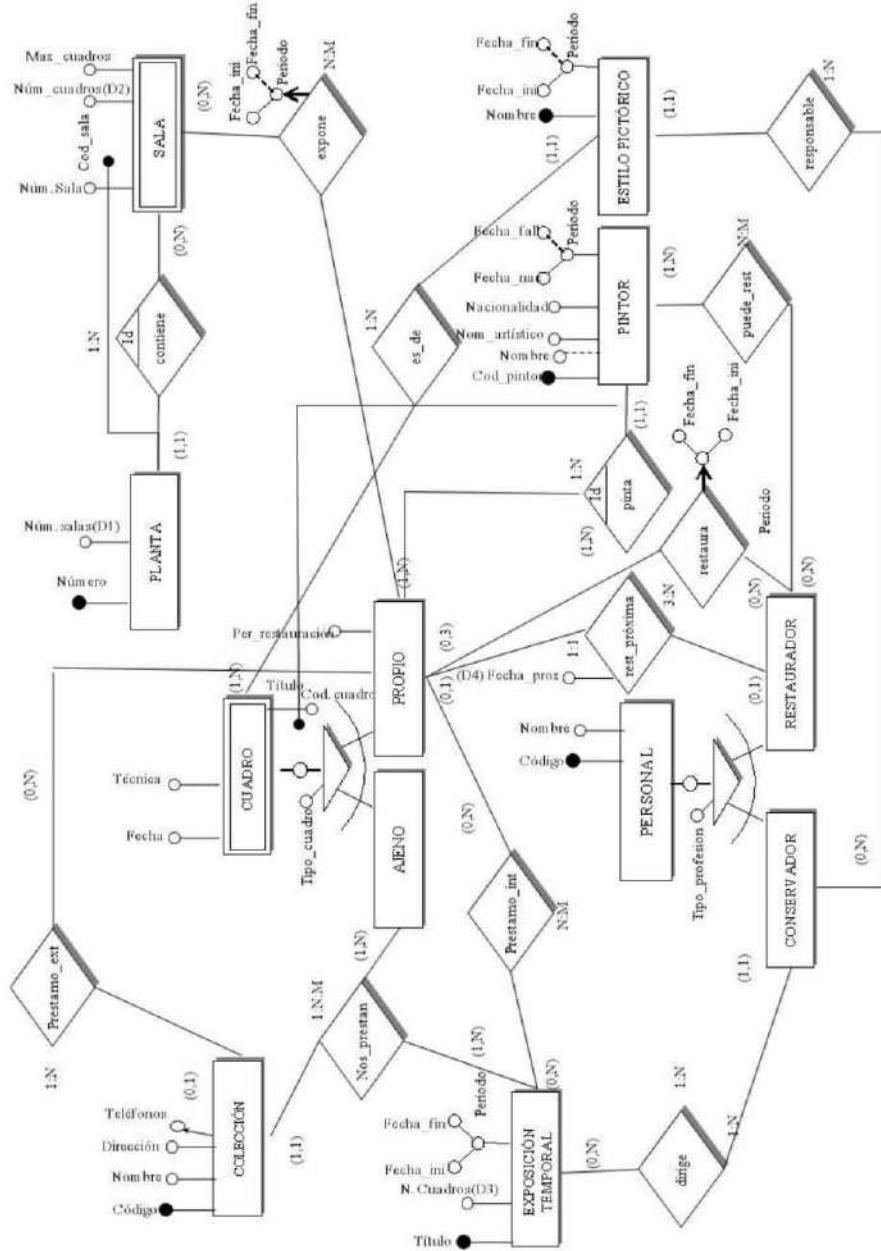


Figura 6.7. Diagrama E/R propuesta solución final

6.1.2 Diseño lógico: Transformación al Esquema Relacional

Partiendo del esquema conceptual del bloque anterior formado por el diagrama E/R y supuestos semánticos no reflejados en el diagrama, se va a pasar a realizar una transformación a un esquema lógico (grafo relacional), indicando por medio de *checks*, aserciones y disparadores la semántica que no se pueda recoger en el grafo.

DISCUSIÓN DEL ENUNCIADO

Cada entidad del diagrama E/R de la Figura 6.7 se transforma en una relación; los atributos univaluados y simples se transforman en columnas de la relación; los atributos compuestos se descomponen y se transforman en distintas columnas de la relación; cada Atributo Identificador Principal (AIP) de cada entidad se convertirá en la clave primaria de la relación; cada Atributo Identificador Alternativo (AIA) se transforma en clave alternativa en la relación.

En cuanto a otro tipo de constructores del modelo E/R (atributos multivaluados, interrelaciones, jerarquías, etc.), su transformación se detalla a continuación, especificando la semántica que se pierde en la transformación al grafo relacional mediante *checks*, aserciones y disparadores. Así, por ejemplo, todo atributo que toma valores definidos en un dominio al hacer la transformación al esquema relacional se controlará con *checks* debido a que esta semántica será imposible de almacenar en el grafo relacional.

Las restricciones de integridad referencial de clave ajena si no se especifican en el enunciado otro tratamiento específico, se van a considerar como borrado sin acción y modificación en cascada (DNA/UC) debido a que son las menos restrictivas. En la explicación de la solución adoptada solo se indicará otro tipo de restricción de integridad cuando sea necesario.

Se irán haciendo transformaciones de partes del diagrama E/R global, hasta construir el grafo relacional global.

PARTE 1

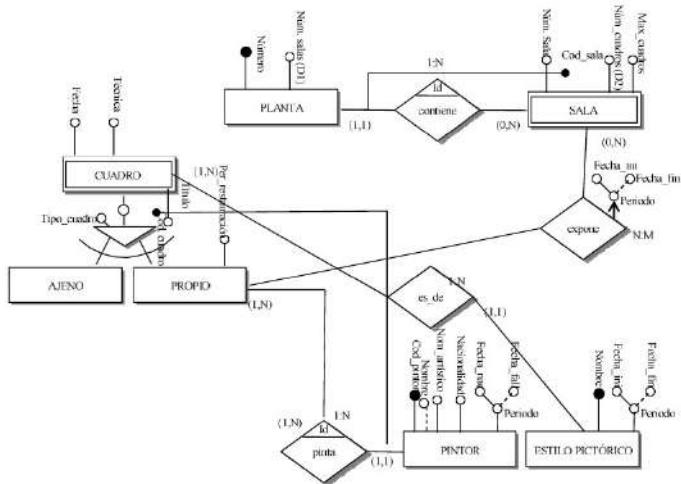


Figura 6.8. Subdiagrama E/R de la parte 1 a transformar

El subdiagrama E/R de la Figura 6.8, se transforma en el grafo relacional de la Figura 6.9.

La interrelación **Contiene** es una interrelación en identificación, por lo que se transforma mediante una propagación de clave de la relación *PLANTAS* que pasará a formar parte de la clave primaria de la relación *SALAS*. Al igual se hará con la interrelación en identificación **Pinta**, se propagará la clave primaria de la relación *PINTORES*, pasando a ser clave primaria de la relación *CUADROS*. En ambos casos por la fuerte dependencia, las restricciones de integridad referencial se tratarán como borrado y modificación en cascada (DC/UC).

La jerarquía *CUADRO* da lugar a tres nuevas relaciones, una por cada entidad de la jerarquía. Las relaciones que provienen de los subtipos *PROPIO* y *AJENO* tendrán como clave primaria la clave primaria *cod_pintor*, *titulo* de la relación *CUADROS*, y serán consideradas como claves ajenas referenciando a dicha relación con opciones de borrado y modificación de las claves ajenas en cascada (DC/UC), debido a su dependencia en existencia respecto a las ocurrencias en la relación *CUADROS*. El atributo discriminante *Tipo_cuadro* de la jerarquía pasa a formar parte como columna en la relación *CUADROS*. Como se trata de una jerarquía total, esta columna no podrá tomar valores nulos (no será opcional). Además, como se trata de una jerarquía exclusiva, la columna *tipo_cuadro* solo puede tomar valores simples de los subtipos de la jerarquía que habrá que comprobar mediante una aserción y un *check* debido a que esta semántica será imposible de almacenar en el grafo relacional.

La interrelación **Es_de** es una interrelación 1:N y como transformación se

va a propagar la clave: la clave primaria de la relación *ESTILOS_PICTORICOS* se propaga como clave ajena a la relación *CUADROS*. La interrelación N:M **Expone** se transforma como una nueva relación *EXPUESTOS* formada por las claves primarias de las relaciones *PROPIOS* y *SALAS* (tratadas como claves ajenas en esta relación) junto con los atributos del atributo compuesto multivaluado *Periodo* de la interrelación. Como un cuadro no puede estar al mismo tiempo en distintas salas, pero sí una sala expone al mismo tiempo más de un cuadro, tenemos dos claves primarias candidatas para la relación *EXPUESTOS* que serían $\{pintor, \text{titulo}, \text{fecha_ini}\}$ y $\{pintor, \text{titulo}, \text{fecha_fin}\}$, como *Fecha_Fin* es opcional, $\{pintor, \text{titulo}, \text{fecha_ini}\}$ será clave primaria y $\{pintor, \text{titulo}, \text{fecha_fin}\}$ clave alternativa. Para asegurar que no haya salas vacías sin cuadros, sería necesario comprobar esta semántica con una asercción, y para no permitir que haya expuestos más cuadros que el máximo fijado por sala habría que añadir un disparador.

No es posible representar en el grafo relacional la forma de calcular los atributos derivados, por lo que será necesario incluir unos disparadores por cada uno de los atributos derivados: *Num_salas* (D1) de la entidad *PLANTA* y *Num_cuadros* (D2) de la entidad *SALA*. Con estos disparadores las columnas *num_salas* de la relación *PLANTAS* y *num_cuadros* de la relación *SALAS* tendrán su valor calculado.

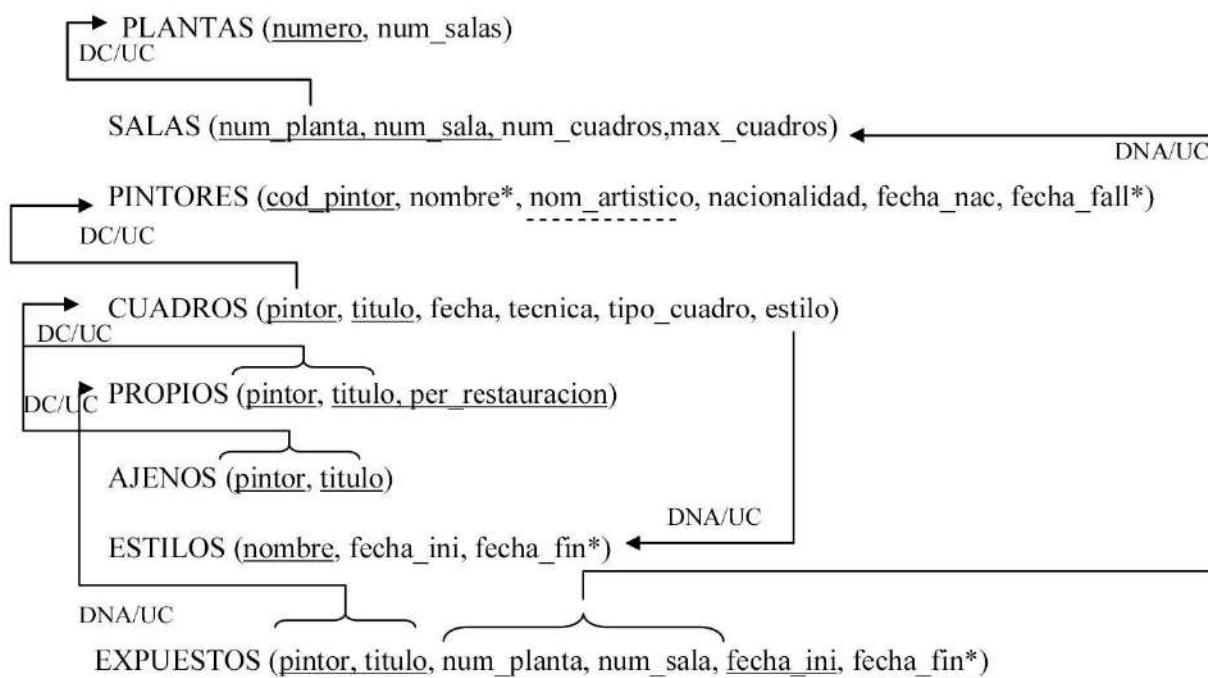


Figura 6.9. Grafo relacional correspondiente a la transformación del subdiagrama E/R parte I

PARTE 2

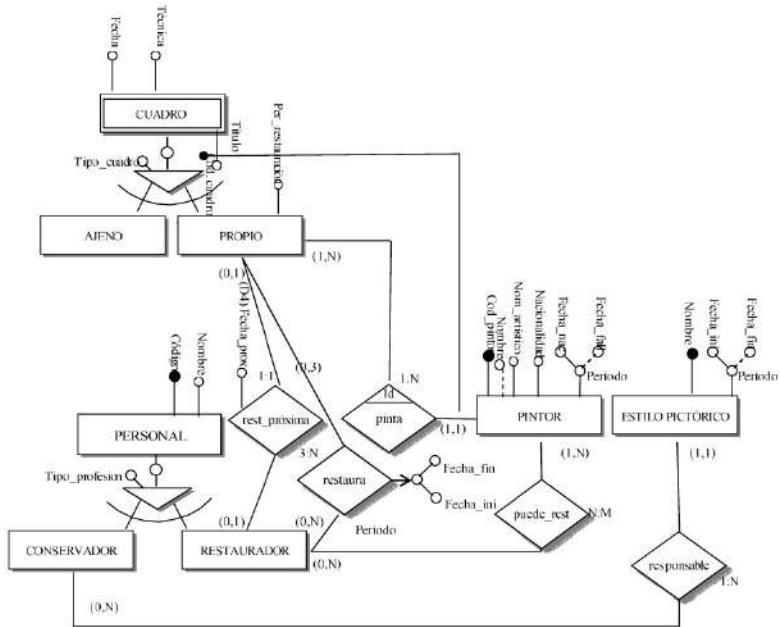


Figura 6.10. Subdiagrama E/R de la parte 2 a transformar

El subdiagrama E/R de la Figura 6.10, se transforma en el grafo relacional de la Figura 6.11.

La jerarquía *PERSONAL* da lugar a tres nuevas relaciones: *PERSONAL*, *CONSERVADORES* y *RESTAURADORES*. *CONSERVADORES* y *RESTAURADORES* tendrán como clave primaria la clave primaria *codigo* de la relación *PERSONAL*, y serán consideradas como clave ajena referenciando a la relación *PERSONAL* con opciones de borrado y modificación de las claves ajenas en cascada (DC/UC). El atributo discriminante *tipo_profesion* pasa a formar parte como columna en la relación *PERSONAL*. Al ser una jerarquía total, la columna *tipo_profesion* no podrá tomar valores nulos (no será opcional) y al ser exclusiva, solo puede tomar valores simples de los subtipos de la jerarquía, que habrá que comprobarlo mediante una asercción y un *check* debido a que esta semántica será imposible de almacenar en el grafo relacional.

La interrelación 1:N **Responsable** se va a transformar propagando la clave primaria de la relación *ESTILOS_PICTORICOS* como clave ajena a la relación *CONSERVADORES*.

Se ha tomado como solución el dejar una relación *RESTAURADORES* con un único atributo, pero por motivos de eficiencia se podría considerar la opción

de quitarla. Así, las claves ajenas que referencian a *RESTAURADORES* referenciarían a la relación *PERSONAL*, pero habría que incluir al grafo relacional aserciones para hacer comprobaciones de que es *Tipo_profesión* restaurador.

La interrelación **Puede_rest** es una interrelación N:M que se transforma como una nueva relación *PUEDE_RESTAURAR* formada por las claves primarias de las relaciones *PINTORES* y *RESTAURADORES* (tratadas como claves ajenas en esta relación). Como un restaurador puede restaurar obras de más de un pintor y cuadros de un mismo pintor pueden ser restaurados por más de un restaurador, tenemos que estas claves ajenas formarán parte de la clave primaria de la relación *PUEDE_RESTAURAR*. Como un restaurador tiene como mínimo un pintor asignado, sería necesario comprobar esta semántica con una aserción.

La interrelación N:M **Restaura** se transformará en una relación *RESTAURACIONES* formada por las claves primarias de las relaciones *PROPIOS* y *RESTAURADORES* (tratadas como claves ajenas en esta relación) junto con los atributos del atributo compuesto multivaluado *Periodo*. Como un restaurador puede al mismo tiempo estar trabajando en más de un cuadro y un cuadro puede ser restaurado al mismo tiempo por varios restauradores, estas claves ajenas formarán parte de la clave primaria junto con *fecha_ini*. Será necesario incluir un disparador asociado al grafo para controlar que un restaurador está trabajando en un máximo de tres cuadros.

En la interrelación 1:1 **Res_próxima** se ha optado por trasformar en una nueva relación *RESTAURACIONES_PROXIMAS*, ya que al tener las dos cardinalidades mínimas con cero se evitan valores nulos en las claves ajenas. Estará formada con las claves primarias de las relaciones *PROPIOS* y *RESTAURADORES* (tratadas como claves ajenas en esta relación) junto con *fecha_prox*, cualquiera de las dos sería candidata para clave primaria, ya que un restaurador solo puede tener un cuadro en cola de espera para restaurar, y un cuadro solo puede estar asignado a un restaurador para su próxima restauración, hemos optado por considerar *restaurador* como clave primaria y *pintor*, *titulo* como clave alternativa. Para determinar el valor de la columna *fecha_prox* por provenir del atributo derivado (D4) será necesario incluir un disparador.

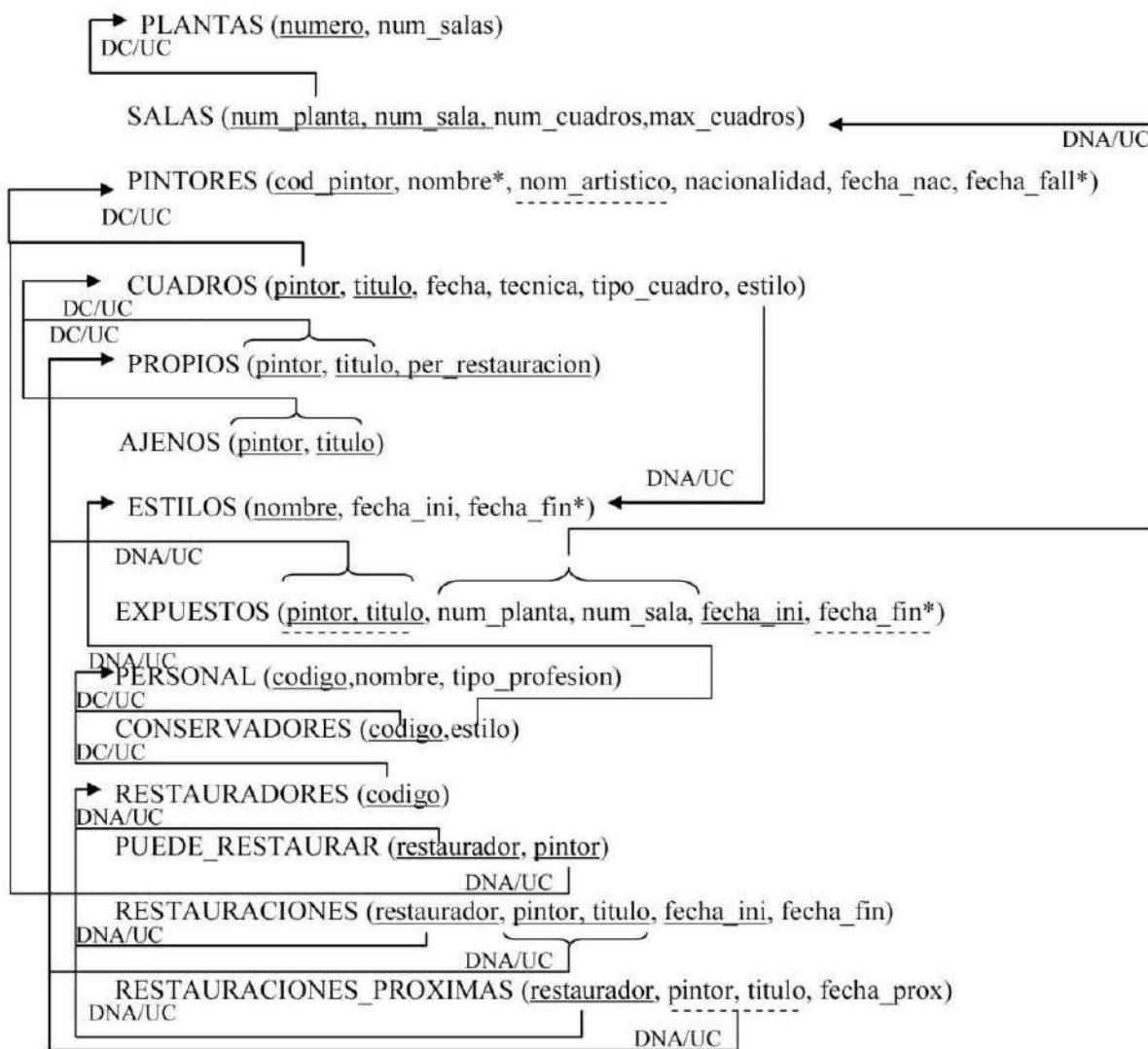


Figura 6.11. Grafo relacional correspondiente a la transformación del subdiagrama E/R parte 1 y 2

PARTE 3

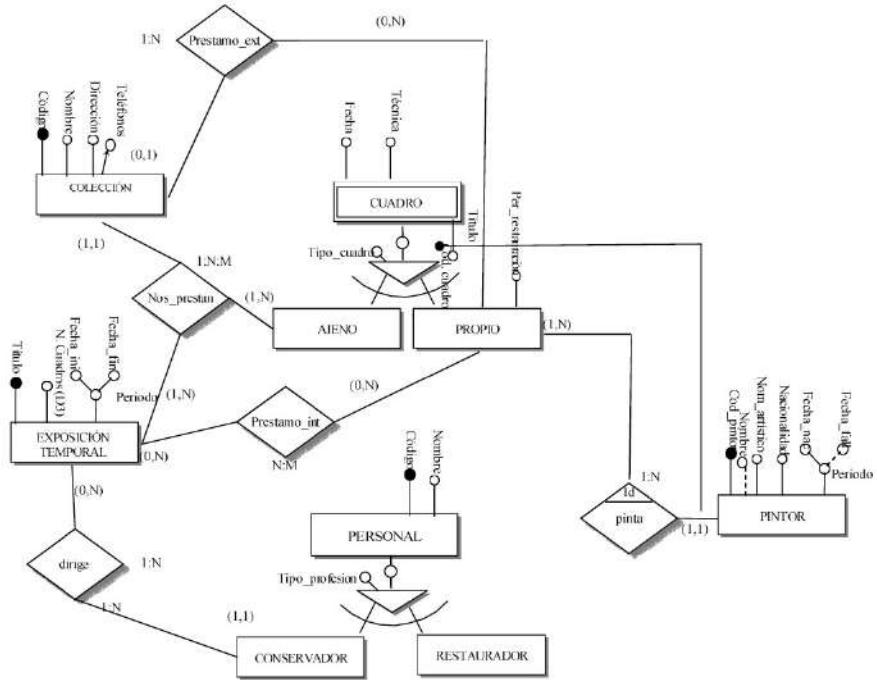


Figura 6.12. Subdiagrama E/R de la parte 3 a transformar

El subdiagrama E/R de la Figura 6.12, se transforma en el grafo relacional de la Figura 6.13.

La interrelación 1:N **Dirige**, se va a transformar migrando la clave como en casos anteriores, así se tiene que la clave primaria de la entidad *CONSERVADORES* pasa a ser clave ajena de la relación *EXPOSICIONES_TEMPORALES*.

La interrelación ternaria 1:N:M **Nos_prestan** se transformará en una nueva relación *PRESTAMOS_EXT_EXPO* formada por las claves primarias de las relaciones *EXPOSICIONES_TEMPORALES*, *COLECCIONES* y *AJENOS* (tratadas como claves ajenas en esta relación). La clave primaria de *PRESTAMOS_EXT_EXPO* estará compuesta de la clave primaria de *EXPOSICIONES* y *AJENOS*; no hay que incluir la de *COLECCIONES* ya que un cuadro prestado a la exposición pertenece a una única colección.

Al transformar la interrelación 1:N **Prestamo_ext**, habrá una migración de la clave primaria de la relación *COLECCIONES* como clave ajena en la relación *PROPIOS*; será opcional ya que puede haber almacenadas colecciones a las que no se les haya prestado cuadros porque pueden estar en la base de datos por realizar préstamos a la exposición temporal. La interrelación **Prestamo_int**, se transformará en una relación *PRESTAMOS_INTERNOS* formada por las claves

primarias de las relaciones *EXPOSICIONES_TEMPORALES* y *PROPIOS* (tratadas como claves ajenas en esta relación) que serán su clave primaria.

No es posible representar en el grafo relacional la forma de calcular el atributo derivado (D3), por lo que será necesario incluir un disparador que calcule el valor de *num_cuadros* de la relación *EXPOSICIONES TEMPORALES*.

Como final de este bloque se ofrece una propuesta de solución del esquema relacional, que se compone del grafo relacional indicado en la Figura 6.13, junto con *check*, aserciones y disparadores que se han ido indicando y que son necesarios para que no haya pérdida de semántica en la transformación del diagrama E/R al relacional.

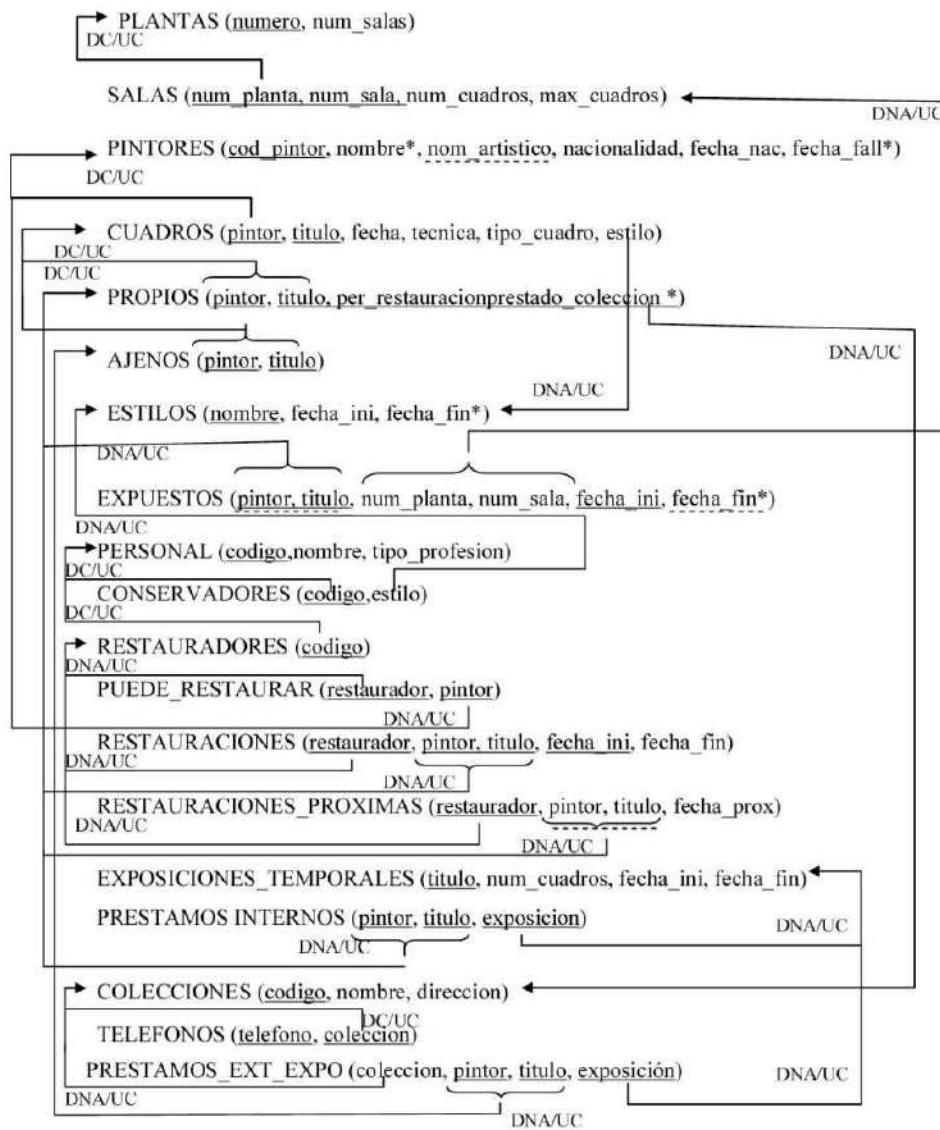


Figura 6.13. Grafo relacional final

6.1.3 Diseño Lógico Específico

En el bloque anterior se ha obtenido el esquema relacional. Como se ha ido comentando hay supuestos que es necesario controlar con mecanismos externos al diagrama o grafo relacional como *checks*, aserciones y disparadores. Entre los supuestos semánticos a incluir al grafo relacional, tal como se indica la Tabla 6.1 hay que considerar:

- Los supuestos que no estaban reflejados en el diagrama E/R (SS1-SS15) y que tampoco se han reflejado en el grafo relacional. Por

ejemplo, el supuesto SS4, por la definición de claves primarias y alternativas, sí se refleja en el grafo relacional y no es necesario considerar ningún mecanismo externo.

- Los supuestos semánticos para evitar la pérdida de semántica en la transformación del diagrama E/R al relacional.

Existen diferencias entre el modelo relacional estándar expuesto y los Sistemas Gestores de Bases de Datos (SGBD) comerciales que no implementan el modelo relacional completo. En este caso práctico que aquí se presenta, la implementación del modelo relacional en el SGBD ORACLE se proporciona en el material adicional del libro, introducido en el Capítulo 5.

Expuestos qué supuestos semánticos hay que controlar, en la Tabla 6.1 se resumen cada uno de ellos con el mecanismo ORACLE a utilizar, a diferencia del Capítulo 2, donde se mostraba el estándar ANSI SQL.

SUPUESTOS SEMÁNTICOS NO REFLEJADOS EN EL DIAGRAMA E/R		MECANISMO ORACLE
SS1	La <i>Fecha_ini < Fecha_fin</i> en todas las fechas del atributo compuesto <i>Periodo</i> y otros.	<i>CHECK</i> en las tablas: <i>PINTORES</i> , <i>EXPUESTOS</i> , <i>RESTAURACIONES</i> , <i>EXPOSICIONES_TEMPORALES</i> .
SS2	El valor calculado del atributo derivado (D1) <i>Num_salas</i> de la entidad <i>PLANTA</i> .	<i>TRIGGER</i> después de insertar, borrar o modificar en la tabla <i>SALAS</i> .
SS3	El valor calculado del atributo derivado (D2) <i>Num_cuadros</i> de la entidad <i>SALA</i> .	<i>TRIGGER</i> después de insertar, borrar, modificar en la tabla <i>EXPUESTOS</i> .

SS4	No se refleja que un cuadro solo puede estar en una sala en un momento dado.	No necesario (reflejado en grafo relacional).
SS5	Que el número de cuadros de una sala no sobrepase al máximo fijado en cada una de las salas.	<i>TRIGGER</i> antes de insertar y/o modificar en la tabla <i>EXPUESTOS</i> . Consulta la tabla <i>SALAS</i> .
SS6	Comprobaciones de fecha como que <i>Fecha</i> de entidad <i>CUADRO</i> debe ser mayor que la <i>Fecha_ini</i> y menor que <i>Fecha_fin</i> de pintor.	<i>TRIGGER</i> antes de <i>INSERTAR</i> en tabla <i>CUADROS</i> . Consulta la tabla <i>PINTORES</i> .
SS7	El atributo <i>Técnica</i> de la entidad <i>CUADROS</i> tomará valores en el dominio TÉCNICA_CUADRO = {óleo, temple, acrílico, esmalte}.	<i>CHECK</i> en la tabla <i>CUADROS</i> .
SS8	Un restaurador solo restaura cuadros de pintores que tenga asignados.	<i>TRIGGER</i> antes de insertar en la tabla <i>RESTAURACIONES</i> . Consulta la tabla <i>PUEDE_RESTAURAR</i> .
SS9	El atributo <i>Tipo_profesión</i> de la entidad <i>PERSONAL</i> toma valores en el dominio TIPO_PROFESIÓN = {restaurador, conservador}.	<i>CHECK</i> en la tabla <i>PERSONAL</i> .
SS10	Un cuadro propio solo puede estar expuesto en sala, restaurándose, prestado a una colección externa o en la exposición temporal.	<i>TRIGGER</i> antes de insertar en <i>PROPIOS</i> . Consulta las tablas <i>EXPUESTOS</i> , <i>RESTAURACIONES</i> , <i>PRESTAMOS_INTERNOS</i> .
		<i>TRIGGER</i> antes de insertar en la tabla <i>EXPUESTOS</i> . Consulta las tablas <i>PROPIOS</i> , <i>RESTAURACIONES</i> , <i>PRESTAMOS_INTERNOS</i> .

	<p>Un cuadro propio solo puede estar expuesto en sala, restaurándose, prestado a una colección externa o en la exposición temporal. (Continuación)</p>	<p><i>TRIGGER</i> antes de insertar en <i>RESTAURACIONES</i>. Consulta las tablas <i>PROPIOS</i>, <i>EXPUESTOS</i>, <i>PRESTAMOS_INTERNOS</i>.</p> <p><i>TRIGGER</i> antes de insertar en <i>PRESTAMOS_INTERNOS</i>. Consulta las tablas <i>PROPIOS</i>, <i>EXPUESTOS</i>, <i>RESTAURACIONES</i>.</p>
SS11	<p>El valor calculado del atributo derivado (D3) <i>Num_cuadros</i> de la entidad <i>EXPOSICIÓN_TEMPORAL</i>.</p>	<p><i>TRIGGER</i> antes de insertar, borrar o modificar en la tabla <i>PRESTAMOS_EXT_EXPO</i>.</p>
		<p><i>TRIGGER</i> antes de insertar, borrar o modificar en la tabla <i>PRESTAMOS_INTERNOS</i>.</p>
SS12	<p>El estilo pictórico del que es responsable un conservador es el estilo que tenga al menos un cuadro propio del museo.</p>	<p><i>TRIGGER</i> antes de insertar en la tabla <i>CONSERVADORES</i>. Consulta las tablas <i>CUADROS</i>, <i>PROPIOS</i>.</p>
SS13	<p>Los pintores de los que está encargado un restaurador serán pintores que sean autores de al menos un cuadro propio del museo.</p>	<p><i>TRIGGER</i> antes de insertar en la tabla <i>PUEDE_RESTAURAR</i>. Consulta sobre la tabla <i>CUADROS</i>.</p>
SS14	<p>Las fechas de la exposición temporal no están solapadas.</p>	<p><i>TRIGGER</i> antes de insertar en la tabla <i>EXPOSICIONES_TEMPORALES</i>. Consulta sobre la misma tabla.</p>
SS15	<p>El valor calculado del atributo derivado (D4) <i>Fecha_prox</i>.</p>	<p><i>TRIGGER</i> antes de insertar en la tabla <i>RESTAURACIONES_PROXIMAS</i>. Consulta sobre la tabla <i>RESTAURACIONES</i>.</p>

SUPUESTOS PÉRDIDA SEMÁNTICA TRANSFORMACIÓN	
Comprobación exclusividad de la jerarquía <i>CUADRO</i> .	<i>CHECK</i> en la tabla <i>CUADROS</i> . <i>TRIGGER</i> antes de insertar sobre la tabla <i>AJENOS</i> . Consulta sobre las tablas <i>CUADROS</i> , <i>PROPIOS</i> .
	<i>TRIGGER</i> antes de insertar sobre la tabla <i>PROPIOS</i> . Consulta sobre las tablas <i>CUADROS</i> , <i>AJENOS</i> .
Comprobación exclusividad de la jerarquía <i>PERSONAL</i> .	<i>CHECK</i> sobre la tabla <i>PERSONAL</i> . <i>TRIGGER</i> antes de insertar sobre la tabla <i>CONSERVADORES</i> . Consulta sobre las tablas <i>PERSONAL</i> , <i>RESTAURADORES</i> .
No hay salas vacías.	<i>TRIGGER</i> antes de insertar sobre la tabla <i>RESTAURADORES</i> . Consulta sobre las tablas <i>PERSONAL</i> , <i>CONSERVADORES</i> .
Un restaurador tiene al menos un pintor asociado.	<i>TRIGGER</i> antes de borrar o modificar en la tabla <i>EXPUESTOS</i> . Consulta a la tabla <i>EXPUESTOS</i> , o a tabla <i>SALAS</i> .
	<i>TRIGGER</i> antes de borrar o modificar en <i>PUEDE_REST</i> . Consulta sobre la tabla <i>PUEDE_REST</i> .
Un restaurador puede estar trabajando en un máximo de tres cuadros.	<i>TRIGGER</i> antes de insertar o modificar en <i>RESTAURACIONES</i> . Consulta a la tabla <i>RESTAURACIONES</i> .

6.2 CASO PRÁCTICO 2: EDITORIAL DE LIBROS DE TEXTO ESCOLAR

Una editorial dedicada a la venta de libros de texto en los centros escolares quiere tener una base de datos donde registrar la actividad comercial de su red.

La editorial visita los centros escolares a través de una red comercial formada por dos tipos de vendedores: contratados y promotores. Los comerciales contratados poseen un contrato con la editorial. El contrato se identifica por un código, una fecha de alta y de baja. Además, cada comercial contratado posee un coche de empresa cuyo modelo depende de su categoría. Respecto a los promotores, estos están dados de alta como autónomos y utilizan su propio coche. Solo se desea almacenar información sobre el modelo de coche de los empleados contratados. Los promotores están supervisados por un vendedor contratado. Cada contratado no puede supervisar a más de tres promotores.

Los datos que se quieren recoger de los vendedores son su código que les identifica, nombre y apellidos, NIF, tipo de vendedor (contratado o autónomo), categoría (1, 2 ó 3 para los contratados y A, B o C para los promotores) y correo personal de empresa.

La editorial cuenta con varias delegaciones comerciales dedicadas al suministro de los centros escolares. Los centros pertenecen a una determinada delegación y están identificados por un código y se desea almacenar de ellos el nombre del centro, dirección (calle, número, población, provincia, código postal), titularidad del centro (público, religioso o laico) y la delegación a la que pertenece. Cada delegación tiene un nombre y un correo propio. Cada vendedor pertenece a una única delegación comercial y tiene asignados una serie de centros (al menos uno) de su delegación a los que acude regularmente. Los centros pueden estar asignados durante un curso académico a varios vendedores (al menos uno).

El vendedor establece una ruta comercial visitando centros durante el día y cuando llega a casa tiene que hacer el informe de la actividad comercial realizada. Se desea almacenar información histórica sobre las visitas que ha hecho cada vendedor a cada centro. Las visitas de los vendedores se identifican en cada centro por la fecha y la hora de inicio, deseando almacenar también información sobre la hora de fin si se conoce y una descripción de la visita.

Será deseable también almacenar información sobre la totalidad de los

gastos asociados a la visita. A una visita pueden acudir varios vendedores al mismo tiempo de los que se deseará almacenar los gastos asociados a la visita de cada uno de ellos si los hubiera.

Los gastos pueden ser de transporte, alojamiento y manutención y se deseará almacenar el código del gasto y su valor. Del transporte se deseará almacenar el medio (coche, autobús, tren, avión) y el número de kilómetros (si van en coche). De la manutención si ha sido desayuno, comida o cena.

6.2.1 Diseño conceptual: Esquema E/R

Siguiendo la descripción dada en el apartado *DISEÑO CONCEPTUAL: ESQUEMA E/R* del Problema 1, pasamos a hacer el esquema E/R del Problema 2. Se seguirá la misma estructura y dinámica, se desglosará el enunciado en grupos de supuestos, se analizarán y se realizarán subdiagramas E/R hasta llegar a un diagrama E/R final de todo el enunciado, que unido con los supuestos semánticos no contemplados en el diagrama darán como resultado el esquema E/R.

DISCUSIÓN DEL ENUNCIADO

PARTE 1

Una editorial dedicada a la venta de libros de texto en los centros escolares quiere tener una base de datos donde registrar la actividad comercial de su red.

La editorial visita los centros escolares a través de una red comercial formada por dos tipos de vendedores: contratados y promotores. Los comerciales contratados poseen un contrato con la editorial. El contrato se identifica por un código, una fecha de alta y de baja. Además, cada comercial contratado posee un coche de empresa cuyo modelo depende de su categoría. Respecto a los promotores, estos están dados de alta como autónomos y utilizan su propio coche. Solo se desea almacenar información sobre el modelo de coche de los empleados contratados. Los promotores están supervisados por un vendedor contratado. Cada contratado no puede supervisar a más de tres promotores.

Los datos que se quieren recoger de los vendedores son su código que les identifica, nombre y apellidos, NIF, tipo de vendedor (contratado o autónomo), categoría (1, 2 ó 3 para los contratados y A, B o C para los promotores) y correo personal de empresa

Se identifica como entidad VENDEDOR. Además, se detectan las entidades CONTRATADO y PROMOTOR. Tanto contratados como promotores son personas que trabajan de vendedores en la editorial con la misma información a almacenar (código, nombre, NIF, tipo de vendedor y correo personal). Sin embargo, se interrelacionan de distinta manera, como se verá más adelante, con la relación **Supervisa**, y tienen atributos propios sobre todo en el caso de contratado. Por ello se creará una generalización donde la entidad VENDEDOR es el supertipo de la jerarquía y las entidades CONTRATADO y PROMOTOR son los subtipos de la misma.

Como las únicas personas vendedoras de la editorial que aparecen en la semántica del enunciado son los contratados y promotores, consideraremos esta jerarquía como total y además exclusiva, ya que un vendedor o tiene contrato y es de tipo contratado, o será promotor.

El atributo *Codigo* será Identificador Principal (AIP) del supertipo VENDEDOR. Además, estará el atributo *Nombre*, compuesto de los atributos *Nombre* y *Apellidos*, y los atributos *NIF* y *Correo*. Atributos Identificadores Alternativos (AIA), ya que el NIF es único y el correo se indica en el enunciado, que es personal. El atributo *Categoría* no será un atributo del supertipo, ya que tiene dominio de valores distinto para cada uno de los subtipos. El atributo discriminante será *Tipo*, que tomará valores en el dominio *TIPO_VENDEDOR* = {*contratado, autónomo*}.

El subtipo *PROMOTOR* tendrá como único atributo propio *Categoría* que tomará valores en el dominio *CATEGORIA_PROMOTOR* = {A, B, C}. Por otro lado, el subtipo *CONTRATADO* tendrá como atributo *Modelo_coche* y *Categoría* que tomará valores en el dominio *CATEGORIA_CONTRATADO* = {1, 2, 3}. El modelo de coche dependerá de la categoría, pero esta semántica no es posible reflejarla en el diagrama. Por último, *CONTRATADO* tendrá un atributo “Contrato” de tipo compuesto de los atributos *Fecha_baja* y *Fecha_alta*. En la semántica del enunciado no viene precisado si siempre debe haber una fecha de baja de contrato, considerándose esta cuestión como un supuesto adicional al enunciado a aclarar para definir el atributo *Fecha_baja* como atributo obligatorio u opcional.

Se detecta como relación **Supervisa** que interrelaciona las entidades subtipos de la jerarquía *CONTRATADO* y *PROMOTOR* con cardinalidad 1:3, ya que los promotores están supervisados por un vendedor tipo contratado y cada contratado no puede supervisar a más de tres promotores. Lo que no queda explícito en la semántica del enunciado es la cardinalidad mínima de cuántos promotores supervisa un contratado, es decir, no se precisa si un contratado puede estar sin supervisar a ningún promotor.

De esta forma, como diseñadores nos encontramos con las siguientes cuestiones a dar respuesta, que darán como resultado supuestos que considerar:

1. ¿El contrato de un vendedor tipo contratado puede ser de carácter indefinido?
2. ¿Un contratado supervisa al menos a un promotor?

La Figura 6.14 muestra el diagrama E/R con los supuestos comentados y quedan indeterminados elementos denotados con “?”.

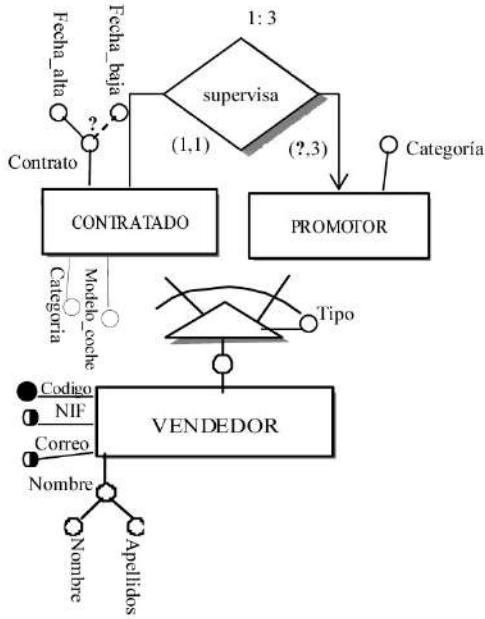


Figura 6.14. Subdiagrama E/R incompleto correspondiente a la parte 1 del enunciado

Tras dar respuesta a las cuestiones detectadas, hay que considerar los siguientes supuestos complementarios al enunciado:

1. Sí puede haber contrato de vendedores de carácter indefinido y, por consiguiente, puede no haber fecha de baja.
2. Pueden existir vendedores contratados que no supervisen a ningún vendedor promotor.

Después de estos supuestos adicionales tenemos que el atributo *Fecha_baja* de **CONTRATADO** puede ser opcional, y queda precisada la cardinalidad mínima de contratado que **Supervisa** a promotor, porque un contratado puede no estar supervisando a ningún promotor.

Así, el diagrama E/R quedará como muestra la Figura 6.15:

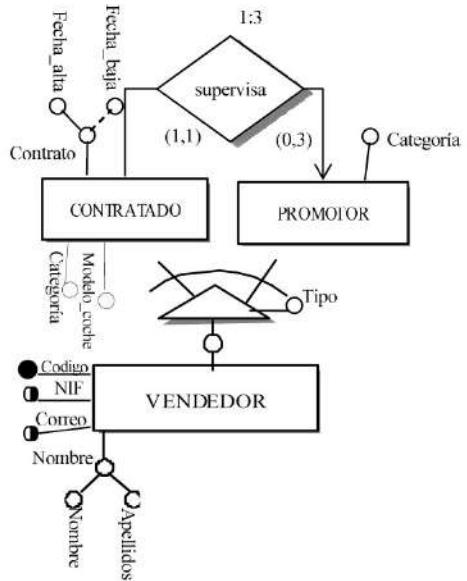


Figura 6.15. Subdiagrama E/R correspondiente a la parte 1 del enunciado

En este subdiagrama E/R hay semántica que no se refleja, dando lugar a los siguientes supuestos semánticos complementarios al diagrama:

- SS1. La *Fecha_alta < Fecha_baja* de *Contrato* en **CONTRATADO**.
- SS2. El atributo *Tipo* de la entidad **VENDEDOR** tomará valores en el dominio *TIPO_VENDEDOR* = {*contratado, autónomo*}.
- SS3. El atributo *Categoría* de la entidad **PROMOTOR** tomará los valores en el dominio *CATEGORIA_PROMOTOR* = {A, B, C}.
- SS4. El atributo *Categoría* de la entidad **CONTRATADO** tomará los valores en el dominio *CATEGORIA_CONTRATADO* = {1, 2, 3}.
- SS5. El valor que tome el atributo *Modelo_coche* en **CONTRATADO** dependerá de la categoría.

PARTE 2

La editorial cuenta con varias delegaciones comerciales dedicadas al suministro de los centros escolares. Los centros pertenecen a una determinada delegación y están identificados por un código, y se desea almacenar de ellos el nombre del centro, dirección (calle, número, población, provincia, código postal), titularidad del centro (público, religioso o laico) y la delegación a la que pertenece. Cada delegación tiene un nombre y un correo propio. Cada vendedor pertenece a una única

delegación comercial y tiene asignados una serie de centros (al menos uno) de su delegación a los que acude regularmente. Los centros pueden estar asignados durante un curso académico a varios vendedores (al menos uno).

El vendedor establece una ruta comercial visitando centros durante el día y cuando llega a casa tiene que hacer el informe de la actividad comercial realizada. Se desea almacenar información histórica sobre las visitas que ha hecho cada vendedor a cada centro. Las visitas de los vendedores se identifican en cada centro por la fecha y la hora de inicio, deseando almacenar también información sobre la hora de fin si se conoce y una descripción de la visita.

Se detectan las entidades *CENTRO*, *DELEGACION* y *VISITA*. La entidad *CENTRO* tiene como AIP a *Codigo*, y como resto de atributos: *Nombre*, *Titularidad* que tomará los valores del dominio *TIPO_TITULARIDAD* = {*publico, religioso*} y el atributo *Direccion*, compuesto de los atributos *Calle*, *No*, *Poblacion*, *Provincia* y *Codigo_Postal*.

En el enunciado se indica que los centros pertenecen a una determinada delegación, así que habrá una relación **Pertenece** que recoja esta semántica e interrelacione las entidades *CENTRO* y *DELEGACION*. Esta relación de cardinalidad 1:N puede considerarse con una dependencia en existencia donde *CENTRO* será entidad débil y *DELEGACION* fuerte. Queda por determinar una cardinalidad mínima, ya que no se precisa si hay delegaciones que no tengan centros. Como la delegación tiene un nombre y correo propio, la entidad *DELEGACION* tiene el atributo *Nombre* como AIP y *Correo* como AIA.

Hay que recoger la asignación de vendedores a delegaciones y centros, pero como un centro solo puede pertenecer a una delegación, basta con interrelacionar *VENDEDOR* con *CENTRO* a través de la relación **Asignado**. La relación **Asignado** tiene cardinalidad N:M porque según el enunciado: cada vendedor tiene asignados una serie de centros (al menos uno), y los centros pueden estar asignados a varios vendedores (al menos uno).

Como atributos de la entidad *VISITA* están *Fecha* y *Hora_inicio*, que serán AIP junto con el código del centro al que pertenece, ya que en el enunciado se dice que “se identifican en cada centro por la fecha y la hora de inicio”. Por esta fuerte dependencia se considerará la relación en identificación **Realiza** entre *CENTRO* (entidad fuerte) y *VISITA* (entidad débil). Además, *VISITA* tiene los atributos *Hora_fin*, que será opcional, y *Descripcion*.

Para almacenar información histórica sobre las visitas que ha hecho cada vendedor a cada centro, se tiene la relación **Visita**, que interrelaciona las entidades **VENDEDOR** y **VISITA**. La cardinalidad de esta relación está sin determinar hasta la parte 2 del enunciado, no se sabe si puede haber visitas de varios vendedores al mismo tiempo, pero si avanzamos en la lectura del enunciado, se indica que varios vendedores pueden ir juntos a una visita. Por otro lado, queda sin determinar si todo vendedor ha realizado al menos una visita.

Así, quedan las siguientes cuestiones a dar respuesta:

1. ¿Cada delegación tiene al menos un centro que le pertenece?
2. ¿Los vendedores han realizado una visita como mínimo?

La Figura 6.16 muestra el diagrama E/R con los supuestos comentados, y quedan indeterminados elementos denotados con “?”.

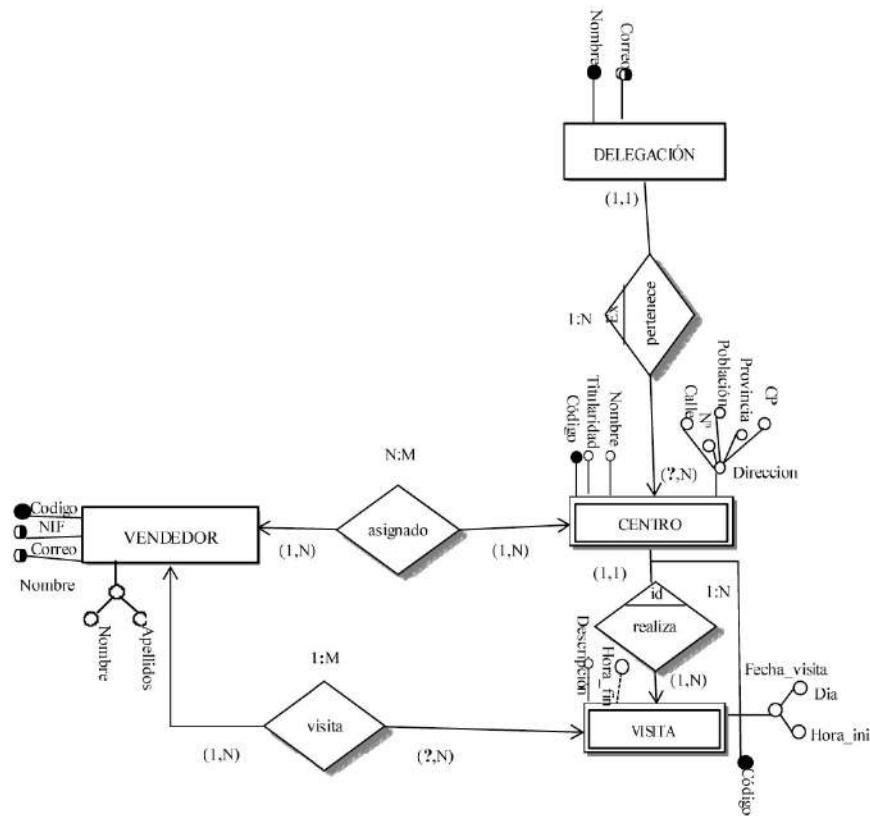


Figura 6.16. Subdiagrama E/R incompleto correspondiente a la parte 2 del enunciado

Tras dar respuesta a las cuestiones detectadas, hay que considerar los

siguientes supuestos complementarios al enunciado:

1. Las delegaciones almacenadas en la base de datos al menos tendrán un centro.
2. Cada uno de los vendedores almacenados en la base de datos habrá realizado al menos una visita.

Con estos supuestos se pueden completar las cardinalidades mínimas que estaban sin determinar, obteniéndose como resultado el diagrama E/R como muestra la Figura 6.17.

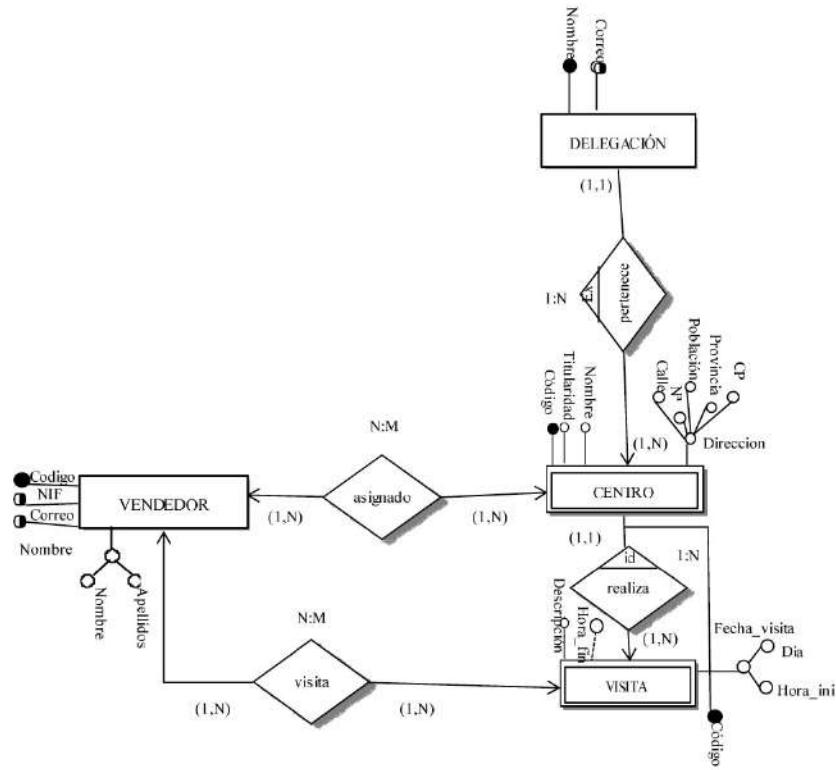


Figura 6.17. Subdiagrama E/R correspondiente a la parte 2 del enunciado

En este subdiagrama E/R hay semántica que no se refleja, dando lugar a los siguientes supuestos semánticos complementarios al diagrama:

- SS6. La *Hora_inicio < Hora_fin* sin *Hora_fin* distinto de nulo en **VISITA**.
- SS7. El atributo *Titularidad* de la entidad **CENTRO** tomará valores en el dominio *TIPO_TITULARIDAD* = {*publico, religioso*}.

- SS8. Los vendedores visitan solo los centros que tienen asignados.
- SS9. Todos los centros de un vendedor han de pertenecer a la misma delegación, delegación a la que él pertenece.
- SS10. La *Fecha_visita* de la entidad VISITA debe ser mayor que la *Fecha_alta* y menor que la *Fecha_baja* si fuera distinta de nulo de la entidad CONTRATADO, para los vendedores de tipo contratado que hagan visitas.

PARTE 3

Será deseable también almacenar información sobre la totalidad de los gastos asociados a la visita. A una visita pueden acudir varios vendedores al mismo tiempo de los que se deseará almacenar los gastos asociados a la visita de cada uno de ellos si los hubiera.

Los gastos pueden ser de transporte, alojamiento y manutención y se deseará almacenar el código del gasto y su valor. Del transporte se deseará almacenar el medio (coche, autobus, tren, avión) y el número de kilómetros (si van en coche). De la manutención, si ha sido desayuno, comida o cena.

Según la semántica la entidad VISITA tiene como atributoa *Total_gastos* que será un atributo derivado (D1), y su valor, semántica no reflejada en el diagrama E/R, será el cálculo de sumar todos los gastos de cada uno de los vendedores para cada una de las visitas.

Hay una nueva entidad GASTO interrelacionada con la entidad VENDEDOR a través de la relación **Hace** que recogerá la semántica de las visitas que hacen los vendedores. Además la entidad GASTO estará interrelacionada con la entidad VISITA a través de la relación **Conlleva** para recoger la información de gastos asociados a esa visita. La relación **Hace** tiene cardinalidad 1:N, ya que un gasto estará asociado a un único vendedor, y cada vendedor cuando visita un centro puede o bien no realizar ningún gasto o bien tener varios que deban ser registrados. La relación **Conlleva** tiene cardinalidad 1:N, ya que para cada visita se pueden generar gastos que hay que registrar.

Los gastos pueden ser de distinto tipo: transporte, alojamiento y manutención, semántica que podría modelarse como una generalización o dejarlo en una única relación. En este caso, como hay dos de los tipos que tienen información propia que recoger, se ha optado por crear una jerarquía donde

GASTO será la entidad supertipo, y *TRANSPORTE*, *ALOJAMIENTO* y *MANUTENCION* serán entidades subtipo. La Jerarquía es total, porque no se contemplan gastos de otro tipo a transporte, alojamiento y manutención, y exclusiva porque el gasto será de un tipo u otro, pero no podrá ser de dos o tres tipos al mismo tiempo.

El atributo *Codigo* será el AIP de la entidad supertipo *GASTO*, que además tendrá el atributo *Precio*. El atributo discriminante *Tipo*, tomará valores en el dominio *TIPO_GASTO* = {*transporte*, *alojamiento*, *manutención*}. El subtipo *TRANSPORTE* tendrá como atributos propios *Km* que será opcional, ya que solo tendrá valor cuando se utilice como medio el coche, y el atributo *Medio* que tomará los valores del dominio *TIPO_MEDIO* = {*coche*, *autobús*, *tren*, *avion*}. El subtipo *MANUTENCION* tiene como único atributo “*Hora*” que tomará los valores del dominio *TIPO_MANUTENCION*= {*desayuno*, *comida*, *cena*}.

En esta parte del enunciado, salvo aclarar que los gastos entre los vendedores en una visita no se comparten, no es necesario considerar supuestos complementarios al enunciado. La Figura 6.18 muestra la propuesta de solución final.

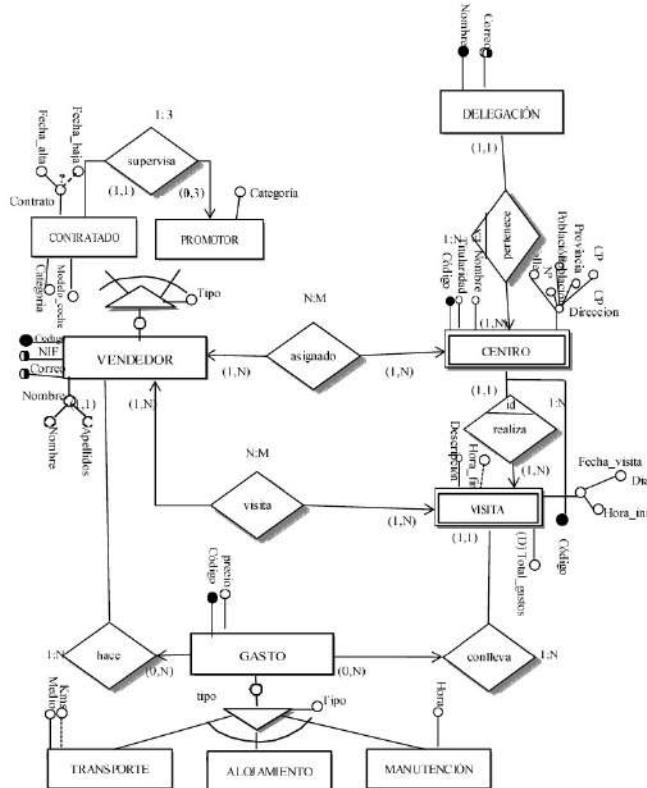


Figura 6.18. Diagrama E/R propuesta solución final

Como supuestos semánticos no contemplados en el diagrama E/R, añadir de la parte 3 del enunciado:

- SS11. El valor calculado del atributo derivado (D1) *Total_gastos* de la entidad *VISITA*, que se calculará a partir de los gastos que se hayan dado en esa visita, para ello se sumarán los gastos de tipo transporte, alojamiento y manutención si los hubiera.
- SS12. El atributo *Tipo* de la entidad *GASTO* tomará valores en el dominio
- $TIPO_GASTO = \{transporte, alojamiento, manutención\}$.
- SS13. El atributo *Medio* de la entidad *TRANSPORTE* tomará valores en el dominio $TIPO_MEDIO = \{coche, autobús, tren, avión\}$.
- SS14. Si $TRANSPORTE.medio = coche$, entonces el atributo *kms* no puede ser opcional.

Como propuesta de solución se obtiene el esquema E/R, compuesto del diagrama E/R representado en la Figura 6.2.5, junto con los supuestos SS1 al SS14 complementarios al diagrama.

6.2.2 Diseño Lógico: Transformación al Esquema Relacional

Siguiendo la información dada en el apartado DISEÑO LÓGICO: TRANSFORMACIÓN AL ESQUEMA RELACIONAL dada en el primer problema, partiendo del esquema conceptual del bloque anterior formado por el diagrama E/R y supuestos semánticos no reflejados en el diagrama, se va a pasar a realizar una transformación a un esquema lógico (grafo relacional), indicando por medio de *checks*, aserciones y disparadores la semántica que no se pueda recoger en el grafo.

En la transformación al esquema relacional se procede como sigue: cada entidad del diagrama E/R (Figura 6.18) se transforma en una relación; los atributos univalueados y simples se transforman en columnas de la relación; los atributos compuestos se descomponen y se transforman en distintas columnas de la relación; cada Atributo Identificador Principal (AIP) de cada entidad se convertirá en la clave primaria de la relación; cada Atributo Identificador Alternativo (AIA) se transforma en clave alternativa en la relación.

En cuanto al resto de constructores del modelo E/R (atributos

multivaluados, interrelaciones, jerarquías, etc.) su transformación se detalla a continuación, especificando la semántica que se pierde en la transformación al grafo relacional mediante *checks*, aserciones y disparadores.

Si para las restricciones de integridad referencial de clave ajena no se especifica un tratamiento específico, se van a considerar como borrado sin acción y modificación en cascada (DNA/UC) debido a que son las menos restrictivas. En la explicación de la solución adoptada solo se indicará otro tipo de restricción de integridad cuando sea necesario.

Se irán haciendo transformaciones de partes del diagrama E/R global, hasta construir el grafo relacional global.

PARTE 1

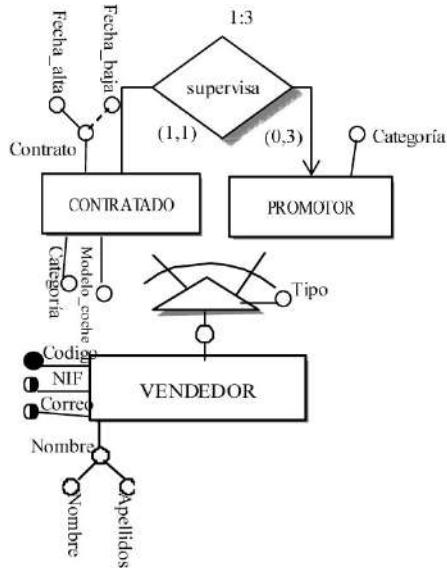


Figura 6.19. Subdiagrama E/R de la parte 1 a transformar

El subdiagrama E/R de la Figura 6.19, se transforma en el grafo relacional de la Figura 6.20.

La jerarquía **VENDEDOR** da lugar a tres nuevas relaciones, una por cada entidad de la jerarquía: **VENDEDOR**, **PROMOTOR** y **CONTRATADO**. Se ha optado por esta estrategia de transformación, ya que las tres se relacionan de distinta manera. Las relaciones (**PROMOTORES** y **CONTRATADOS**) que provienen de los subtipos **PROMOTOR** y **CONTRATADO** tendrán como clave primaria la clave primaria `código` de la relación **VENDEDORES**, y serán

consideradas como claves ajenas referenciando a la relación **VENDEDORES** con opciones de borrado y modificación de las claves ajenas en cascada (DC/UC), debido a su dependencia en existencia respecto a las ocurrencias en la relación **VENDEDORES**. El atributo discriminante *Tipo* de la jerarquía pasa a formar parte como columna *tipo_vendedor* en la relación **VENDEDORES**. Como se trata de una jerarquía total, esta columna no podrá tomar valores nulos (no será opcional). Además, como se trata de una jerarquía exclusiva, la columna *tipo_vendedor* solo puede tomar valores simples de los subtipos de la jerarquía que habrá que comprobar mediante una aserción y un *check* debido a que esta semántica será imposible de almacenar en el grafo relacional.

La interrelación **Supervisa** es una interrelación de tipo 1:N, en este caso 1:3. Al transformar la clave primaria de la relación **CONTRATADO** se propaga como clave ajena a la relación **PROMOTOR**.

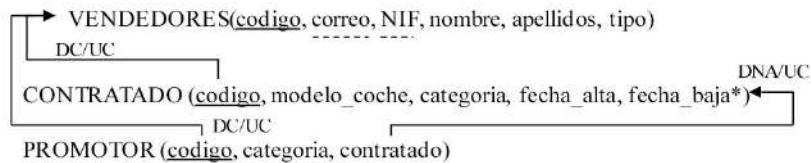


Figura 6.20. Grafo relacional correspondiente a la transformación del subdiagrama E/R parte 1

PARTE 2

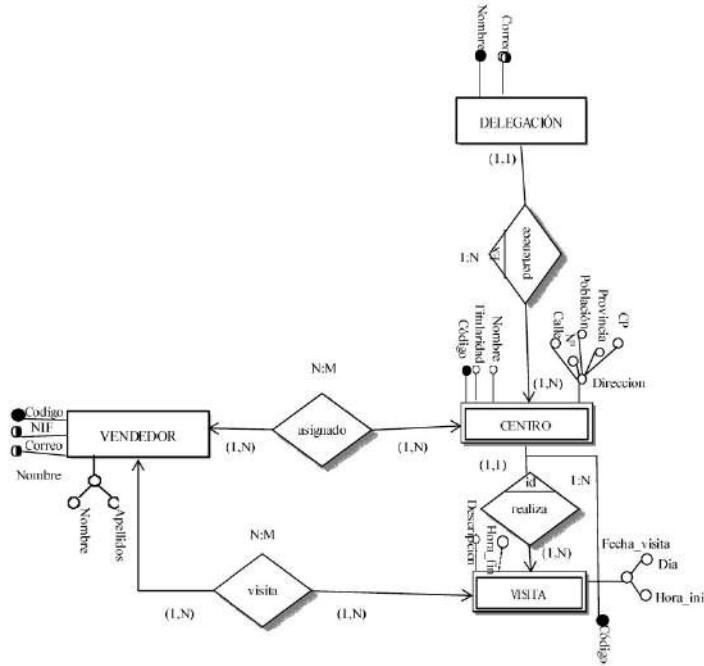


Figura 6.21. Subdiagrama E/R de la parte 2 a transformar

El subdiagrama E/R de la Figura 6.21, se transforma en el grafo relacional de la Figura 6.22.

La interrelación **Pertenece** es una interrelación en existencia, por lo que se transforma mediante una propagación de clave de la relación *DELEGACIONES* a la relación *CENTROS*. La interrelación **Realiza** es una interrelación en identificación, por lo que se transforma mediante una propagación de clave de la relación *CENTRO* que pasará a formar parte de la clave primaria de la relación *VISITA*. En ambos casos, por la fuerte dependencia, las restricciones de integridad referencial se tratarán como borrado y modificación en cascada (DC/UC). En la transformación hay semántica de las cardinalidades que se pierde, por ello habrá que comprobar con aserciones que toda delegación tiene al menos un centro y que para todo centro se realiza al menos una visita.

La interrelación **Asignado** es una interrelación N:M que se transforma como una nueva relación *ASIGNACIONES* formada por las claves primarias de las relaciones *CENTRO* y *VENDEDOR* (tratadas como claves ajenas en esta relación), donde estas claves ajenas formarán parte de la clave primaria de la relación *ASIGNACIONES*. De igual manera **Visita** se transformará en una nueva relación *VISITAS_VENDEDORES* donde su clave primaria estará formada por las claves primarias de las relaciones *VENDEDORES* y *VISITAS* tratadas como claves ajenas. En ambos casos, al haber cardinalidades mínimas de 1, hay pérdida de semántica al transformar al grafo relacional. Será necesario comprobar la siguiente semántica por medio de aserciones: todo vendedor tiene un centro asignado, todo centro tiene asignado al menos un vendedor, todo vendedor realiza al menos una visita y para cada visita al menos la hace un vendedor.

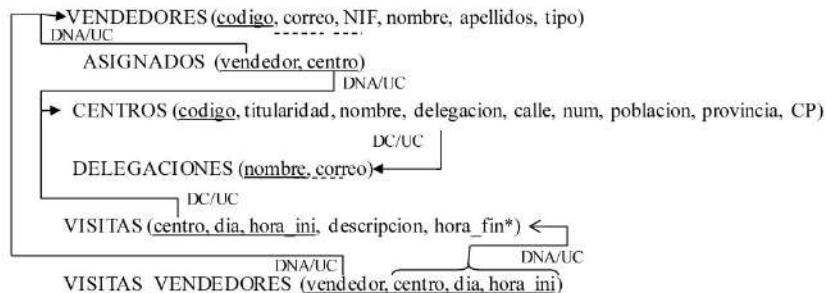


Figura 6.22. Grafo relacional correspondiente a la transformación del subdiagrama E/R parte 2

PARTE 3

En el grafo relacional no es posible representar la semántica de cómo

calcular el valor de un atributo derivado, por lo que será necesario incluir un disparador para la columna *gastos_total* de la tabla *VISITAS*.

Para transformar la jerarquía *GASTO*, se pueden seguir distintas estrategias de transformación. Se ha optado por transformar en tres nuevas relaciones: una de la entidad supertipo *GASTO* y una por cada entidad subtipo de la jerarquía con atributos propios, así se crean las relaciones *TRANSPORTES* y *MANUTENCIONES*, el alojamiento se recogerá en la relación *GASTOS*. Las relaciones (*TRANSPORTES* y *MANUTENCIONES*) tendrán como clave primaria la clave primaria *código* de la relación *GASTOS*, y serán consideradas como claves ajena referenciando a la relación *GASTOS* con opciones de borrado y modificación de las claves ajena en cascada (DC/UC). El atributo discriminante *Tipo* de la jerarquía pasa a formar parte como columna *tipo_gasto* en la relación *GASTOS*. Como se trata de una jerarquía total, esta columna no podrá tomar valores nulos (no será opcional). Además, como se trata de una jerarquía exclusiva, la columna *tipo_gasto* solo puede tomar valores simples de los subtipos de la jerarquía que habrá que comprobar mediante una aserción y un *check* debido a que esta semántica será imposible de almacenar en el grafo relacional.

La interrelación **Hace** es una interrelación de tipo 1:N, y como transformación se va a propagar la clave primaria de la relación *VENDEDORES* como clave ajena a la relación *GASTOS*. De igual manera, la interrelación **Conlleva** se transforma con una propagación de la clave primaria de la relación *VISITAS* a *GASTOS* como clave ajena.

Con las transformaciones descritas de este último bloque se ofrece una propuesta de solución del esquema relacional, que se compone del grafo relacional indicado en la Figura 6.23, junto con *checks*, aserciones y disparadores que se han ido indicando, y que son necesarios para que no haya pérdida de semántica en la transformación del diagrama E/R al relacional.

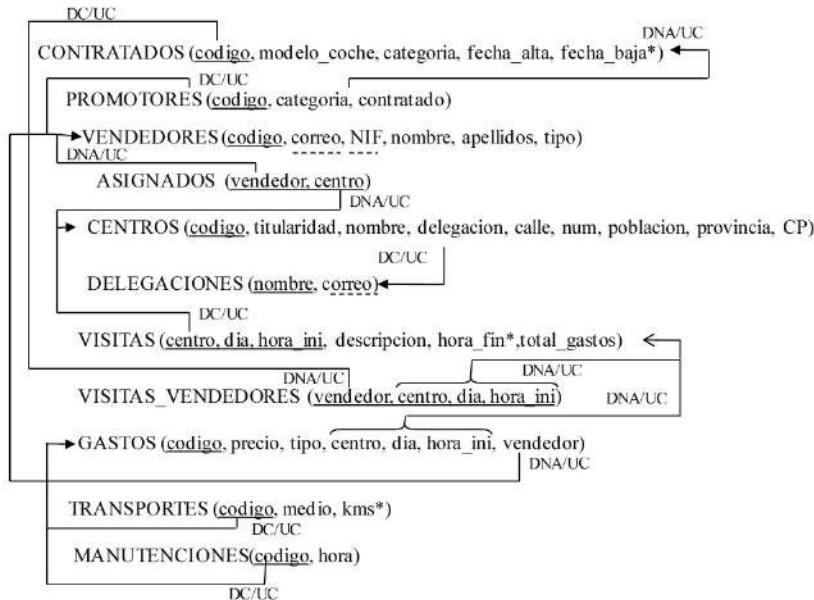


Figura 6.23. Grafo relacional final

6.2.3 Diseño Lógico Específico

En el bloque anterior se ha obtenido el esquema relacional. Como se ha ido comentando hay supuestos que es necesario controlar con mecanismos externos al diagrama o grafo relacional como *checks*, aserciones y disparadores. Entre los supuestos semánticos a incluir al grafo relacional, tal como se indica la Tabla 6.2, hay que considerar:

- Los supuestos que no estaban reflejados en el diagrama E/R (SS1-SS14), y que tampoco se han reflejado en el grafo relacional.
- Los supuestos semánticos para evitar la pérdida de semántica en la transformación del diagrama E/R al relacional.

En el material adicional del libro se proporciona la implementación del modelo relacional en el SGBD ORACLE, introducido en el Capítulo 5.

SUPUESTOS SEMÁNTICOS NO REFLEJADOS EN EL DIAGRAMA E/R		MECANISMO ORACLE
SS1	La <i>Fecha_alta < Fecha_baja</i> de Contrato en <i>CONTRATADO</i> .	<i>CHECK</i> en la tabla <i>CONTRATADOS</i> .
SS2	El atributo <i>Tipo</i> de la entidad <i>VENDEDOR</i> tomará valores en el dominio <i>TIPO_VENDEDOR</i> = {contratado, autónomo}.	<i>CHECK</i> en la tabla <i>VENDEDORES</i> .
SS3	El atributo <i>Categoría</i> de la entidad <i>PROMOTOR</i> tomará los valores en el dominio <i>CATEGORIA_PROMOTOR</i> = {A, B, C}.	<i>CHECK</i> en la tabla <i>PROMOTORES</i> .
SS4	El atributo <i>Categoría</i> de la entidad <i>CONTRATADO</i> tomará los valores en el dominio <i>CATEGORIA_CONTRATADO</i> = {1, 2, 3}.	<i>CHECK</i> en la tabla <i>CONTRATADOS</i> .
SS5	El valor que tome el atributo <i>Modelo_coche</i> en <i>CONTRATADO</i> dependerá del valor de <i>Categoría</i> .	<i>CHECK</i> en la tabla <i>CONTRATADOS</i> .

SS6	La <i>Hora_inicio < Hora_fin</i> sin <i>Hora_fin</i> distinto de nulo en <i>VISITA</i> .	<i>CHECK</i> en la tabla <i>VISITAS</i> .
SS7	El atributo <i>Titularidad</i> de la entidad <i>CENTRO</i> tomará valores en el dominio <i>TIPO_TITULARIDAD</i> = {"publico", religioso"}.	<i>CHECK</i> en la tabla <i>CENTROS</i> .
SS8	Los vendedores visitan solo los centros que tienen asignados.	<i>TRIGGER</i> antes de insertar <i>VISITAS_VENDEDORES</i> , consultar la tabla <i>ASIGNADOS</i> .
SS9	Todos los centros de un vendedor han de pertenecer a la misma delegación, delegación a la que él pertenece.	<i>TRIGGER</i> antes de insertar y/o modificar en <i>ASIGNADOS</i> , consultar las tablas: <i>ASIGNADOS</i> , <i>CENTROS</i> y <i>DELEGACIONES</i> .
SS10	La <i>Fecha_visita</i> de la entidad <i>VISITA</i> debe ser mayor que la <i>Fecha_alta</i> y menor que <i>Fecha_baja</i> si fuera distinta de nulo de la entidad <i>CONTRATADO</i> , para los vendedores de tipo contrato que hagan visitas.	<i>TRIGGER</i> antes de insertar en <i>VISITAS</i> , consultar la tabla <i>CONTRATADOS</i> .
SS11	El valor calculado del atributo derivado (D1) <i>Total_gastos</i> de la entidad <i>VISITA</i> , que se calculará a partir de los gastos que se hayan dado en esa visita, para ello se sumarán los gastos de tipo transporte, alojamiento o manutención si los hubiera.	<i>TRIGGER</i> después de insertar un gasto (en <i>GASTOS</i> , <i>TRANSPORTES</i> y <i>MANUTENCIONES</i>) actualice el valor <i>Total_gastos</i> en la tabla <i>VISITAS</i> .
SS12	El atributo <i>Tipo</i> de la entidad <i>GASTO</i> tomará valores en el dominio <i>TIPO_GASTO</i> = {transporte, alojamiento, manutención}.	<i>CHECK</i> en la tabla <i>GASTOS</i> .

SS13	El atributo <i>Medio</i> de la entidad TRANSPORTE tomará valores en el dominio TIPO_MEDIO = {"coche", "autobús", "tren", "avion"}.	<i>CHECK</i> en la tabla TRANSPORTES.
SS14	Si TRANSPORTE.medio = 'coche', entonces el atributo 'kms' no puede ser opcional.	<i>CHECK</i> en la tabla TRANSPORTES.
SUPUESTOS PÉRDIDA SEMÁNTICA TRANSFORMACIÓN		MECANISMO ORACLE
Comprobación exclusividad de la jerarquía VENDEDOR.		<i>CHECK</i> en la tabla VENDEDOR.
		<i>TRIGGER</i> antes de insertar sobre la tabla CONTRATADOS, consulta sobre la tabla PROMOTORES.
		<i>TRIGGER</i> antes de insertar sobre la tabla PROMOTORES, consulta sobre la tabla CONTRATADOS.
Un contratado supervisa como máximo a tres promotores.		<i>TRIGGER</i> antes de insertar y/o modificar en PROMOTORES, consulta sobre tabla PROMOTORES.
Toda delegación hay al menos un centro que le pertenece.		<i>TRIGGER</i> que controle que toda delegación debe aparecer al menos una vez en la tabla CENTROS.
En todo centro se realiza al menos una visita.		<i>TRIGGER</i> que controle que todo centro debe aparecer al menos una vez en la tabla VISITAS.

Todo vendedor tiene un centro asignado, y todo centro tiene asignado al menos un vendedor.	<i>TRIGGER</i> que controle que para cada vendedor tiene asignado un centro en la tabla <i>ASIGNACIONES</i> , y que todo centro tiene asignado un vendedor en la tabla <i>ASIGNACIONES</i> .
Todo vendedor realiza al menos una visita y para cada visita al menos la hace un vendedor.	<i>TRIGGER</i> que controle que todo vendedor tiene una visita asociada en la tabla <i>VISITAS_VENDEDORES</i> , y que toda visita tiene un vendedor que la realiza en la tabla <i>VISITAS-VENDEDORES</i> .
	<i>CHECK</i> en la tabla <i>GASTOS</i> .
Comprobación exclusividad de la jerarquía <i>GASTO</i> .	<i>TRIGGER</i> antes de insertar sobre la tabla <i>TRANSPORTES</i> , consulta sobre las tablas <i>MANUTENCIONES</i> , <i>GASTOS</i> . <i>TRIGGER</i> antes de insertar sobre la tabla <i>MANUTENCIONES</i> , consulta sobre las tablas <i>TRANSPORTES</i> , <i>GASTOS</i> .

Tabla 6.2. Tabla supuestos semánticos a añadir al grafo relacional

BIBLIOGRAFÍA

- CHEN, P. (1976). *The Entity/Relationship Model: Toward a Unified View of Data*. En: *CACM*, 13, 6.
- COOD (1970). *A Relational Model of Data for Large Shared Data Banks*. En: *CACM*, 1, 1.
- CONNOLLY, T. y BEGG, C (2005). *Sistemas de Bases de Datos (4^a ed)*. Ed. Prentice-Hall.
- DATE, C. J. (2004). *An introduction to Database Systems (8^a ed)*. Ed. Pearson Addison Wesley.
- ELMASRI, R., NAVATHE, S. B. (2008). *Fundamentos de Sistemas de Bases de Datos (3^a ed)*. Ed. Pearson Education.
- FOLK, M. J., ZOELLICK, B, & RICCARDI, G. (1998). *File structures: an object-oriented approach with C++*. Ed. AddisonWesley.
- GARCÍA MOLINA, H., ULLMAN, J., WIDOM, J. (2009). *Database systems: the complete book. (2^a ed)*. Ed. Pearson Education International.
- HANSEN, G.W.; HANSEN, J. V. (1998). *Diseño y Administración de Bases de datos (2^a ed)*. Ed. Prentice-Hall.
- PIATTINI, M., MARCOS, E., CALERO C., VELA B. (2006). *Tecnología y diseño de Bases de Datos*. Ed. Ra-Ma, Madrid.
- SILBERSCHATZ, A. (2007). *Fundamentos de Bases de Datos (5^a ed.)*. Ed. McGraw-Hill.

Descarga de Material Adicional

Este E-book tiene disponible un material adicional que complementa el contenido del mismo.

Este material se encuentra disponible en nuestra página Web www.ra-ma.com.

Para descargarlo debe dirigirse a la ficha del libro de papel que se corresponde con el libro electrónico que Ud. ha adquirido. Para localizar la ficha del libro de papel puede utilizar el buscador de la Web.

Una vez en la ficha del libro encontrará un enlace con un texto similar a este:

“Descarga del material adicional del libro”

Pulsando sobre este enlace, el fichero comenzará a descargarse.

Una vez concluida la descarga dispondrá de un archivo comprimido. Debe utilizar un software descompresor adecuado para completar la operación. En el proceso de descompresión se le solicitará una contraseña, dicha contraseña coincide con los 13 dígitos del ISBN del libro de papel (incluidos los guiones).

Encontrará este dato en la misma ficha del libro donde descargó el material adicional.

Si tiene cualquier pregunta no dude en ponerse en contacto con nosotros en la siguiente dirección de correo: ebooks@ra-ma.com

ÍNDICE ALFABÉTICO

A

- Acceso invertido 357, 392, 393
Agrupación 140
Álgebra relacional 138
Almacenamiento secundario 337
Aplicaciones distribuidas 267
Árbol B 354, 355, 368, 373, 383
Árbol B* 384
Árbol B+ 355, 385, 429
Archivo 338
Área de desbordamiento 347, 348, 366
Área desordenada 343
Arquitectura ANSI/X3/SPARC 269
ASSERTION 126
Atributo 16
Atributo compuesto 17, 131
Atributo derivado 17, 131
Atributo multivaluado 130
Atributo obligatorio 123, 131
Atributo obligatorio/opcional 17
Atributo opcional 131

Atributo univaluado 130

B

Bases de datos distribuidas 265

Bloque 338, 412

Borrado físico 342

Borrado lógico 342

Borrado/Modificación cascada 125

Borrado/Modificación con puesta a valor por defecto 126

Borrado/Modificación no acción 125

Bottom-up 271

Buffer 339

Búsqueda dicotómica 343, 365, 383

Búsqueda dicotómica extendida 343, 372

Byte 338

C

Campos 338

Campos de control 339

Cardinalidad del dominio 345, 359, 371, 372

CHECK 126, 175, 176

Clave 340

Clave ajena 123, 124, 420

Clave alternativa 123, 144, 157

Clave de búsqueda 340

Clave de direccionamiento 340, 345

Clave de identificación 340
Clave de indización 340, 350
Clave de ordenación 340, 342
Clave primaria 123, 144, 173
Codificación de campos 339
Combinación 140
Compactación 364
Coste global 338
Cubo 344
Cúmulo 366

D

Datos locales 266
DBLinks 268
Densidad de ocupación 340, 366
Densidad ideal 340, 363, 390, 396
Densidad real 340, 363, 390
Dependencia en existencia 134
Dependencia en identificación 17, 134
Diferencia 139
Dirección física 345
Dirección real 346
Dirección relativa 345
Dirección virtual 346
Direccionamiento abierto 347
Direccionamiento directo 345

Direccionamiento disperso 347, 357
Directorio de ocurrencias 346
Directorio global 267
Directorio virtual 346
Diseño conceptual 435
Diseño físico 339
Diseño lógico 339, 435
Diseño lógico específico 435
Diseño relacional 435
Disparador 453, 475
Dispersión 371
División 140
Dominio 16, 121, 122, 131

E

Elementos de datos 338
Encadenamiento 348
Encadenamiento de las hojas 355
Entidad 16, 128
Entidad débil 17
Entidad fuerte 17
Entrada 140, 350
Entrada Atributo univaluado multivaluado 17
Entrada Borrado/Modificación con puestas a nulos 126
Espacio de cubo 344, 372

Espacio de direccionamiento 345
Espacio de tabla 411, 412, 423
Espacio libre distribuido 344, 353, 383 Esquema de bits 391
Esquema de bits de puntero implícito 359
Esquema E/R 435
Esquemas de bits 359, 371, 392
Esquemas de bits de puntero implícito 393
Exclusividad/solapamiento 19, 135

F

Factor de bloqueo 338
Fragmentación 272, 273
Fragmentación horizontal 276
Fragmentación HV 278
Fragmentación mixta 278
Fragmentación vertical 274
Fragmentación VH 278
Función de transformación 345

G

Generalizaciones 18
Gestión de desbordamientos 347, 372
Gestión desbordamientos encadenada a cubo 378
Gestión desbordamientos encadenada a registro 378
Grupo repetitivo 338

H

Huecos 342, 363, 383

I

Identificador alternativo 16, 131

Identificador principal 16, 131

Implementación 435

Inclusión de índices 351

Índice 350

Índice exhaustivo 352

Índice multinivel 353

Índice no denso 352

Índice parcial 352

Índice primario 351

Índice secuencial 386

Índice secundario 351

Índice serial 351, 382, 386

Índices invertidos 391

Integridad referencial 125, 128

Interrelación 132, 133

Interrelaciones 17

Interrelaciones ternarias 136

Intersección 153

L P

Lista de punteros 351, 371, 386

Listas invertidas 358, 391

M

Mapa de bits 417

Marcas 339

Memoria intermedia 339, 353

Metodología ascendente 271

Metodología descendente 272

Modelo E/R 15

Modificación cascada 169

Modificación en cascada 181, 213

Multivaluación 359

N

Naturaleza de los procesos 341

O

Ocupación mínima 354, 368

Oracle 267, 435

Oracle Streams 268

Orden del árbol 354, 356, 368

Organización direccional 366

Organización direccional dispersa 377

Organización direccional dispersa
virtual 377, 379

Organización no consecutiva 364

Organización secuencial 342

Organización secuencial consecutiva 376
Organización serial 341, 372
Organización serial consecutiva 375
Organización serial no consecutiva 375
Organizaciones auxiliares 350
Organizaciones base 341, 357
Organizaciones direccionadas 345
Organizaciones no consecutivas. 344
Organizaciones secuencial no
consecutiva 376
PCTFREE 414, 421, 427
PCTUSED 414
Potencia de direccionamiento 345
Procesos selectivos 340
Producto cartesiano 139
Profundidad de un árbol B 346, 355, 358
Profundidad del árbol 373, 391 Profundidad del
árbol B+ 385, 400, 405, 406, 408
Proyección 138, 177
Puntero externo 353, 368, 373, 382, 391
Puntero interno 353, 368, 373

R

Rebotes 366
Recorrido serial 341, 364
Registro 338

Relación 121, 128
Reorganización 365
Replicación 272, 279
Restricción de cardinalidad 17
Restricciones inherentes 122
Restricciones semánticas 123
Rotación 355
ROWID 415

S

Saturación 347
Saturación encadenada 347
Segmento 412
Selección 138
SGBD 266, 461
SGBD multibase 270
Sistemas compuestos 269
Sistemas federados 269
Sistemas multibase de datos 269
Sondeo lineal 366
Soporte secundario 337
Subtipo 18
Supertipo 18

T

Tamaño de bloque 338
Tamaño de cubo 344, 382

Tamaño del registro 338
Tasas de volatilidad 341
Tiempo de acceso a bloque 338
Tipo de correspondencia 17
Top-Down 271
Totalidad/parcialidad 19, 135
TRIGGER 127

U

Unión 139

V

Vector 338
Vistas materializadas 267