

Diseño de bases de datos relacionales

Adoración de Miguel Castaño
Universidad Carlos III de Madrid
Mario Piattini Velthuis
Universidad de Castilla-La Mancha
Esperanza Marcos Martínez
Universidad Rey Juan Carlos

Alfaomega  *za-ma*

Diseño de bases de datos relacionales
© Adoración de Miguel Castaño, Mario Piattini Velthuis,
Esperanza Marcos Martínez

ISBN 84-7897-385-0, edición original publicada por RA-MA Editorial,
MADRID, España. Derechos reservados © RA-MA Editorial

MARCAS COMERCIALES: RA-MA ha intentado a lo largo de este libro distinguir las marcas registradas de los términos descriptivos, siguiendo el estilo de mayúsculas que utiliza el fabricante, sin intención de infringir la marca y sólo en beneficio del propietario de la misma.

© 2000 ALFAOMEGA GRUPO EDITOR, S.A. de C.V.
Pitágoras 1139, Col. Del Valle, 03100 México, D.F.

Miembro de la Cámara Nacional de la Industria Editorial
Registro No. 2317

Internet: <http://www.alfaomega.com.mx>
Email: ventas@alfaomega.com.mx
ISBN 970-15-0526-3

Derechos reservados.

Esta obra es propiedad intelectual de su autor y los derechos de publicación en lengua española han sido legalmente transferidos al editor. Prohibida su reproducción parcial o total por cualquier medio sin permiso por escrito del propietario de los derechos del copyright.

NOTA IMPORTANTE

La información contenida en esta obra tiene un fin exclusivamente didáctico y, por lo tanto, no está previsto su aprovechamiento a nivel profesional o industrial. Las indicaciones técnicas y programas incluidos, han sido elaborados con gran cuidado por el autor y reproducidos bajo estrictas normas de control. ALFAOMEGA GRUPO EDITOR, S.A. de C.V. no será jurídicamente responsable por: errores u omisiones; daños y perjuicios que se pudieran atribuir al uso de la información comprendida en este libro y en el disquete adjunto, ni por la utilización indebida que pudiera dársele.

Edición autorizada para venta en México y todo el continente americano

Impreso en México – Printed in Mexico

CONTENIDO

PRÓLOGO	XV
PREFACIO.....	XVII
PARTE I.....	1
CAPÍTULO 1. MODELO DE DATOS.....	3
1. INTRODUCCIÓN	3
2. MODELO, ESQUEMA Y EJEMPLAR	8
3. TIPOS DE ABSTRACCIÓN EN EL DISEÑO DE BASES DE DATOS	12
3.1. Clasificación/Particularización	13
3.2. Agregación/Desagregación	15
3.3. Generalización/Especialización	18
3.4. Asociación/Disociación	20
3.5. Jerarquías de abstracciones	21
4. CONCEPTO DE MODELO DE DATOS.....	23
4.1. Estática.....	25
4.2. Dinámica.....	26
5. RESTRICCIONES DE INTEGRIDAD	28
5.1. Componentes de una restricción	31
5.2. Clasificación de las restricciones	33
6. LOS MODELOS DE DATOS EN EL PROCESO DE DISEÑO DE UNA BASE DE DATOS.....	38
ANEXO. CLASIFICACIÓN DE LAS RESTRICCIONES.....	42

CAPÍTULO 1

MODELO DE DATOS

En este capítulo se analizan los modelos de datos como herramientas de abstracción que permiten representar la realidad, captando su semántica. Discutimos primero el concepto de modelo y de esquema, para pasar a presentar diferentes tipos de abstracciones que se utilizan en el modelado de datos y a definir la estática y la dinámica de los modelos de datos. Se estudian en profundidad las restricciones semánticas, para terminar viendo el papel que desempeñan los modelos de datos en el diseño de una base de datos.

1. INTRODUCCIÓN

Desde tiempos remotos, los datos han sido registrados por el hombre en algún tipo de soporte (papel, piedra, madera, etc.) a fin de que quedara constancia de un fenómeno o idea. Los **datos** han de ser interpretados (incorporándolos significado) para que se conviertan en **información** útil. Cuando utilizamos el lenguaje natural y decimos, por ejemplo, que una persona ha nacido en 1965, el dato (1965) va acompañado de su interpretación (año de nacimiento de una cierta persona); sin embargo, en la informática, desde sus inicios, se separó el dato de su significado. Por ello, a fin de facilitar la interpretación de los datos, surgen los modelos de datos como instrumentos que ayudan a incorporar significado a los datos.

Según FLORY (1982), “modelar consiste en definir un mundo abstracto y teórico tal que las conclusiones que se puedan sacar de él coincidan con las manifestaciones aparentes del mundo real”. Siendo un *modelo*, “un conjunto de conceptos que permite construir una representación organizacional de la empresa”. Como señalan TSICHRITZIS y LOCHOVSKY (1982), “un modelo de datos es un dispositivo de

abstracción que nos permite ver el bosque (esto es, la información contenida en los datos) en oposición a los árboles (valores individuales de los datos)".

Según el DRAE¹, la *abstracción* es la acción y el efecto de abstraer, "separar por medio de una operación intelectual las cualidades de un objeto para considerarlas aisladamente o para considerar el mismo objeto en su pura esencia o noción". Por tanto, la abstracción, como proceso mental capaz de ocultar detalles y fijarse en lo esencial, busca las propiedades comunes de un conjunto de objetos, reduciendo así la complejidad y ayudando a la comprensión del mundo real.

Los modelos de datos proporcionan mecanismos de abstracción que permiten la representación de aquella parcela del mundo real cuyos datos nos interesa registrar, lo que habitualmente se denomina *universo del discurso* o, en palabras de DITTRICH (1994) *mini-mundo*. Dicha representación se concibe en dos niveles: el de las **estructuras** que hacen posible la representación de la información, y el de la **información** en sí misma. Estos dos niveles dan lugar, en el ámbito de las bases de datos, a la distinción entre **esquema** y **base de datos**; conceptos que DITTRICH (1994) define como: "*La descripción específica de un mini-mundo determinado, en términos de un modelo de datos, recibe el nombre de esquema (esquema de datos o esquema de base de datos) de dicho mini-mundo. La colección de datos que en sí misma representa la información del mini-mundo da lugar a la base de datos*".

Asociados a los modelos de datos están los lenguajes de datos que permiten definir y manipular (consultar y actualizar) la base de datos. En lo que respecta a la relación entre los modelos y los lenguajes de datos, hay que destacar que los modelos son la base para los lenguajes, aunque el nivel de abstracción de estos últimos es menor, ya que el lenguaje es el modelo más una sintaxis. La existencia de distintos lenguajes puede proceder tanto del modelo como de la sintaxis; por ejemplo, el lenguaje SQL es el resultado de aplicar una determinada sintaxis al modelo relacional, mientras que el QUEL es otro lenguaje relacional ya que la sintaxis es distinta aunque el modelo sea el mismo; el OQL es el resultado de asociar a otro modelo (el modelo de objetos –MO–) una cierta sintaxis (ver figura 1.1).

En la arquitectura de una base de datos propuesta por ANSI² –ANSI (1975) y (1978)– se suelen diferenciar tres niveles de abstracción: **Global**³, **Externo** e **Interno**. El nivel global contiene una representación del conjunto de los datos de una organización; en el nivel externo, los datos (en general, sólo una parte de los mismos) se describen para atender las necesidades de uno o varios procesos o de un grupo de usuarios en particular; el nivel interno describe las características de los datos tal como han de encontrarse almacenados físicamente, siendo sus elementos de descripción punteros, índices, agrupamientos, etc.

¹ Diccionario de la Real Academia Española. Vigésima primera edición (1992).

² ANSI es el acrónimo de American National Standard Institute, es decir, la organización oficial de estándares de Estados Unidos.

³ Para ANSI, **Conceptual**; posteriormente haremos la distinción entre global y conceptual.

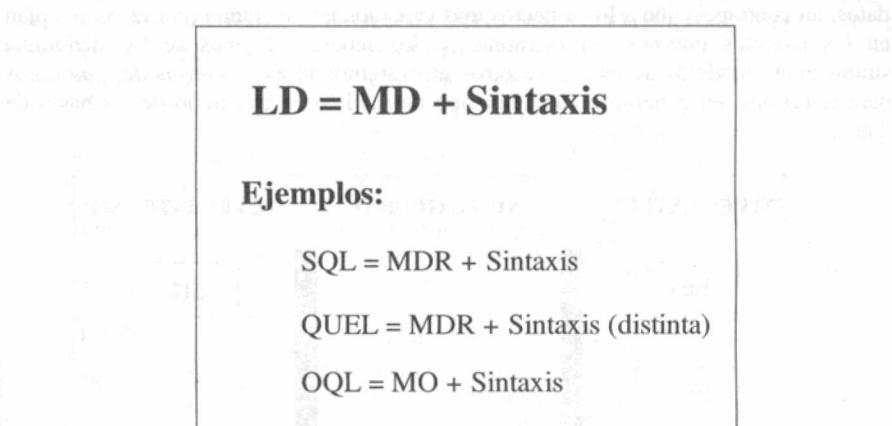


Figura 1.1. Modelos de datos y lenguajes de datos

Existen, por tanto, en una base de datos tres clases de esquemas: el **esquema global**, los **esquemas externos** (tantos como necesiten las aplicaciones), y el **esquema interno** que, en un momento determinado, es único, aunque un mismo esquema global admite distintos esquemas internos entre los cuales se seleccionará aquel que cumpla mejor los requisitos de eficiencia, seguridad, etc.⁴; entre estos tres tipos de esquemas existen dos tipos de funciones de correspondencia (*mapping*), la que permite la transformación esquema global/esquemas externos y la que realiza la transformación esquema global/esquema interno; el Sistema de Gestión de la Base de Datos (SGBD) ha de proporcionar estas funciones de correspondencia. En la figura 1.2 pueden verse los tres tipos de esquemas: Esquemas Externos –EE–, Esquema Global –EG– y Esquema Interno –EI–, el cual puede variar según se va “afinando” la base de datos, pero es único en un momento determinado –EI_x–; también se han presentado en la figura las dos funciones de correspondencia.

Según el nivel de abstracción de la arquitectura a tres niveles en el que se encuentre la estructura descrita, el modelo que permite su descripción será un **modelo global, externo o interno**. De entre los distintos tipos de modelos, son los globales en los que vamos a centrar nuestra atención, ya que los externos suelen utilizar conceptos parecidos a los de los correspondientes globales y los internos, aunque tienen características comunes, realmente no existen como tales modelos ya que son propios de cada producto comercial.

A veces se utiliza la expresión **modelo lógico** para hacer referencia tanto a los modelos globales como externos, ya que ambos describen aspectos lógicos de los

⁴ El esquema interno se va afinando (*tuning*) a fin de conseguir un mejor rendimiento global de las aplicaciones, y más especialmente de las aplicaciones críticas.

datos, en contraposición a los aspectos más cercanos a la máquina que se contemplan en los modelos internos; en ocasiones, a los modelos lógicos se los denomina simplemente modelos de datos. Nosotros utilizaremos la expresión *modelo de datos* para referirnos, en general, a cualquier tipo de modelo en el campo de las bases de datos.

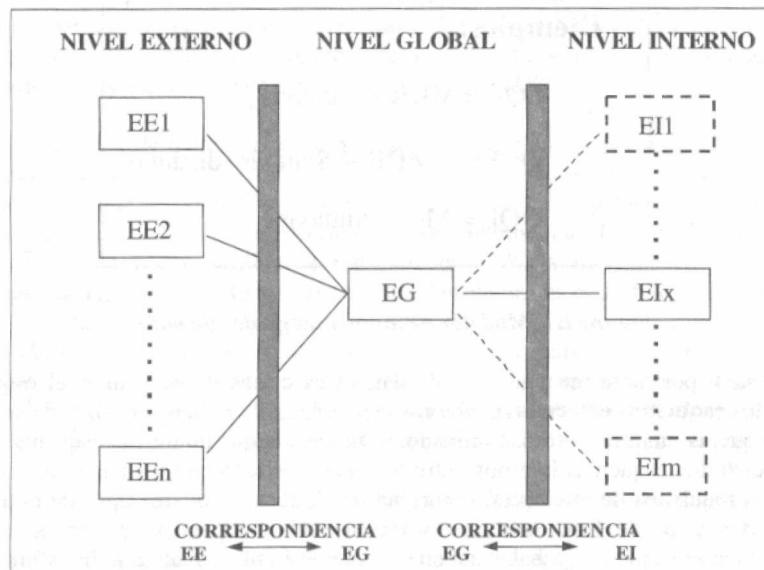


Figura 1.2. Los tres niveles de abstracción de la arquitectura ANSI

En la figura 1.3 se puede ver un ejemplo que describe una pequeña parte de una base de datos para la gestión de los cursos de doctorado de una universidad, donde aparece el esquema global, el esquema interno y dos esquemas externos que describen los datos para dos aplicaciones. En el esquema global tenemos tres tipos de objetos⁵: CURSO, PROFESOR e IMPARTE, que se transforman en registros almacenados en el esquema interno; los dos esquemas externos (uno en SQLForms y otro en Pascal) describen sólo una parte del esquema, aquella que necesitan las correspondientes aplicaciones.

⁵ El término objeto no tiene aquí el significado específico que se le atribuye en la orientación al objeto, sino la acepción del lenguaje común.

a) ESQUEMA GLOBAL	b) ESQUEMA INTERNO	c) ESQUEMAS EXTERNOS
CURSO DE DOCTORADO /* Tipo de Objeto */	CURSOS DE DOCTORADO /* Registro Almacenado */	
CURSO CODIGO Caracter (5) NOMBRE Caracter(50) NUM_HORAS Numérico (3) DESCRIPCION Carácter variable (200) Clave CODIGO	CURSO COD_CURSO Byte (3) NOMBRE Byte (50) NUM_HORAS Byte (2) DESCRIPCION Byte (200) Índice de 2 niveles sobre COD_CURSO	EN ORACLE FORMS (listado de cursos) CODIGO Varchar2 (5) NOMBRE Varchar2 (50) HORAS Number (3,0) DESCRIPCION Varchar2 (200)
PROFESOR CODIGO Caracter (3) NOMBRE Caracter(30) DNI Caracter (10) DIRECCION Caracter(50) SALARIO Numérico (7) Clave CODIGO	PROFESOR COD_PROFE Byte (2) NOMBRE Byte (30) DNI Byte (10) DIRECCION Byte (50) SALARIO Byte (4) Indice 1 nivel sobre COD_PROFE	EN PASCAL (asignación cursos a profesores) CURSO Char (5) NOMBRE Char (30) HORAS Integer(10) COD_PROFE Char (3) PROFESOR Char (30) INICIO String(10) FIN String(10)
IMPARTE PROFESOR Caracter (3) CURSO Caracter (5) FECHA_INICIO Fecha FECHA_FINAL Fecha Clave PROFESOR, CURSO	IMPARTE FECHA_INI Byte (8) FECHA_FIN Byte (8) PUNTERO_CURSO Byte (4) PUNTERO_PROFESOR Byte (4)	

Figura 1.3. Ejemplo de esquema global, esquema interno y dos esquemas externos para una base de datos que describe los cursos de doctorado de una universidad

Los modelos globales se clasifican, a su vez, en conceptuales y convencionales. Los **modelos conceptuales** (también denominados de alto nivel) facilitan la descripción global del conjunto de información de la empresa al nivel más próximo al usuario, por lo que sus conceptos son cercanos al mundo real (entidades, atributos, interrelaciones, etc.); los **modelos convencionales** se encuentran instrumentados en los SGBD y están orientados a describir los datos a nivel lógico para el SGBD (de ahí que también reciban el nombre de modelos lógicos o modelos de bases de datos⁶), por lo que sus conceptos son tablas o relaciones en el caso del modelo relacional, redes en el Codasyl, jerarquías en el jerárquico, etc. En la figura 1.4 se resume esta clasificación de los modelos globales.

⁶ Como podemos apreciar, la terminología no es única y esta escasa precisión terminológica es la causa muchas veces de la confusión de los usuarios, en especial de los que se están iniciando en el estudio de las bases de datos. En PASCAL (1993), apéndice 7A (pgs. 135 a 137) se critica este confusionismo terminológico y, aunque nosotros no estamos totalmente de acuerdo con algunas de sus aseveraciones, recomendamos su lectura así como hacer un estudio comparativo de su punto de vista con el que nosotros proponemos.

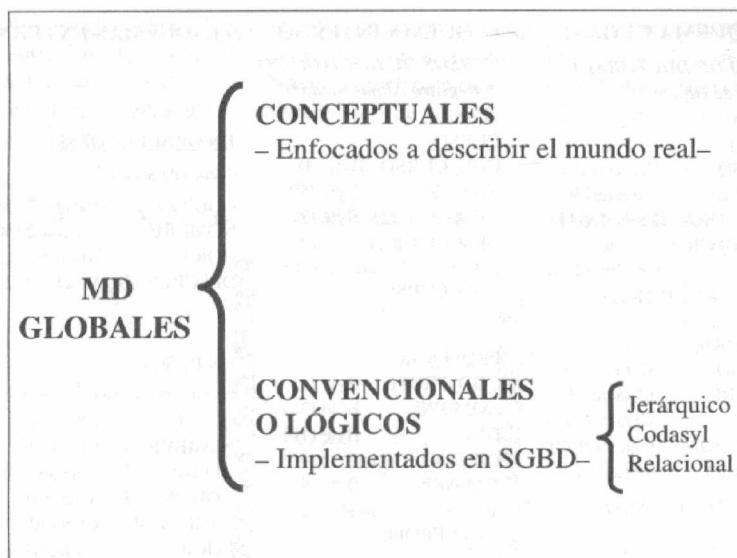


Figura 1.4. Clasificación de los modelos de datos globales

2. MODELO, ESQUEMA Y EJEMPLAR

Para algunos autores, como DITTRICH (1994), la expresión *modelo de datos* no está bien elegida para hacer referencia al concepto que representa en el ámbito de las bases de datos, ya que “... *es menos un modelo en sí mismo, que un marco para concebir modelos (del mundo real)*”⁷.

Puede que esto haya llevado en el campo de la Ingeniería del software, especialmente en las metodologías de desarrollo de sistemas de información y en su práctica, a llamar modelo de datos tanto al instrumento de descripción (lo que realmente es para nosotros el modelo) como al resultado de la misma (para nosotros esquema), sobrecargando así la expresión modelo de datos y aumentando con ello el confusionismo al que acabamos de aludir; la falta de distinción entre modelo y esquema es, en nuestra opinión, bastante perniciosa⁸.

⁷ En la tesis MARCOS (1997), se puede encontrar una interesante discusión sobre la expresión modelo de datos, llegándose a la conclusión de que, a pesar de las consideraciones de DITTRICH, sí es adecuado utilizar dicha expresión con el sentido que se le da en las bases de datos.

⁸ En la metodología MERISE se habla de *formalismos*, en lugar de modelos, como marcos para definir esquemas, ROCHFELD (1992). En Métrica, versión 2.1 y en la propuesta de un grupo de investigadores de la Universidad Carlos III de Madrid, en el marco del proyecto CICYT(TIC960753) para una nueva versión de Métrica, se mantiene la distinción entre modelo de datos y esquema.

Esta ambigüedad se puede apreciar en expresiones como: “el modelo Entidad/Interrelación (E/R) de una universidad (para referirse al esquema de una universidad descrito en el modelo E/R) o “el modelo E/R” (para hacer referencia al modelo de datos E/R)⁹. Si no evitamos el doble significado del término, o si, al menos, no somos conscientes de ello, esta confusión terminológica puede dar lugar a otro tipo de ambigüedades conceptuales mucho más graves.

Consideraremos que el término esquema es muy apropiado en el sentido que aquí se ha dado (y que es el habitual en el campo de las bases de datos), lo que podemos confirmar si acudimos al DRAE que define dicho término como “una representación gráfica y simbólica de una cosa atendiendo sólo a sus líneas o caracteres más significativos”. Nosotros, por tanto, distinguiremos entre estos dos conceptos: **modelo** y **esquema**, aplicándolos tal como acabamos de exponer y aparece en la figura 1.5, donde se puede observar que la descripción de un cierto mundo real (por ejemplo una universidad) por medio de un modelo de datos da como resultado un esquema.

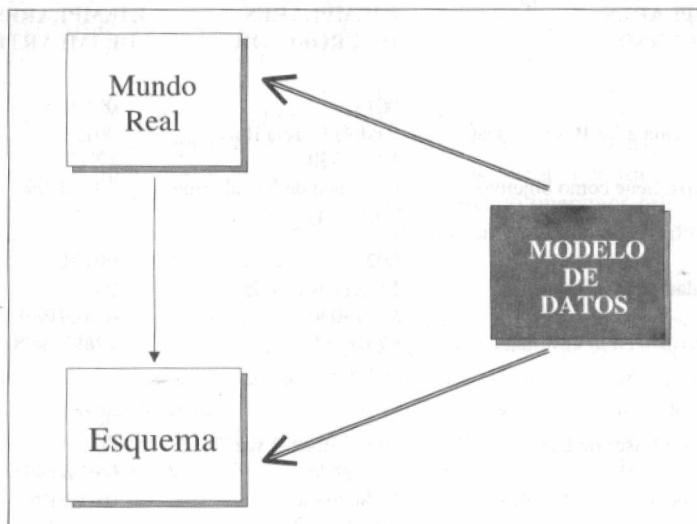


Figura 1.5. Diferencia entre modelo y esquema

⁹ Se propone al lector que, en lo sucesivo, cuando lea, en el campo de la Ingeniería del Software, algún texto relativo a los modelos de datos, lo haga siendo consciente de esta distinción, y podrá comprobar cómo, en la mayor parte de los casos, aparece la ambigüedad que aquí comentamos oscureciendo y dificultando el entendimiento del texto. En la orientación al objeto se llama “modelo de clases”, además de al modelo en sí mismo, a lo que sería, siguiendo las consideraciones aquí expuestas, un esquema de clases.

Es preciso distinguir también entre **esquema**, como descripción de la estructura de la base de datos, y **ejemplar**¹⁰ del esquema, que son los datos que en un determinado momento se encuentran almacenados en el esquema. Así, mediante un modelo de datos podríamos describir los cursos de doctorado que se imparten en nuestro mundo real, obteniendo el esquema que aparece en la figura 1.3 (a), en donde el curso se describe por aquellas características o propiedades que hemos considerado de interés (*Cód_curso*, *Nombre*, *Núm_hours*, ...), lo mismo los profesores, etc.

Los datos concretos en un determinado momento, es decir, el curso con *Nombre* Introducción a las Bases de Datos, cuyo *Cód_curso* es 00101, *Núm_hours* 20, etc. sería un *ejemplar* de CURSO. La colección de ejemplares de todos los elementos de un esquema en un momento determinado constituyen un ejemplar del esquema; en la figura 1.6 hemos representado tres ejemplares de CURSO, al igual que de PROFESOR y de IMPARTE.

EJEMPLARES DE CURSO	EJEMPLARES DE PROFESOR	EJEMPLARES DE IMPARTE
00101	001	00101
Introducción a las Bases de Datos	Andrés García Ruiz	001
030	12312330	12/12/1997
Este curso tiene como objetivo...	C/ Conde de Vistahermosa	20/12/1997
	5.621.333	
00102	002	00101
Seguridad de la información	Mercedes García Arias	003
020	50179030	01/03/1998
La seguridad en la informática...	C/ Río Miño	12/03/1998
	3.928.352	
00203	003	00203
Diseño de Bases de Datos	Julio López Pérez	002
100	52342876	05/11/1997
Dentro de las bases de datos...	C/ Segovia	07/12/1997
	6.564.125	
*****	*****	*****

Figura 1.6. Ejemplar del esquema descrito en la figura 1.3

¹⁰ Aunque no es el más usado, emplearemos el término **ejemplar** por considerar que es el que mejor expresa la idea relativa a este concepto. “*Ocurrencia*”, que es el término más extendido, pero no tiene, según el DRAE, un significado coincidente con la idea que tratamos de expresar; también se usa a veces “*instancia*”, cuyo significado no responde en absoluto a este concepto. *Realización* o *estado* son vocablos mucho más apropiados pero muy poco utilizados.

El esquema es relativamente invariante en el tiempo; mientras no cambie el mundo real (o nuestra interpretación del mismo) el esquema permanece. Sin embargo, los datos, es decir los ejemplares, son distintos en el transcurso del tiempo; por ejemplo, se da de alta un nuevo curso (inserción), desaparece un profesor (borrado), se cambia el profesor que imparte un curso (modificación), etc.

La expresión *base de datos* también se suele utilizar de forma ambigua, ya que con ella se hace referencia unas veces a un determinado ejemplar del esquema, mientras que otras se llama así a todos los valores que puede tomar un esquema en el transcurso del tiempo, es decir, a la serie de ejemplares del esquema. Si analizamos con detenimiento la definición de esquema y de base de datos de DITTRICH (1994) que aparece anteriormente, no queda claro si la base de datos es la colección de datos en un momento determinado (para nosotros, un ejemplar) o es el conjunto de posibles ejemplares, ya que la expresión de Dittrich “la colección de datos en sí misma” puede referirse a cualquiera de las dos cosas. Por ello, consideramos muy acertada la precisión de DATE (1995) que observa que, al igual que en los lenguajes de programación existen *variables* (constituidas por un *tipo* y un *contenido*), las cuales tienen, en un momento determinado, un cierto *valor*, del mismo modo en las bases de datos se debería hablar de *variables de base de datos*, cuyo tipo sería el esquema y su contenido todos los posibles valores del esquema; su valor, en un momento determinado, sería un ejemplar del esquema. Nosotros utilizaremos la expresión “base de datos” en el sentido abstracto de todos los posibles ejemplares (que no se debe confundir con el esquema que es su descripción), y cuando queramos referirnos a su contenido en un cierto momento o bien hablaremos de un ejemplar o bien diremos “la base de datos en el momento i” (BD_i).

La relación entre los conceptos modelo, esquema y ejemplar se representa en la figura 1.7, donde un modelo determinado (entre todos los existentes), como instrumento que ayuda a interpretar la realidad, permite obtener distintos esquemas al aplicarlo a realidades distintas¹¹. Cada uno de estos esquemas será una determinada percepción de una cierta realidad, y podrá tener múltiples ejemplares en el transcurso del tiempo; en un momento determinado, habrá un único ejemplar de dicho esquema.

El esquema, en sí mismo, tiene que estar descrito en términos de datos, por lo que a estos datos se los llama a veces **metadatos**, es decir, datos acerca de los datos. Si los conceptos del modelo de datos son descritos recursivamente utilizando el mismo modelo de datos, el esquema que los describe recibe el nombre de **metaesquema**.

¹¹ Incluso, el mismo modelo aplicado a la misma realidad puede dar lugar a distintos esquemas dependiendo de las distintas interpretaciones que, para el diseñador y/o el usuario, tenga esa realidad.

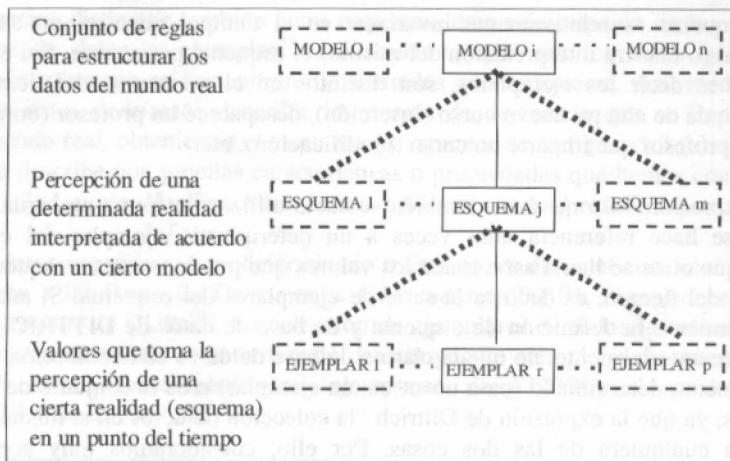


Figura 1.7. Relación entre modelo, esquema y ejemplar

3. TIPOS DE ABSTRACCIÓN EN EL DISEÑO DE BASES DE DATOS

Decíamos anteriormente que el proceso de abstracción nos ayuda a modelar los datos al hacer que nos centremos en lo esencial, pasando por alto aspectos que no consideramos relevantes para nuestros objetivos en la representación del mundo real.

En la figura 1.8 presentamos el concepto de *ambulancia* como una abstracción en la que únicamente recogemos aquellas características (chasis, ruedas, sirena, etc.), comunes a todas las ambulancias y que la distinguen de otros vehículos, que son de interés para nuestros fines.

Los modelos de datos ofrecen distintos tipos de abstracciones a fin de facilitar la representación de los datos; siendo el esquema el resultado de aplicar un proceso de abstracción a un determinado mundo real. Suelen ser cuatro¹² los tipos de abstracción que utilizan los modelos de datos en el diseño de bases de datos: **Clasificación**, **Aggregación**, **Generalización**¹³ y **Asociación**¹⁴, las cuales pueden combinarse entre sí ofreciendo interesantes mecanismos semánticos para estructurar los datos.

¹² Algunos autores –ver, por ejemplo ELSMARI (1989)– añaden la identificación; bastantes otros –ver, por ejemplo BORGIDA (1984) y BATINI (1992)– sólo consideran la clasificación, la agregación y la generalización como las abstracciones básicas.

¹³ La generalización no se utiliza en todos los modelos de datos, aunque sí se está introduciendo en extensiones, como ha ocurrido en el modelo E/R y en el estándar SQL: 1999.

¹⁴ La asociación no se utiliza en el modelo relacional, y algunos autores –como TSCHIRITZIS (1982) y BATINI (1992)– no la consideran un tipo de abstracción específico, sino una agregación.

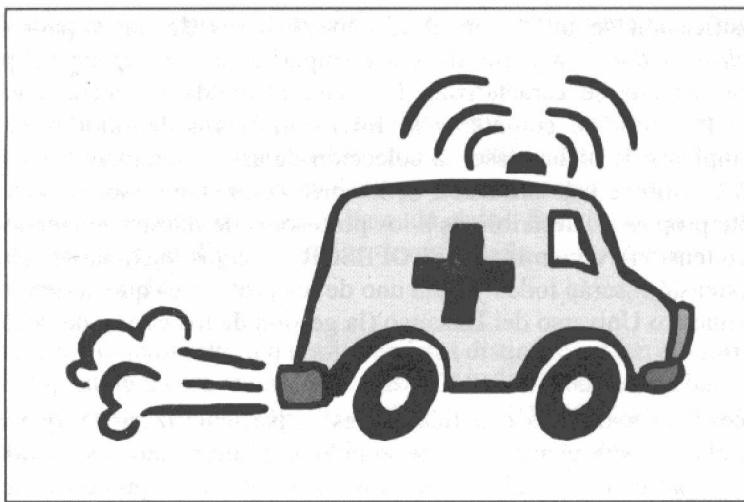


Figura 1.8. Ejemplo de abstracción

Estas abstracciones permiten establecer vinculaciones entre los elementos de un modelo. La primera (clasificación) establece una vinculación entre una categoría de objetos y cada objeto en particular (ejemplar) que pertenece a dicha categoría, mientras que en las otras tres (agregación, generalización y asociación) la relación se establece entre categorías de objetos y, por tanto, también entre los correspondientes ejemplares de dichas categorías.

3.1. Clasificación/Particularización

La clasificación es la acción de abstraer las características comunes a un conjunto de ejemplares para crear una categoría a la cual pertenecen dichos ejemplares.

BRODIE (1984) define la clasificación como “una forma de abstracción en la que una colección de objetos se considera como una clase de objetos de más alto nivel. Una clase de objetos es una caracterización precisa de todas las propiedades compartidas por todos los objetos en la colección. Un objeto es un ejemplar de una clase de objetos si tiene las propiedades definidas en la clase”. Así, podemos abstraer una clase “Mes” definiendo las características comunes a todos los meses (período de tiempo que se mide en días –aproximadamente 30 días– y cuyos límites están bien definidos) y pasando por alto diferencias que no son importantes para nuestros fines (como que unos meses comprenden algunos días más que otros); cada uno de los meses (enero, febrero, etc.) será un ejemplar de la clase “Mes”.

La clasificación se utiliza en el diseño de bases de datos para definir una categoría (*clase* o *tipo*¹⁵) a partir de sus ejemplares, las cuales tienen propiedades comunes por las que se caracterizan. La clase abstraída se puede considerar, de acuerdo con la teoría de conjuntos, la **intensión** (parte definitoria) de todos los posibles ejemplares de dicha clase; la colección de estos ejemplares en un momento determinado constituye una **extensión** de la correspondiente clase. Así, en el ejemplo anteriormente propuesto, describimos a los profesores de nuestra universidad creando una clase (intensión) denominada PROFESOR¹⁶, cuyos ejemplares en un cierto momento (extensión) serán todos y cada uno de los profesores que pertenezcan, en ese momento, a nuestro Universo del Discurso (la gestión de los cursos de doctorado de la universidad).

El proceso inverso de la clasificación es la particularización¹⁷ que consiste en pasar de la clase a sus ejemplares, generando o examinando los distintos objetos particulares a partir de la clase que los describe. Los procesos de clasificación/particularización y un ejemplo de los mismos se representan en la figura 1.9.

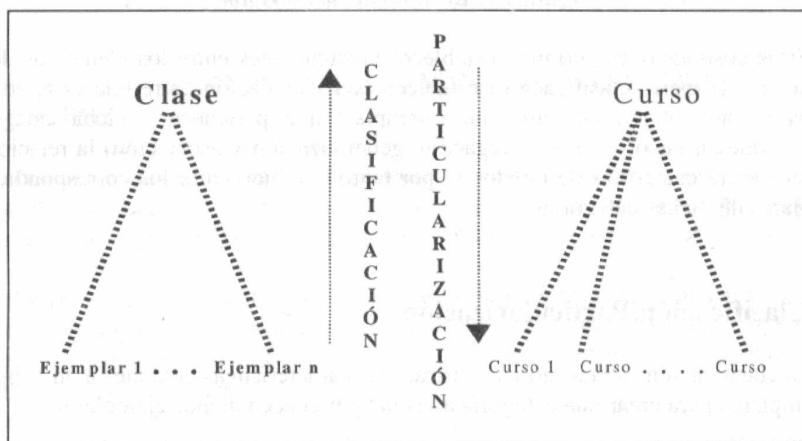


Figura 1.9. Representación de la clasificación/particularización y un ejemplo de la misma

¹⁵ Es más común en los modelos de datos utilizar el término tipo que clase; en los modelos de objetos se suele utilizar clase, aunque en alguno de ellos aparecen clase y tipo como conceptos distintos. Aquí usaremos clase o categoría en un sentido genérico, mientras que en el próximo capítulo, al tratar el modelo Entidad/Interrelación, hablaremos de tipos, que es el término que emplea este modelo.

¹⁶ Utilizaremos las mayúsculas siempre que nos estemos refiriendo a una determinada clase (o tipo) de objetos de nuestro universo del discurso como PROFESOR, CURSO, DEPARTAMENTO, PROGRAMA, etc.

¹⁷ Aunque en bases de datos se utiliza "instanciación" para expresar en castellano esta idea, este término no existe en el DRAE. Además, como hemos dicho anteriormente, tampoco el significado de instancia, según el DRAE, se corresponde en absoluto con el concepto que aquí tratamos de expresar.

La clasificación se corresponde con el concepto de **pertenencia** a un conjunto (*es miembro de*¹⁸).

Los ejemplares de una clase tienen características similares por medio de las cuales describimos la correspondiente clase; estas características toman valores concretos para cada uno de los ejemplares que pertenecen a la clase; por ejemplo, el *Nombre* de un *curso* concreto de la base de datos descrita en la figura 1.3, cuyo ejemplar aparece en la figura 1.6, es Introducción a las Bases de Datos, y el *Número_horas* es de 30¹⁹.

Los mismos objetos admiten clasificaciones distintas; así, en el caso de los cursos de doctorado, en lugar de crear una clase CURSO, podríamos haber optado por abstraer dos clases: CURSO_OBLIGATORIO y CURSO_OPCIONAL, o también CURSO_EN_ESPAÑOL y CURSO_EN_INGLÉS, por ejemplo.

Todos los modelos de datos de las bases de datos admiten la abstracción de clasificación.

3.2. Agregación/Desagregación

La abstracción de agregación consiste en construir un nuevo elemento del modelo como **compuesto** de otros elementos (**componentes**); los componentes “*son parte de*” el elemento compuesto. Así ocurre en la agregación de las clases RUEDA (4), CHASIS (1), SIRENA (1), ... para obtener la clase AMBULANCIA²⁰ –ver figura 1.10–. Esto supone que también un ejemplar de ambulancia (una ambulancia concreta) es una agregación de ejemplares de las clases compuestas (un determinado chasis, cuatro ruedas y una sirena).

Se pueden considerar tres tipos distintos de agregación:

- **Agregación de clases para obtener una clase compuesta**²¹. Por ejemplo, DEPARTAMENTO se puede modelar, mediante una abstracción de agregación, a partir de la clase ÁREA, considerándolo un conjunto de diferentes áreas, tal como aparece en la figura 1.11; en ella se puede ver también un ejemplar de dicha abstracción, donde el Departamento de Informática es una agregación del área de Lenguajes y Sistemas Informáticos

¹⁸ También se utiliza la expresión “es ejemplar de”.

¹⁹ En los modelos de objetos se permite que existan características que tomen valores para toda la clase, no para cada uno de los objetos en particular; por ejemplo, el número de cursos de doctorado que se imparten en la universidad o la media de horas por curso son características de la clase CURSO, no de sus ejemplares.

²⁰ Se puede llegar a un cierto objeto mediante diferentes procesos de abstracción. Así, anteriormente habíamos modelado la clase AMBULANCIA mediante una *clasificación* a partir de sus ejemplares; ahora abstraemos la clase AMBULANCIA como una agregación de sus componentes.

²¹ El ejemplo de la ambulancia corresponde a este tipo de agregación. La agregación de clases se puede, a su vez, subdividir en varios tipos distintos en los cuales no vamos a entrar en este texto.

(L. y S.I.) y del área de Ciencias de la Computación e Inteligencia Artificial (C.C. e I.A.).

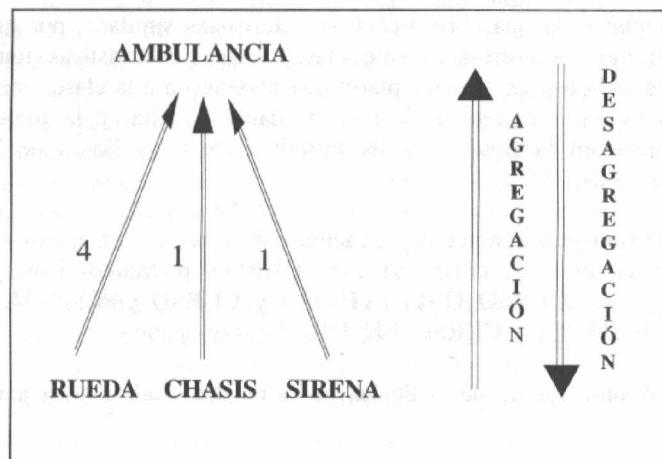


Figura 1.10. Ejemplo de agregación

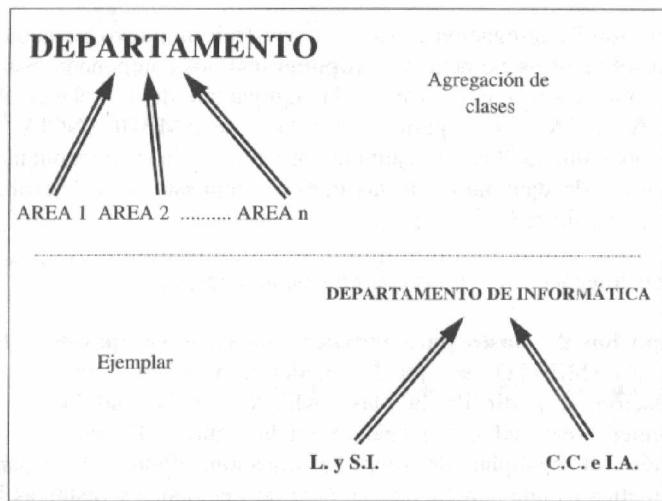


Figura 1.11. Ejemplo de agregación de clases y de un ejemplar de la misma

- **Agregación de propiedades para obtener una clase.** Por ejemplo CURSO se puede considerar como agregación de sus propiedades *Cód_course*, *Nombre*, *Núm_hours*, etc. –véase figura 1.12–. Esto supone también una agregación de valores de las correspondientes propiedades (un determinado *Nombre*, etc.) para obtener un cierto ejemplar de la clase (un curso en concreto).

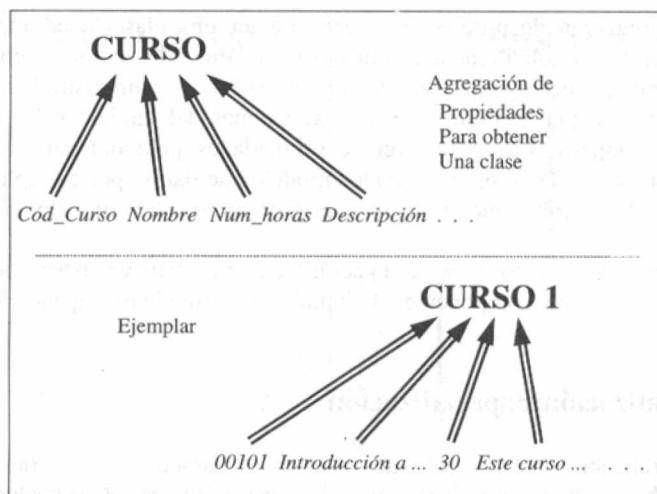


Figura 1.12. Ejemplo de agregación de propiedades y de un ejemplar de la misma

- **Agregación de propiedades para obtener una propiedad compuesta.** Por ejemplo, la agregación de Día, Mes y Año para obtener una Fecha. Esto supone una agregación de valores para obtener un valor compuesto –véase figura. 1.13–

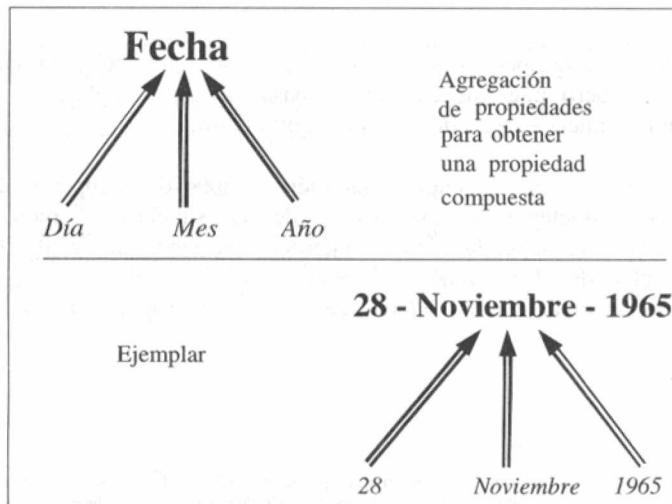


Figura 1.13. Ejemplo de agregación de propiedades para obtener una propiedad compuesta y de la correspondiente agregación en el ejemplar

La agregación de propiedades para obtener una clase la admiten, bien sea explícita o implícitamente, todos los modelos de datos; mientras que el mecanismo de agregación de clases sólo lo suministran los modelos semánticos (por ejemplo, algunas extensiones del modelo E/R) y los modelos de objetos. La agregación de propiedades para obtener una propiedad compuesta no la admiten todos los modelos de datos (por ejemplo, el modelo Codasyl la admite, mientras que no está permitida en el modelo relacional²²).

El proceso inverso a la agregación, que consiste en pasar del elemento compuesto a sus componentes, lo hemos denominado **desagregación**²³.

3.3. Generalización/Especialización

La generalización es la acción de abstraer las características comunes a varias clases (**subclases**) para constituir una clase más general (**superclase**) que las comprenda a todas; cada generalización es un árbol (jerarquía) de un solo nivel, donde la raíz es la superclase y las hojas son las subclases; el proceso inverso de la generalización es la especialización, en la que se pasa de la superclase a las subclases. Por ejemplo, se puede generalizar las clases PROFESOR y ESTUDIANTE en la superclase PERSONA, la cual tendrá las características comunes a ambas subclases, como el *Código*, *Nombre*, *Apellidos*, etc.; también podríamos, a partir de PERSONA, pasar a las clases PROFESOR y ESTUDIANTE mediante una especialización –véase figura 1.14–.

La generalización/especialización es un proceso parecido a la clasificación/particularización, pero mientras en ésta se pasa de los ejemplares a la clase (o viceversa), en la primera se pasa de una clase a otra clase.

Todo ejemplar de una subclase es también ejemplar de la superclase y, además de poseer las características específicas de la subclase, hereda todas las correspondientes a la superclase²⁴; por ejemplo, un ejemplar de PROFESOR es también ejemplar de PERSONA y hereda el *Código*, el *Nombre*, etc. de la correspondiente persona, teniendo además características propias como *Materia*, *Tipo*, etc.

²² El modelo relacional, tal como fue inicialmente propuesto –CODD (1970)– no admite los agregados de datos –atributos compuestos–; en cambio, la versión 2 del modelo, CODD (1990), sí los admite, lo que es criticado por DATE (1992).

²³ Aunque en los modelos de objetos se suele utilizar el término *desensamblaje*, hemos considerado preferible, al menos cuando estamos presentando el concepto de modelo de datos en general y no un modelo concreto, denominar desagregación al proceso inverso a la agregación.

²⁴ En algunos modelos de objetos se permite la *inhibición de herencia* (mediante la cual se puede no heredar **todas** las características de la superclase).

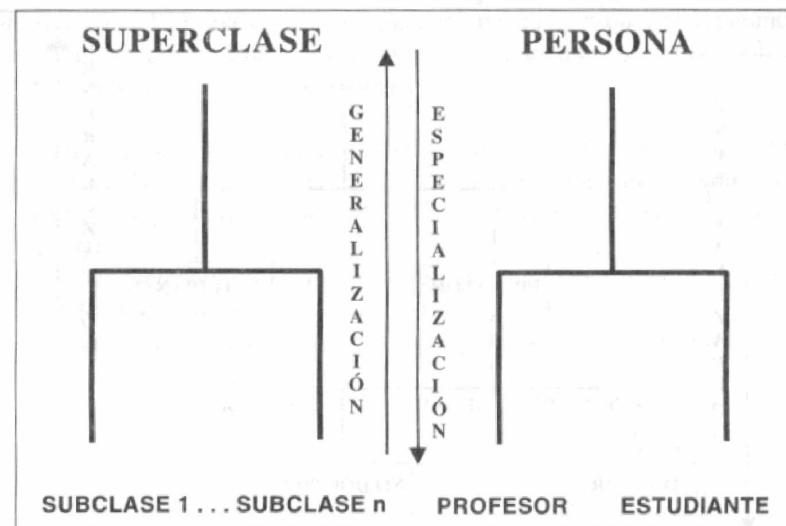


Figura 1.14. Representación y ejemplo de generalización/especialización

El conjunto de ejemplares de una subclase “es un” subconjunto de los ejemplares de la correspondiente superclase, por lo que se crea la expresión “ES_UN” (en inglés “IS_A”) para denominar estas jerarquías que tuvieron su origen en el campo de la inteligencia artificial.

Se admite la aplicación de sucesivas generalizaciones/especializaciones formando jerarquías de varios niveles tal como aparece en la figura 1.15, donde se combina la generalización de ESTUDIANTE y PROFESOR en PERSONA y la especialización de PROFESOR en DOCTOR y NO_DOCTOR.

Aunque esta abstracción es muy intuitiva y muy útil en el proceso de diseño por la semántica que permite captar, se ignora en bastantes modelos de datos; aquellos que la admiten suministran distintos mecanismos de generalización a los que se pueden aplicar distintos tipos de restricciones²⁵.

²⁵ En el siguiente capítulo, al estudiar el modelo E/R, volveremos de nuevo a esta abstracción, analizando con más profundidad sus características.

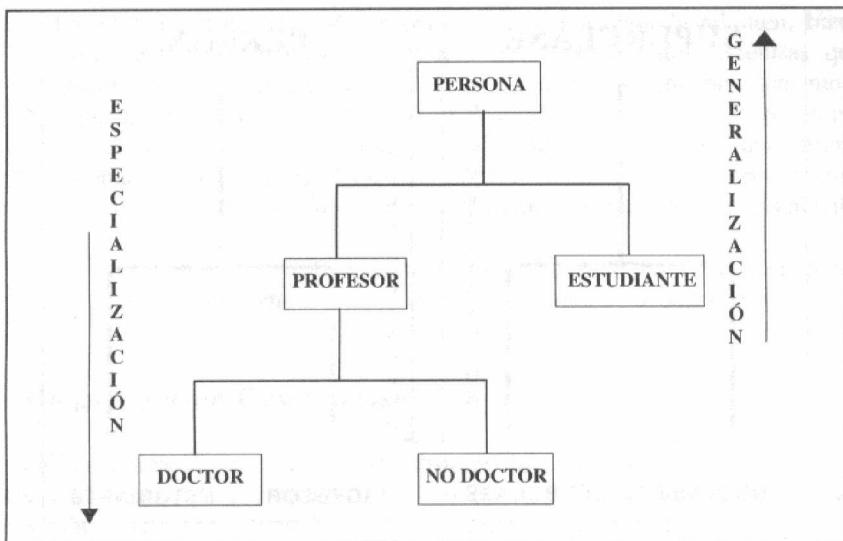


Figura 1.15. Ejemplo de jerarquía a dos niveles con una generalización y una especialización

3.4. Asociación/Disociación

Es una abstracción que se utiliza para vincular dos o más clases (y, por tanto sus ejemplares), creándose un elemento de un tipo distinto. Por ejemplo, en la figura 1.16 aparece la asociación **Imparte** entre las clases PROFESOR y CURSO (un determinado profesor “imparte” un cierto curso y un determinado curso “se imparte” por un cierto profesor); “imparte” es un elemento de un tipo distinto de PROFESOR y CURSO, y tampoco, en nuestra opinión, es una agregación de las mismas como consideran algunos autores.

Existen notables diferencias entre la asociación y las abstracciones anteriormente estudiadas; diferencias que se escapan al alcance de este libro, pero que llevan a que, como ya hemos señalado, no siempre se incluya la asociación entre las abstracciones.

Ciertos autores como BORGIDA (1984), no consideran la asociación entre los tipos de abstracción, mientras que para otros, como TSICHRITZIS (1982), la asociación es una agregación. En algunos modelos de datos tampoco aparece esta abstracción como tal, no existiendo ningún concepto especial para representarla²⁶. Para nosotros, aun cuando consideramos que la asociación puede parecerse a la agregación,

²⁶ Por ejemplo, en el modelo relacional, tanto las clases como las asociaciones entre clases se representan por medio de **relaciones**, no haciendo distinción alguna entre unas y otras.

tiene ciertos rasgos distintivos que aconsejan, en nuestra opinión, tratarla como un tipo diferente de abstracción²⁷. Entre otras, existen las siguientes diferencias entre ambos tipos de abstracción:

- Cuando se asocian dos o más categorías, el nuevo elemento que aparece tiene determinadas características que lo distinguen de las categorías normales, por lo que, en general, los modelos de datos crean un nuevo concepto para representarlo.
- El nuevo elemento no *está compuesto*, como en el caso de la agregación, por los elementos que asocia.
- En la agregación puede existir herencia, y no así en la asociación.

Al proceso inverso a la asociación lo hemos llamado *disociación*.

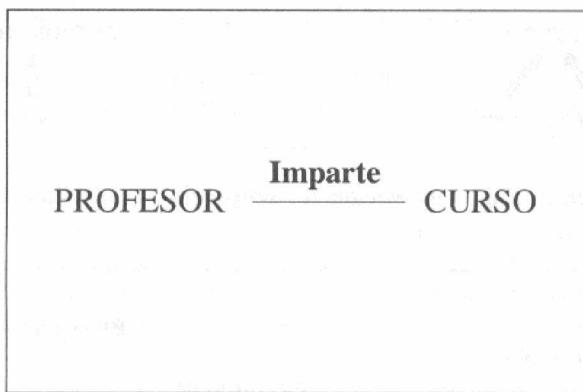


Figura 1.16. Ejemplo de asociación

3.5. Jerarquías de abstracciones

En el proceso de modelado de una determinada realidad, es preciso a menudo combinar distintas abstracciones, formando lo que se conoce como una jerarquía de abstracciones. En la figura 1.17 se combina la agregación de propiedades con la generalización, apareciendo la clase PERSONA como una **agregación** de sus características (*DNI, Nombre, Dirección*) y también como una generalización de PROFESOR y ESTUDIANTE, éstos además de las características heredadas (*DNI, Nombre, Dirección*), pueden tener características propias (PROFESOR tiene *Materia* y *Tipo*; ESTUDIANTE tiene *Curso*). Vemos, por consiguiente, que una misma clase

²⁷ Un análisis más en profundidad de este tipo de abstracción se hará en el próximo capítulo, al estudiar el modelo Entidad/Interrelación, ya que la asociación es un concepto fundamental en este modelo, en el que la nueva clase creada (así como los ejemplares de la misma) constituyen elementos distintos del modelo que reciben el nombre de **interrelación**.

(PERSONA) se puede abstraer tanto por agregación de propiedades como por generalización de otras clases.

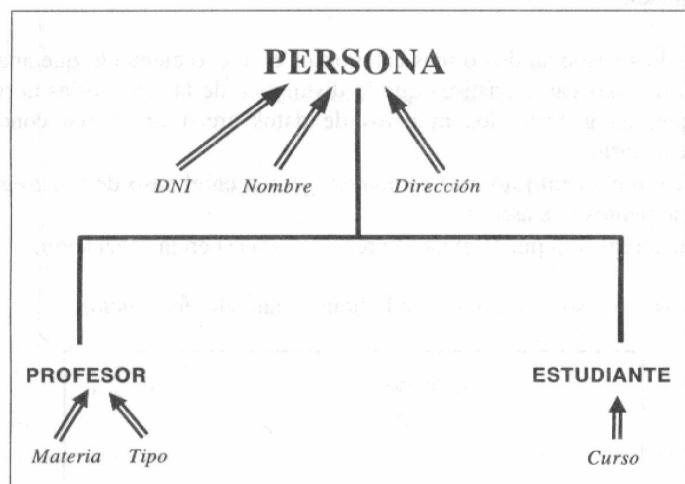


Figura 1.17. Combinación de agregación y generalización

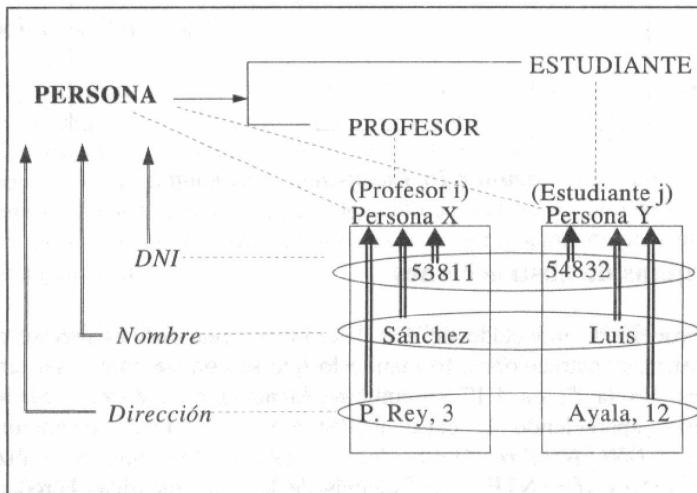


Figura 1.18. Ejemplo de abstracciones de clasificación, agregación y generalización

También es posible abstraer una categoría por clasificación de sus ejemplares. En la figura 1.18 la categoría PERSONA se obtiene, además de por generalización o por agregación como en el ejemplo anterior, por clasificación de sus ejemplares (persona

X, Y, ...); cada una de las subclases PROFESOR y ESTUDIANTE se pueden obtener también por clasificación de sus respectivos ejemplares.

4. CONCEPTO DE MODELO DE DATOS

Aunque existen muchos modelos de datos es posible abstraer una serie de características comunes a todos ellos, definiendo así el concepto de modelo de datos en general, que posteriormente se ha de particularizar para describir cada modelo en concreto.

Podemos ya definir de forma más precisa el concepto de modelo de datos como “*un conjunto de conceptos, reglas y convenciones bien definidos²⁸ que nos permiten aplicar una serie de abstracciones a fin de describir y manipular²⁹ los datos de un cierto mundo real que deseamos almacenar en la base de datos*”.

Los modelos de datos facilitan la creación de categorías mediante la aplicación de los tipos de abstracción anteriormente considerados, lo que lleva a diferenciar dos tipos de modelos:

- *Modelos de datos estrictamente tipados*, en los que cada dato tiene que pertenecer forzosamente a una categoría previamente definida en el esquema; los datos (ejemplares) que no pertenecen a una categoría no pueden ser manejados por el modelo. En algunos casos, no se permite la evolución de las categorías, y los datos tienen que permanecer en la categoría en la que fueron creados.
- *Modelos de datos débilmente tipados*, en los que no es obligatorio que los datos (ejemplares) pertenezcan a categorías, sino que pueden existir por sí mismos. Se permite la existencia de categorías en aquellos casos en que es conveniente para representar ciertas extensiones. Las categorías, si existen, se tratan como los datos individuales.

En las bases de datos se usan modelos estrictamente tipados, dado que, a pesar de sus inconvenientes, en especial su falta de flexibilidad, tienen como ventaja el posibilitar el tratamiento de grandes cantidades de datos al agruparlos en categorías. Éstos son los modelos de datos a los que nos referiremos en este libro.

²⁸ A veces se añade “matemáticamente”, es decir se considera que los conceptos de un modelo de datos han de estar “bien definidos matemáticamente”, en la realidad muchos modelos de datos han surgido en la práctica soportados por los correspondientes Sistemas de Gestión de Bases de Datos (SGBD), no habiendo sido por tanto formalmente definidos; por esta razón, aunque otros modelos de datos, como el relacional, si han sido definidos en términos matemáticos, nosotros hemos preferido no introducir esta restricción que no cumplen todos los modelos.

²⁹ En algunos casos no se incluye la manipulación como parte del modelo, sino que sólo se consideran sus aspectos estáticos, especialmente en aquellos orientados al diseño de alto nivel (modelos conceptuales).

Un modelo de datos define las reglas según las cuales han de ser estructurados los datos acerca del mundo real. Como ya hemos dicho, la representación de una determinada realidad mediante un modelo (instrumento que nos facilita el proceso de representación) da lugar a un esquema, el cual describe las categorías existentes en dicha realidad. Sin embargo, la realidad no contempla sólo aspectos estáticos, como son aquellos que se representan en el esquema, sino también propiedades dinámicas, ya que los ejemplares de las categorías varían en el transcurso del tiempo, y estas propiedades dinámicas han de ser también especificadas en operaciones de consulta y actualización de la base de datos.

Por tanto, podemos decir que las propiedades del mundo real son de dos tipos:

- *Estáticas*, o relativamente invariantes en el tiempo, que responden a lo que se suele entender como estructuras.
- *Dinámicas*, que son las operaciones que se aplican a los datos o valores almacenados en las estructuras, los cuales varían en el transcurso del tiempo al aplicárseles dichas operaciones.

El modelo de datos ha de proporcionar facilidades para recoger ambos aspectos, por lo que se define formalmente como el par:

$$\text{MD} = \langle \mathbf{G}, \mathbf{O} \rangle$$

donde **G** es el conjunto de reglas de generación que permiten representar la componente estática, es decir, describir las estructuras de nuestro universo del discurso, y **O** es el conjunto de operaciones autorizadas sobre la correspondiente estructura, operaciones que permiten representar la componente dinámica.

La componente estática de un determinado modelo de datos expresado con una sintaxis es el *Lenguaje de Definición de Datos (LDD)*, y la componente dinámica el *Lenguaje de Manipulación de Datos (LMD)*; ambos constituyen el *Lenguaje de Datos (LD)*³⁰.

Un modelo de datos define reglas generales para especificar las estructuras de datos y las operaciones permitidas sobre los datos³¹; estas operaciones han de ser ejecutadas en el contexto proporcionado por las estructuras.

Analicemos a continuación, con mayor detalle, cada uno de estos componentes.

³⁰ Los SGBD suelen tener además un Lenguaje de Consulta (en inglés “Query Language” –QL–) y un *Lenguaje de Control* (en inglés Control Language).

³¹ En los modelos conceptuales la componente dinámica o bien no se define o se le concede poca importancia; en cambio, en los modelos de implementación (modelos convencionales), la componente dinámica es tan importante como la estática.

4.1. Estática

La estática de un modelo de datos está compuesta por:

- A) **Elementos permitidos.** No son los mismos para todos los modelos de datos (varían especialmente en terminología), pero en general son:

- A1) **Objetos**³² (entidades, relaciones, registros, etc.)
- A2) **Asociaciones** entre objetos (interrelaciones, “set”, etc.)
- A3) **Propiedades** o características de los objetos o asociaciones (atributos, campos, elementos de datos, etc.)
- A4) **Dominios**, que son conjuntos nominados de valores homogéneos sobre los que se definen las propiedades.

A estos elementos permitidos se les podrán aplicar aquellas abstracciones reconocidas por el modelo. La representación de estos elementos depende de cada modelo de datos, pudiendo hacerse en forma de **grafos** (como en el modelo E/R, o en el Codasyl) o de **tablas** (como en el modelo relacional).

- B) **Elementos no permitidos o restricciones.** No todos los valores, cambio de valor o estructuras están permitidos en el mundo real; por ejemplo, un niño de tres años no puede estar casado, ni una persona puede pasar directamente de soltera a viuda, etc. Además, cada modelo de datos también impone por sí mismo limitaciones a las estructuras que admite³³. Estas limitaciones, que unas veces vienen impuestas por el mismo modelo de datos y otras nos las impone el universo del discurso que estamos modelando, se denominan *restricciones*; las que son impuestas por el modelo son *restricciones inherentes* y las que responden al deseo de que el sistema de información sea un reflejo lo más fiel posible del mundo real son las *restricciones de integridad o semánticas*. Existen, por tanto, dos tipos de restricciones:

- B1) **Restricciones inherentes** (del modelo), son aquellas que vienen impuestas por la misma naturaleza del modelo de datos, el cual no admite ciertas estructuras, introduciendo así rigideces a la hora de modelar. El usuario (diseñador) no define estas restricciones, siendo el SGBD, en el que subyace el modelo, el que impide, en el momento de la definición del esquema, que se introduzcan estructuras no admitidas por el correspondiente modelo.

³² En este capítulo, que trata de los modelos de datos en general, hemos preferido referirnos a objetos en lugar de entidades, término que se suela asociar a un determinado modelo (el Modelo Entidad/Interrelación); tampoco hemos querido emplear el vocablo “cosas”, como hacen algunos autores, por ejemplo KENT (1978). No entramos aquí en las dificultades que entraña dar una definición rigurosa de lo que es un objeto dentro de este contexto, ni en la diferenciación o similitud de objeto con asociación o propiedad; por ahora, para nosotros, un objeto tiene, como ya hemos dicho, el significado, que conoce cualquier lector, del lenguaje común.

³³ El modelo relacional, por ejemplo, no permite que dos ejemplares (filas) de una tabla sean iguales.

B2) **Restricciones de integridad o semánticas³⁴** (de usuario), son aquellas que permiten captar la semántica del universo del discurso que se quiere modelar y verificar la corrección de los datos almacenados en la base. El usuario (diseñador) ha de definir, y a veces programar, estas restricciones a fin de rechazar ciertas asociaciones o de limitar los valores que pueden tomar los datos de la base de datos o de impedir ciertos cambios de los mismos. Según los instrumentos que proporcione el modelo de datos para definir y gestionar las restricciones, éstas pueden ser:

- **Reconocidas por el MD.** Su definición le corresponde al diseñador, pero su gestión es responsabilidad del modelo de datos, el cual las reconoce y recoge en el esquema, suministrando instrumentos para su definición y obligando a su cumplimiento.
- **Ajenas al MD.** Son, por completo, responsabilidad del diseñador, ya que el modelo de datos no las reconoce ni proporciona instrumentos para manejarlas.

Podemos, definir la componente estática del modelo de datos como el par:

$$G = \langle G_e, G_r \rangle$$

donde G_e es el conjunto de reglas de generación de estructuras (objetos del modelo y restricciones inherentes) y G_r es el conjunto de restricciones de usuario.

La aplicación de la componente estática (reglas de generación) de un modelo de datos a un determinado Universo del Discurso (UD) nos da como resultado un esquema, que es la estructura de datos que describe, en el correspondiente modelo, las categorías que han resultado de las abstracciones aplicadas al mundo real que se trata de modelar; es decir:

$$G[UD] = E$$

4.2. Dinámica

El conjunto de valores que toman las distintas categorías de un esquema en un momento determinado t_i recibe el nombre de ejemplar del esquema o estado de la base de datos³⁵ en el tiempo t_i (BD_i); en otro momento t_j el ejemplar del esquema será BD_j . Si entre t_i y t_j se ha producido un cambio en algún valor de la base de datos (alta, baja

³⁴ En el siguiente epígrafe hacemos un estudio más completo de estas restricciones.

³⁵ Ya hemos advertido que también se denomina instancia y, a veces, se habla simplemente de base de datos en el momento t_i .

o modificación) $BD_i \neq BD_j$. Tanto BD_i como BD_j deben ser ejemplares válidos de la base de datos, es decir, los valores de ambos deben pertenecer a alguna de las categorías definidas en el esquema³⁶ y cumplir las restricciones de integridad (también deben cumplir, en caso de que existan, las posibles restricciones asociadas al cambio de estado).

La componente dinámica del modelo consta de un conjunto de operadores que se definen sobre la estructura del correspondiente modelo de datos, ya que no todas las estructuras admiten el mismo tipo de operaciones. La aplicación de un operador a un exemplar de un esquema transforma éste en otro exemplar:

$$O [BD_i] = BD_j$$

Pudiendo ser $BD_i = BD_j$, por ejemplo en caso de consulta o cuando falla una operación por haberse producido un error³⁷.

Una operación tiene dos componentes:

1. **Localización** o “enfoque”, consiste en localizar un exemplar de un objeto indicando un camino, o un conjunto de ejemplares especificando una condición. En el primer caso se trata de un sistema **navegacional**, mientras que el segundo se dice que es de **especificación**.
2. **Acción**, que se realiza sobre el(es) exemplar(es) previamente localizado(s) mediante una operación de localización, y puede consistir en una recuperación o en una actualización (inserción, borrado o modificación).

Sin seguir una sintaxis concreta, sino más bien en un plano conceptual, podemos expresar una sentencia del LMD de la siguiente forma:

LOCALIZACIÓN <condición>

ACCIÓN <objetivo>

donde **LOCALIZACIÓN** y **ACCION**³⁸ son mandatos del LMD, **<condición>** representa una expresión lógica que deben cumplir los objetos que se desea localizar o

³⁶ Ya hemos dicho que nos estamos refiriendo a modelos de datos estrictamente tipados.

³⁷ Si consideramos que el estado de la base de datos viene determinado no sólo por los valores que toman los objetos del esquema, sino también por los valores de sus indicadores (por ejemplo el indicador de error), cualquier operación hace variar el estado de la BD, bien porque cambian los valores de los objetos (en caso de una actualización), bien porque cambian los indicadores (en caso de fallo o de consulta). En algunos MD, como el Codasyl, la manipulación de los datos está basada en los indicadores.

³⁸ La distinción entre localización y acción es de tipo formal; y si bien algunos lenguajes, como el LMD de Codasyl, tienen dos mandatos distintos para expresar la selección y la acción, distinguiendo claramente entre ambos tipos de operación, otros lenguajes, como el SQL, reúnen ambas operaciones en un único operador.

señala el camino que permite llegar a esos objetos, mientras que <objetivo> indica los objetos (o las propiedades de éstos) sobre los que se aplica la acción³⁹.

5. RESTRICCIONES DE INTEGRIDAD

En el mundo real existen ciertas *reglas* que deben cumplir los elementos en él existentes; por ejemplo, una persona sólo puede tener un número de DNI y una única dirección oficial (la que figura en el padrón); además, un número de DNI sólo puede corresponder a una única persona, etc. Cuando diseñamos una base de datos deseamos que ésta refleje lo más fielmente posible el universo del discurso que estamos tratando de recoger en nuestro sistema de información, por lo que en el esquema de la base de datos, junto con los objetos, las asociaciones y las propiedades de los mismos, debemos describir también estas reglas, llamadas restricciones semánticas o de integridad⁴⁰, las cuales pueden ser definidas como condiciones que limitan el conjunto de ejemplares válidos de un esquema.

La *semántica* y la *integridad* son conceptos que en las bases de datos suelen ir asociados, aunque no son idénticos. Con el término semántica nos referimos al significado de los datos y con el de integridad a la corrección de los mismos y a su consistencia respecto al mundo real del cual proceden. Cuando en el esquema de una base de datos se encuentra descrita la semántica del mundo real, será posible comprobar si los valores de los datos se atienen o no a esta semántica previamente definida, comprobándose la integridad de los mismos; de ahí que digamos que ambos conceptos suelen ir unidos. Nosotros utilizaremos indistintamente las expresiones *restricciones semánticas* o *restricciones de integridad* o, a veces, diremos simplemente *restricciones* cuando por el contexto se comprenda que no nos estamos refiriendo a las restricciones inherentes.

La semántica de los datos, es decir todo lo que conocemos acerca de los datos, se encontraba en un principio en la mente del usuario, el cual comprobaba manualmente si los datos cumplían o no las reglas a ellos asociadas; después fue migrando desde la mente del usuario a los programas; y, por último, ha pasado desde éstos a la base de datos, tal como se muestra en la figura 1.19.

³⁹ Si el SGBD se adaptase estrictamente a la arquitectura a tres niveles de ANSI, el <objetivo> sería el nombre de un esquema externo previamente definido; sin embargo, algunos SGBD, especialmente los basados en el modelo relacional, no obligan a definir previamente el esquema externo, permitiendo describir el objetivo dentro de la misma sentencia de manipulación.

⁴⁰ Algunos autores –véase, por ejemplo, DATE (1995)– prefieren llamarlas reglas, reservando el término “restricción” para la condición que se debe satisfacer; sin embargo, la denominación más habitual es la de restricción.

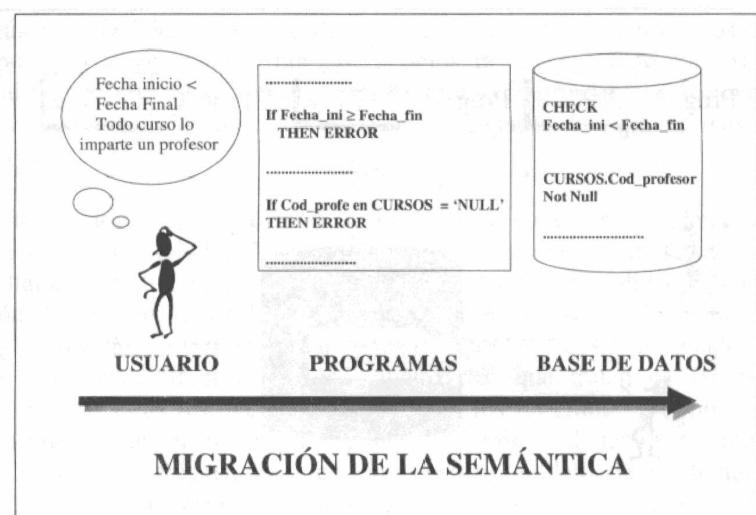


Figura 1.19. Migración de la semántica de los datos

Son muchas las ventajas de tener integrada la descripción de las restricciones junto con la de los datos en el esquema de la base de datos, en lugar de que esté dispersa entre los diferentes programas de aplicación. Por un lado, tiene ventajas relativas a la integridad, ya que al ser única la descripción de las restricciones no se pueden producir inconsistencias debidas a que los distintos programadores hayan definido, cada uno en su programa y no de manera uniforme, las restricciones de integridad (por ejemplo, un programador se puede olvidar de incluir en su aplicación una determinada comprobación que otros sí han incluido); de esta forma puede, además, disminuirse drásticamente la carga de programación (se considera que la programación de las sentencias necesarias para controlar la integridad puede llegar a suponer en algunos casos hasta un 90% del total de una aplicación).

Por otro lado, tiene también ventajas semánticas, ya que es importante que el significado de los datos, como parte fundamental de los mismos, se encuentre descrito junto con el resto de sus características y que sea únicamente el diseñador el que se ocupe, por una sola vez, de definir la semántica, no dejando esta responsabilidad en manos de los programadores de aplicaciones; lo cual evita redundancias y permite a los usuarios, siempre que tengan la debida autorización, conocer el significado de los datos sólo con consultar el esquema de la base de datos en lugar de tener que indagar en las diferentes aplicaciones, tal como se muestra en la figura 1.20.

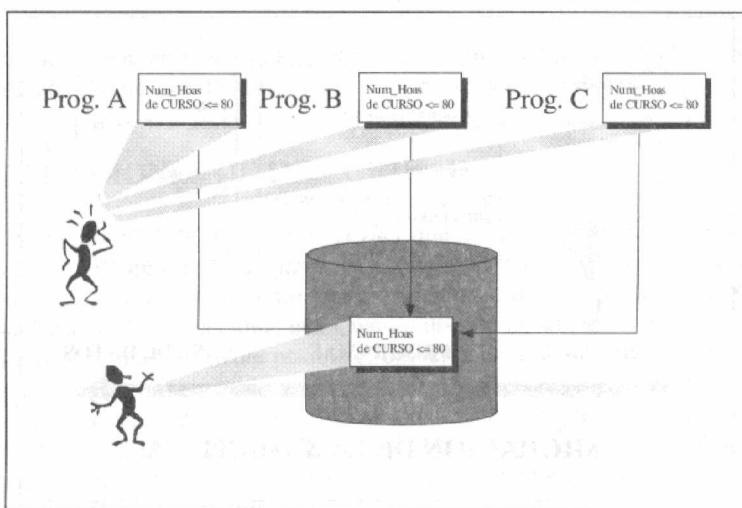


Figura 1.20. Semántica de los datos “dispersa” entre los diferentes programas de aplicación, frente a semántica integrada en la base de datos

Todas estas razones nos muestran la necesidad de que la semántica del mundo real se encuentre descrita en el esquema, es decir, esté centralizada en lugar de dispersa en los diferentes programas de aplicación que actualizan la base de datos; pero, para conseguir este objetivo, los modelos de datos han de permitir representar las restricciones semánticas dando instrumentos para ello, y los SGBD en los que están soportados los modelos tienen que reconocer y gestionar estas restricciones. Las restricciones semánticas que pueden ser especificadas en un determinado modelo de datos y representadas en sus esquemas decimos que son restricciones semánticas *propias*⁴¹ del modelo.

Sin embargo, ningún modelo de datos es capaz de capturar toda la semántica del mundo real mediante restricciones propias, por lo que, en general, es necesario tener restricciones adicionales que no estarán soportadas por el modelo de datos, a las que llamamos restricciones semánticas *ajena*s al correspondiente modelo. También debemos precisar que ciertas restricciones propias de un modelo no son a veces soportadas por algunos SGBD basados en ese modelo⁴².

Tampoco debemos confundir, en especial en el caso del relacional, el modelo con el estándar de un cierto lenguaje para ese modelo, como es el SQL; por lo general, serán distintas las restricciones propias de un estándar y las de los diferentes productos comerciales basados en él.

⁴¹ Que no hay que confundir con las restricciones “inherentes”.

⁴² Por ejemplo, la semántica de los dominios no está totalmente soportada en estos momentos en casi ningún SGBD comercial, a pesar de que los dominios siempre han sido elementos propios del modelo relacional.

Según CODD (1979), la tarea de capturar la semántica de los datos no termina nunca; por esta razón, surgen nuevos modelos de datos (como los orientados al objeto), al tiempo que los modelos existentes (como ocurre con el relacional y con el E/R) se van extendiendo con el objetivo de ser capaces de capturar más elementos semánticos.

Las restricciones ajenas al modelo no son otra cosa que programas o procedimientos escritos en cualquier lenguaje de propósito general (lenguaje anfitrión) con posibles llamadas al lenguaje de manipulación de datos. Su finalidad es comprobar la corrección de una operación de actualización, impidiendo que la violación de una cierta regla existente en el mundo real dé lugar a que los datos almacenados en la base de datos sean inconsistentes con ese mundo real que tratan de representar; sin embargo, al estar estas restricciones dispersas en los programas de aplicación tienen los inconvenientes que acabamos de exponer. La finalidad de las restricciones propias de un modelo es la misma, pero en este caso el modelo da facilidades para su definición y considera esta definición como parte integrante de su esquema. Un lenguaje estándar que asuma completamente un modelo debe proporcionar facilidades (es decir, una sintaxis) para definir todos los elementos del modelo (por tanto, todas las restricciones reconocidas por éste). Del mismo modo, un producto basado en un modelo o en un estándar debe ser capaz de reconocer y gestionar todas las restricciones propias del modelo o del estándar. En general, cuando en el campo de las bases de datos se habla de restricciones de integridad se está haciendo referencia a las restricciones de integridad propias del modelo; además, es habitual suponer que el correspondiente SGBD asume totalmente el modelo, aunque ello no sea cierto en muchos casos.

5.1. Componentes de una restricción

En general, en una restricción de integridad es posible distinguir los siguientes componentes:

- La *operación* de actualización (inserción, borrado o modificación) cuya ejecución ha de dar lugar a la comprobación del cumplimiento de la restricción.
- La *condición*⁴³ que debe cumplirse, la cual es en general una proposición lógica, definida sobre uno o varios elementos del esquema, que puede tomar uno de los valores de verdad (cierto o falso⁴⁴).

⁴³ Es lo que hemos dicho que, a veces, se llama propiamente restricción.

⁴⁴ Podría también tomar el valor de verdad “quizás” en el caso de admitirse valores nulos, teniendo entonces que evaluarse la condición mediante una lógica trivaluada. Obviamos aquí este tema de los valores nulos o información “faltante”, porque introduce una importante complejidad adicional sin aportar nada al concepto de las restricciones de integridad en sí mismo.

- La *acción* que debe llevarse a cabo dependiendo del resultado de evaluar la condición⁴⁵.

Las restricciones de integridad se pueden considerar, en cierto modo como reglas ECA (Evento, Condición, Acción), en las cuales, al ocurrir un evento (en este caso una actualización), se comprueba una condición y dependiendo de su resultado se pone en marcha una acción (rechazar la operación, informar al usuario, corregir el error, etc.)⁴⁶.

Además de estos elementos, las reglas de integridad pueden tener un nombre por medio del cual es posible identificarlas, y también puede indicarse el *momento* en el que ha de evaluarse la condición (antes o después de la operación, de forma inmediata o diferida al final de una transacción, etc.).

En bastantes casos es posible prescindir de alguno de estos componentes, de modo que, por defecto, se tome una cierta opción, simplificando así la definición de las restricciones.

Las restricciones han de ser definidas en la fase de diseño y el cumplimiento de la condición tiene que ser verificado en la de ejecución cuando se está procesando la operación de actualización que provoca cambios en el estado de la base de datos. En una restricción es por tanto necesario distinguir lo que ocurre en la fase de definición y en la de ejecución (actualización de la base de datos).

- *Fase de definición.* En ella, el diseñador ha de describir la restricción especificando sus componentes. El sistema debe comprobar que la definición de la restricción es correcta (respecto al modelo de datos soportado por el sistema) y que el conjunto de restricciones es consistente en sí mismo; por ejemplo, el diseñador puede, por equivocación, definir una restricción sobre atributos inexistentes o imponer una restricción prohibiendo valores nulos para un cierto atributo y en otra restricción declarar una acción de puesta a nulos para ese mismo atributo; en estos casos, el sistema debería detectar la falta de corrección o de consistencia de las restricciones. Una vez comprobada la validez de una restricción, ésta debe ser compilada, junto con los otros elementos, por el SGBD e incluida en el esquema.
- *Fase de ejecución.* En el momento de la ejecución de una sentencia de actualización sobre la que se ha definido una restricción en la que están implicados elementos que van a ser actualizados, es preciso que el sistema

⁴⁵ En general, la acción se desencadena en caso de fallo de la condición, es decir, la respuesta se produce ante un intento de violación de la restricción, aunque —como veremos posteriormente— en ciertos tipos de restricciones como son los *disparadores* la acción se desencadena cuando se cumple la condición.

⁴⁶ Son los mismos componentes de las reglas en las bases de datos activas, pero es que la integridad no es otra cosa que un caso especial de actividad, en la que la operación desencadenante es siempre una actualización, mientras que, en las bases de datos activas, la operación desencadenante puede ser de otro tipo.

compruebe la condición a fin de que si se estuviese produciendo un intento de violación poner en marcha la acción especificada en el momento indicado.

DATE (1990) hace un estudio de la integridad de las bases de datos, proponiendo un enfoque hacia un lenguaje general para formular reglas de integridad en forma declarativa.

5.2. Clasificación de las restricciones

Es muy difícil hacer una clasificación rigurosa de las restricciones, ya que éstas varían mucho dependiendo de los modelos y de los productos; tampoco en los trabajos consultados hemos encontrado una clasificación completa y consistente de las mismas; en la figura 1.21 proponemos una jerarquía de clasificación de las restricciones⁴⁷.

Dentro de las restricciones semánticas, las que hemos llamado *ajenas al modelo de datos*, son procedimientos específicos incluidos en los programas de aplicación a fin de recoger la semántica del UD, que permiten comprobar la consistencia de los datos de la base (obsérvese que su creación está ligada a la función de manipulación y no a la de definición). No están almacenados en el esquema de la BD y, por tanto, pueden ser violadas en actualizaciones en las que no se haya programado la correspondiente restricción. Por esta razón el SGBD tampoco tiene conocimiento de su existencia y el optimizador no puede tomarlas en consideración⁴⁸. Suponen una importante carga de programación y de mantenimiento –ya que en general son procedimentales–, a cambio de lo cual proporcionan el máximo de flexibilidad. Como su nombre indica, son totalmente ajenas al modelo de datos; pero los productos pueden dar facilidades para definirlas. Dentro de ellas se puede distinguir las que se embeben en programas de propósito general y las que constituyen facilidades proporcionadas por algún módulo o lenguaje del SGBD⁴⁹ (no por el núcleo, ya que en este caso no se trataría de restricciones ajenas al modelo).

⁴⁷ La caracterización de las restricciones inherentes ya se ha expuesto cuando se ha definido la estática de un modelo de datos, por lo que en este epígrafe nos limitaremos a estudiar las restricciones semánticas; aunque en la figura 1.21 y en el cuadro que aparece al final del epígrafe también se han incluido las restricciones inherentes.

⁴⁸ El optimizador de los SGBD relacionales tiene como objetivo la búsqueda, para cada sentencia de manipulación, de técnicas eficientes de acceso a los datos almacenados. En los lenguajes navegacionales el usuario indica el camino a seguir para acceder a los datos, pero en los lenguajes de especificación es un componente del SGBD, el optimizador (o planificador de consultas), el que ha de ocuparse de esta tarea. Si el modelo de datos conoce una determinada restricción, el optimizador podrá apoyarse en ella a fin de mejorar la eficiencia en el acceso a la base de datos; si las restricciones, aunque existan, son ajenas al SGBD, éste no puede revisar todos los programas para tener conocimiento de estas restricciones y, por tanto, el optimizador no las tendrá en cuenta.

⁴⁹ Por ejemplo, el SQL Forms de Oracle da facilidades para la definición de la integridad referencial.

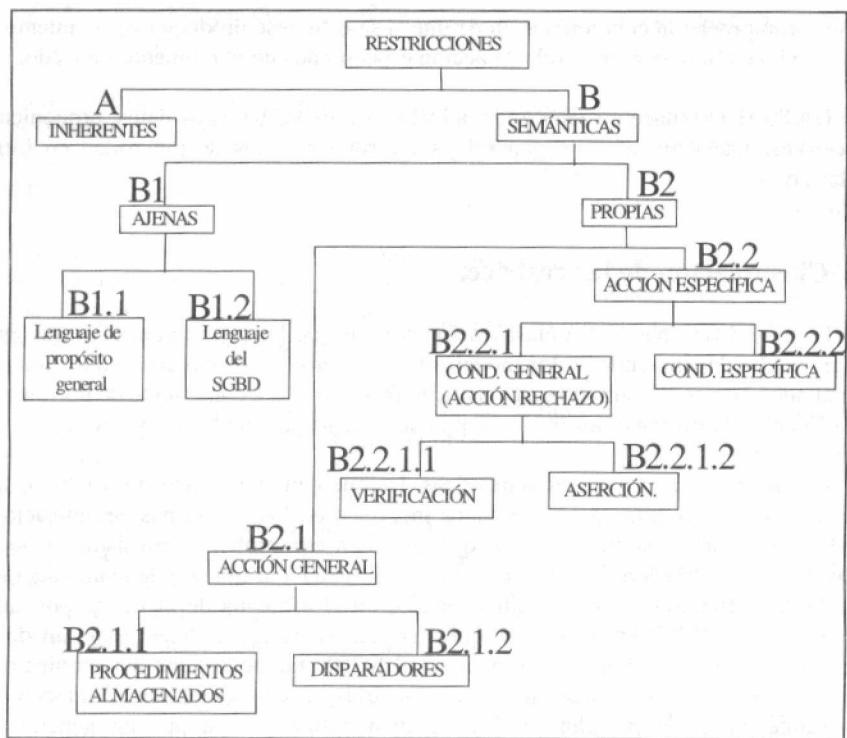


Figura 1.21. Jerarquía de clasificación de las restricciones

Las *restricciones propias del modelo de datos* se especifican al definir el esquema mediante las facilidades que proporciona la función de definición de datos, almacenándose en la base de datos (no en los programas), por lo que no pueden ser violadas por ninguna aplicación, es decir, cualquier actualización está obligada a respetarlas.

Dependiendo de los componentes (acción y/o condición) que haya que especificar al definir una restricción y a la forma de hacerlo (declarativa o procedimental) tendremos distintos tipos de restricciones. En primer lugar, dependiendo de que sea o no preciso definir la acción tendremos restricciones de *acción general* y restricciones de *acción específica*; en las primeras es preciso programar un procedimiento que determine la acción que hay que llevar a cabo, mientras que en las segundas la acción (en general rechazo, aunque puede ser otra, bien predeterminada bien elegida mediante opciones) está implícita en la misma restricción.

Las de acción general son las más flexibles de las restricciones propias del modelo, pero suponen una importante carga de programación; además, el sistema

desconoce su semántica, ya que pueden estar escritas en cualquier lenguaje, por lo que no le es posible comprobar su consistencia ni tampoco el optimizador puede tenerlas en cuenta a fin de mejorar el acceso físico. Son por tanto muy parecidas, en estos aspectos, a las restricciones ajenas al modelo; sin embargo, se diferencian de ellas en que están almacenadas en el esquema, en que su descripción se realiza en el momento de definir el esquema y, principalmente, en que no pueden ser violadas por los programas de aplicación. Se dividen a su vez en:

- *Procedimientos almacenados*

Se definen totalmente de forma procedimental (tanto la acción como la condición); son, entre todas las restricciones propias, las que más se asemejan a las ajenas al modelo⁵⁰.

- *Restricciones de disparo*

Los disparadores⁵¹ permiten definir restricciones de integridad, a las que hemos llamado *restricciones de disparo*. En ellas se formula una condición de forma declarativa, mediante una proposición lógica; el *cumplimiento* de la misma "dispara" una acción especificada de forma procedimental; es decir, al contrario de lo que pasa en otros tipos de restricciones, la acción se desencadena ante un resultado de *cierto* en la condición⁵². La acción ha de programarse mediante un procedimiento, lo que proporciona bastante flexibilidad. Son una mezcla de ambos enfoques declarativo (en la formulación de la condición) y procedimental (en la especificación de la acción).

En las *restricciones de acción específica*, la acción (que puede ser de rechazo o de otro tipo) está determinada por la misma restricción. Son totalmente declarativas porque la acción no hay que definirla y la condición, en el caso de que haya que especificarla, se define de forma declarativa. Dentro de ellas es preciso distinguir:

- *Restricciones de condición general* (se denominan a veces *restricciones generales*⁵³).

La condición se define mediante una proposición lógica, por lo que su complejidad es arbitraria (dentro de lo que permite la proposición) proporcionando, por tanto, una mayor flexibilidad que las restricciones de condición específica que analizaremos posteriormente. La operación será

⁵⁰ Hemos dudado mucho si incluirlas entre las restricciones ajenas al modelo. El haberlas clasificado por fin como restricciones propias se debe, como acabamos de indicar, al hecho de que se definen en el esquema y, lo que consideramos aún más importante, que no pueden ser violadas por los programas.

⁵¹ Los disparadores (*triggers*) son también instrumentos de las bases de datos activas que permiten definir reglas distintas de las restricciones; en realidad, las restricciones, como ya hemos indicado, no son otra cosa que un tipo especial de reglas de las bases de datos activas en las que el evento que las activa es una actualización.

⁵² Si no se especifica condición (la condición es opcional) se considera el resultado como cierto y la acción se dispara siempre que tiene lugar la operación.

⁵³ Dentro de lo que algunos autores denominan *restricciones generales* se considera también a veces los disparadores.

cualquiera que implique asignar valores a los atributos que aparecen en la condición, es decir, una actualización. No se declara la acción porque este tipo de restricción lleva siempre asociado el *rechazo* de la operación cuando no se cumple la condición, es decir, el sistema evalúa la condición y si el resultado es cierto se actualiza, y si no es cierto, no se lleva a cabo la operación.

El SQL 92 distingue dos tipos de restricciones que son de condición general (definida mediante una expresión lógica) y de acción específica (rechazo), que consideramos de interés incluir en esta clasificación:

– *Restricciones de verificación*

Son las cláusulas "CHECK" de algunos lenguajes. La expresión lógica mediante la cual se formula la condición está definida sobre uno o varios atributos de un mismo elemento (por ejemplo, una cierta tabla o un determinado dominio), cuyos valores (o cuyos cambios) han de atenerse a lo especificado en dicha expresión. Este tipo de restricciones se declara al tiempo que se define el elemento del esquema al cual afecta; por ejemplo, si el estado civil de toda persona menor de 14 años tiene que ser soltero, se ha de incluir una restricción de verificación que compruebe que, para persona, "Edad < 14 años y Estado_Civil = Soltero". Puede dárselas un nombre, pero, al no tener existencia por sí mismas sino dentro del elemento al que afectan, el nombre no es obligatorio.

– *Restricciones de aserción*

Son análogas a las anteriores, aunque se diferencian de ellas en que pueden estar referidas a más de un elemento del esquema (por ejemplo, varias tablas), ya que tienen existencia por sí mismas; por tanto, exigen un nombre.

• *Restricciones de condición específica.*

Reglas de "caso especial" en palabras de CODD (1993) y restricciones "implícitas" para ELSMARI (1989), aunque no siempre está muy clara la distinción que éste hace entre restricciones implícitas y explícitas. Los distintos modelos de datos, cuando se definen los elementos de su esquema, facilitan opciones que son en realidad restricciones; por ejemplo, el modelo relacional, al definir una tabla, da la opción de decir que un atributo o un conjunto de atributos de la misma constituyen una clave primaria, lo que lleva consigo la prohibición de que en la base de datos dos filas de una tabla tengan el mismo valor para estos atributos. Otra restricción específica del modelo relacional es la definición de clave ajena; en este caso, la acción, ante un intento de violación por borrado o modificación de una tupla, la determinará el usuario eligiendo alguna de las opciones que se le ofrecen (según el estándar SQL92, impedir la operación –NO ACTION–, borrar o modificar en cascada –CASCADE–, etc.). Estas restricciones, propias de cada modelo, se declaran

directamente en el esquema mediante opciones que permite el modelo; su intento de violación por una operación de actualización (inserción, borrado o modificación) da como resultado una acción bien determinada⁵⁴. Las hemos llamado restricciones *específicas* para diferenciarlas de todas aquellas en las cuales se declara de forma general la condición o la acción de la restricción; por tanto, estas restricciones se caracterizan porque en ellas no se especifica ninguno de los componentes de una restricción, no existiendo la posibilidad de declarar una condición de tipo general, aunque sí es posible elegir entre opciones que ofrece la misma restricción.

Se podría considerar, como hace ELSMARI (1989), que son únicamente las restricciones específicas las que forman parte de la definición del esquema, y que para las restricciones generales, como propone DATE (1990), debiera existir un lenguaje de definición independiente del modelo, que formaría parte del subsistema de gestión de integridad.

En el cuadro que aparece en el anexo a este capítulo se resume la caracterización de las distintas categorías de restricciones definidas anteriormente.

Existe otra clasificación de las restricciones, conceptualmente ortogonal con la anterior, que las divide en restricciones de *estado* y restricciones de *transición*. En general, las restricciones se aplican a un determinado estado de una base de datos y no hay necesidad de conocer los estados anteriores para saber si se cumple o no la condición, se trata de las restricciones denominadas de *estado*; por ejemplo, la restricción que obliga a que con una edad menor de 14 años el estado civil sea soltero, es una restricción de estado (también llamada *estática*), ya que sólo es necesario comprobar qué ocurre, en ese momento, en la base de datos sin tener en cuenta estados anteriores. Sin embargo, hay veces en que la restricción es de *transición* (o *dinámica*) porque hay que aplicarla a la transición entre dos estados; por ejemplo, el cambio de estado civil, o la que impone que el salario de un empleado no puede disminuir.

A veces se dividen también las restricciones entre aquellas que afectan a un único ejemplar (como "Edad < 14 y Estado_Civil = Soltero") o a más de un ejemplar (como que el sueldo de un empleado del departamento tiene que ser menor que el de su jefe). Dentro de estas últimas es posible distinguir aquellas que sólo afectan a algunos ejemplares y las que afectan a todos los ejemplares de un cierto tipo, en general aplicando alguna medida estadística como un promedio; por ejemplo, que la media de los sueldos de todos los empleados tiene que ser inferior a un cierto valor.

⁵⁴ La acción, por lo general, es un rechazo de la operación, pero también puede ser otra distinta (por ejemplo, el borrado de otras tuplas), dándose en algunas de estas restricciones la posibilidad de optar entre varias acciones siempre bien determinadas.

También se puede distinguir entre las *restricciones de valor* y las *restricciones estructurales*. Las primeras son todas aquellas en las que en la condición se comparan los valores que pueden tomar las propiedades (casi todos los ejemplos que hemos puesto son relativos a este tipo de restricción); existen, además de éstas, otras restricciones que imponen limitaciones a la estructura de los elementos del modelo⁵⁵ (como que un atributo no puede tomar más que un valor o la de que uno o más atributos constituyen la clave primaria⁵⁶), son las *restricciones estructurales*.

En otros casos, se clasifican las restricciones atendiendo a los elementos del modelo de datos a los cuales afecta la condición; así tendríamos restricciones sobre dominios, o sobre relaciones, etc. Esta clasificación es propia de cada modelo de datos, ya que los elementos de los modelos varían de unos a otros.

6. LOS MODELOS DE DATOS EN EL PROCESO DE DISEÑO DE UNA BASE DE DATOS

Se conoce como proceso de diseño de una base de datos al conjunto de etapas necesarias para pasar de una determinada realidad (Universo del Discurso) a la base de datos que la representa. Los modelos de datos desempeñan un importante papel en el proceso de diseño de una base de datos al ofrecernos facilidades de abstracción que nos ayudan a representar la realidad.

Los objetivos que persigue todo modelo de datos son de dos tipos:

- a) **Formalización**, ya que el modelo de datos permite definir formalmente las estructuras permitidas y las restricciones; también el modelo de datos establece la base para la definición de un lenguaje de datos y facilita una apreciación más objetiva de la rigidez o flexibilidad de las estructuras de datos, ayudando a la comparación formal de distintos modelos de datos y a la evaluación de los SGBD.
- b) **Diseño**, ya que el modelo de datos es un elemento fundamental en el desarrollo de una metodología de diseño de bases de datos, en el cual se basan los otros componentes de la metodología (lenguajes, documentación y otras herramientas); permiten, además, prever el impacto de los cambios del mundo real en nuestro sistema de información.

⁵⁵ No confundir con las restricciones inherentes, en las cuales las limitaciones de las estructuras no vienen impuestas por el mundo real sino por el mismo modelo.

⁵⁶ La exigencia de un modelo de la obligatoriedad de la clave primaria es una restricción inherente, que no hay que confundir con la declaración de que uno o más atributos constituyen la clave primaria que es una restricción semántica específica.

La magnitud de la distancia que separa el mundo real de las estructuras de datos almacenadas en los soportes físicos de un computador hace aconsejable abordar el proceso de diseño de una base de datos dividiéndolo en una serie de etapas sucesivas, cada una de las cuales se apoyará en un tipo distinto de modelo de datos adecuado a la correspondiente fase del diseño. Antes de presentar estas etapas hemos de insistir en el concepto de Universo del Discurso (UD) como *visión* que del mundo real tiene el diseñador (véase figura 1.22).

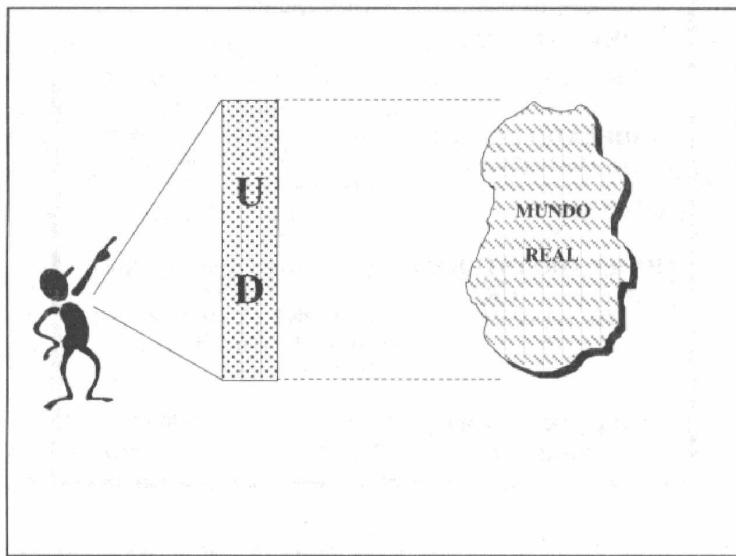


Figura 1.22. Universo del discurso y mundo real

El primer paso en la concepción de una base de datos es definir el universo del discurso, fijando para ello una serie de objetivos sobre el mundo real que se va a analizar; así, por ejemplo, de un mismo mundo real, como puede ser el que constituye una universidad, podemos definir universos del discurso tan distintos como uno relativo a los cursos de doctorado, con los profesores que los imparten, sus departamentos y áreas, los estudiantes, etc.; y otro, concerniente a la gestión de los empleados de la universidad (tanto docentes como no docentes), nóminas, contabilidad, facturación, etc. Es decir, el mundo real es el mismo, pero nuestro objetivo en el primer caso es la docencia de tercer ciclo, mientras que en el segundo es la gestión económica y de personal de la universidad.

Una vez definido el universo del discurso acerca del cual deseamos recoger información en nuestra base de datos, hemos de proceder a su estructuración, paso a paso, hasta llegar a la base de datos física tal como se muestra en la figura 1.23.

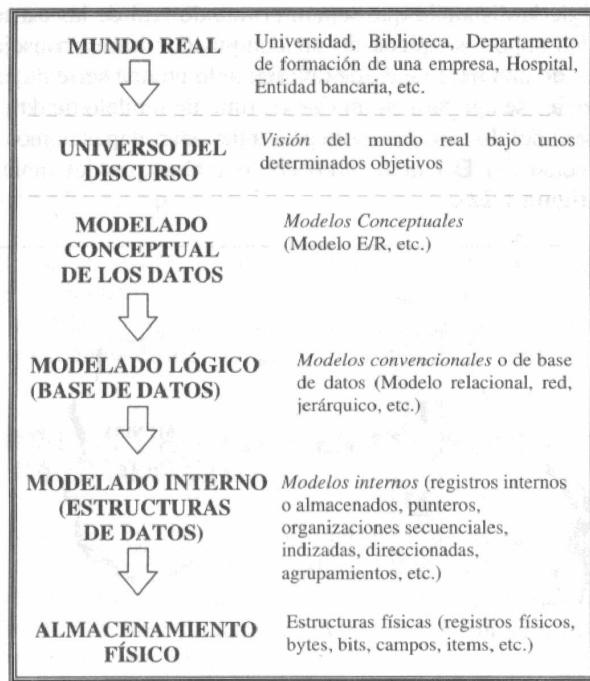


Figura 1.23. Etapas en el diseño de una base de datos y tipos de modelos en los que se apoyan

En el diseño de una base de datos es conveniente distinguir la fase de modelado conceptual, que es la descripción del mundo real (empresa o administración) de acuerdo con un modelo conceptual (en la metodología que proponemos será el modelo Entidad/Interrelación –E/R–). El modelo conceptual deberá ser altamente semántico e independiente del SGBD en el que posteriormente se vaya a realizar la implementación de la base de datos, y también independiente de la fase de diseño lógico en la cual se ha de obtener un esquema lógico que responda a la estructura específica (por ejemplo, relacional) del SGBD que se aplique en cada caso concreto; esquema que está sometido a las restricciones que imponga el correspondiente modelo. Posteriormente, el diseño interno permitirá obtener el esquema interno y, por último, se implementará la base de datos física en los soportes secundarios.

Los modelos de datos soportados por los SGBD (Jerárquico, Codasyl y, en la actualidad, principalmente Relacional) no tienen el suficiente poder expresivo para captar la semántica del mundo real que al usuario le gustaría ver plasmada en su base de datos, y sus conceptos, al estar orientados hacia el computador, no suelen ser fácilmente comprendidos por los usuarios. Por estas razones surgen modelos más semánticos y cercanos al usuario, como el modelo Entidad/Interrelación (CHEN,

1976). TSICHRITZIS (1982) insiste en que la forma en que los modelos de datos que hemos llamado convencionales permiten expresar los requisitos de información no se corresponden con la forma en que las personas en general deberían percibir la información representada en la base de datos, y que el papel de los modelos de datos en el diseño de las bases de datos no se debe limitar a abstraer las propiedades de la futura base de datos, sino que ha de servir también como un medio de comunicación entre el usuario y el diseñador. Un modelo de datos cuyos constructores estén basados en el modo en el que las personas perciben los hechos y se los comunican asegura que las estructuras impuestas por el modelo de datos no entrarán en conflicto con las estructuras tal como las percibe el ser humano, es decir, serán modelos orientados a representar la **información** con todo su contenido semántico sin estar “contaminada” por conceptos cercanos al computador, por lo que a estos modelos se los llama **infológicos** (LANGEFORS, 1980 y SUNDGREN, 1974). En cambio, los modelos convencionales están enfocados a representar los **datos**, de acuerdo a cómo han de ser interpretados por los SGBD, recibiendo el nombre de modelos **datológicos**.

El problema se encuentra en que los modelos conceptuales no suelen estar implementados en los SGBD, por lo que los SGBD no “entienden” la estructura conceptual, teniéndose que transformar ésta en una estructura lógica adaptada a las exigencias y restricciones del modelo de datos propio del SGBD que vaya a ser utilizado⁵⁷. Es decir, se debe llegar a un esquema lógico admitido por el SGBD, obteniendo posteriormente el esquema interno, en donde el objetivo es conseguir la máxima eficiencia de cara a la máquina y al problema concreto.

A veces se prescinde de la etapa de modelado conceptual, y el diseñador, sin una metodología precisa, hace una abstracción del mundo real, representándolo directamente en las estructuras del SGBD concreto que va a utilizar; forma de actuar que no consideramos aconsejable, ya que la aplicación de una rigurosa metodología de desarrollo de bases de datos⁵⁸, basada en su primera etapa en un modelo conceptual, ayuda a conseguir una mejor representación del mundo real.

La estructura física resultante del proceso de diseño se ha de *rellenar* con los valores (ejemplares) que se obtienen por observación de los sucesos del mundo real. Estas *cadenas de bits* estarían totalmente carentes de significado si no dispusiéramos de los medios que nos permiten recorrer el camino inverso, pasando de nuevo al mundo real con ayuda del lenguaje de manipulación, por medio del cual actualizaremos o recuperaremos los datos almacenados en la base, reincorporándoles su contenido semántico y obteniendo la información que necesita el usuario.

⁵⁷ Muchas herramientas CASE (*Computer Aided Software Engineering*) pueden realizar de forma automática la transformación de un esquema conceptual en la estructura interna de los SGBD comerciales más extendidos.

⁵⁸ Incluida, en muchos casos, en una metodología general de desarrollo de sistemas de información.

ANEXO. CLASIFICACIÓN DE LAS RESTRICCIONES

A) Inherentes

- Impuestas por el modelo
- No tienen que ser definidas por el usuario, ya que se encuentran en el propio modelo
- Se activan en el momento de la definición del esquema cuando se produce un intento de violación
- Se rechaza todo esquema que no cumple estas restricciones
- Introducen rigideces en el modelo

B) Semánticas

- Impuestas por el universo del discurso
- Tienen que ser definidas por los usuarios (diseñadores)
- Se activan en el momento de la actualización de la base de datos
- Se rechaza todo ejemplar que no cumpla estas restricciones (o se ponen en marcha otros medios a fin de que no se produzca un estado inconsistente)
- Ayudan a capturar la semántica de los datos y a conseguir su consistencia

B1) AJENAS

- Se especifican en los programas de aplicación
- No están almacenadas en el esquema de la base de datos
- Pueden ser violadas por actualizaciones en las que no se haya programado la restricción.
- El SGBD no puede comprobar si son consistentes en sí mismas
- El optimizador no puede tomarlas en consideración
- Proporcionan el máximo de flexibilidad
- Pueden ser programadas en un lenguaje de propósito general o en algún lenguaje propio del SGBD
- Suponen una importante carga de programación y de mantenimiento

B2) PROPIAS

- Se especifican en el esquema
- Están almacenadas en el esquema de la base de datos
- No pueden ser violadas por ninguna actualización

B 2.1) Acción general

- Es obligatorio especificar la condición y la acción
- Son procedimentales (al menos en parte, ya que la acción se especifica siempre mediante un procedimiento)
- Suponen carga de programación
- Es muy difícil (prácticamente imposible en la mayor parte de los casos) que el SGBD pueda comprobar su consistencia
- El optimizador no puede tomarlas en consideración
- Hasta ahora no están estandarizadas⁵⁹
- Están muy ligadas a los productos
- Son muy flexibles
- Tienen nombre y existencia propia dentro del programa

B 2.1.1) Procedimientos almacenados

- Es obligatorio especificar la condición (además de la acción)
- Son totalmente procedimentales
- Pueden ser tan complejas como imponga la semántica del mundo real (tanto en la condición como en la acción)
- Son las más flexibles dentro de las restricciones propias

B 2.1.2) Disparadores

- Combinan los enfoques declarativo (en la condición) y procedural (en la acción)
- Pueden ser tan complejas como imponga la semántica del mundo real en cuanto a la acción, y bastante complejas en la condición (todo lo que permite la proposición lógica mediante la que se expresa la condición)
- El cumplimiento de la condición (la evaluación de la proposición con resultado de “cierto”) dispara la acción⁶⁰.
- Son más flexibles que las restricciones de acción específica

B 2.2) Acción específica

- La acción está implícita en la misma restricción, por lo que no hay que definirla (está determinada)

⁵⁹ En el estándar conocido como SQL:1999, que será aprobado este año, ya se estandarizan los disparadores.

⁶⁰ En las de acción específica ocurre lo contrario: la acción se dispara cuando la condición no se cumple. En los procedimientos almacenados “en teoría” puede ocurrir tanto una cosa como otra, dependerá de cómo se proponga el procedimiento (en la práctica, los productos pueden imponer ciertas limitaciones).

- Son declarativas, puesto que no se especifica la acción y la condición, si se define, es de forma declarativa
- El no cumplimiento de la condición (su evaluación con resultado de “*no cierto*”⁶¹) lleva a aplicar la acción
- Podrían ser definidas mediante un lenguaje de tipo general
- El SGBD puede comprobar si son consistentes en sí mismas
- El optimizador puede tomarlas en consideración
- No suponen carga de programación, sólo de definición

B 2.2.1) Condición general

- No se especifica la acción, que es siempre el rechazo (el no cumplimiento de la condición lleva consigo el rechazo de la actualización)
- Es obligatorio declarar la condición mediante una proposición lógica que permite condiciones de complejidad arbitraria (las que permite la proposición)
- Además de la condición, se puede especificar algún otro componente (en especial el momento)
- Son más flexibles que las de condición específica
- Es más difícil optimizar su ejecución que en el caso de las de condición específica

B 2.2.1.1) Verificación

- No tienen existencia por sí mismas
- Su definición forma parte de la definición del elemento afectado por la restricción
- Se aplican a un único elemento y aunque pueden afectar a otros, en este caso se complica su definición (por lo que es más adecuado utilizar aserciones)
- Pueden no tener nombre

B 2.2.1.2) Aserción

- Tienen existencia por sí mismas
- Se definen con independencia de cualquier elemento del esquema (no forma parte de la definición de ningún elemento)
- Pueden afectar a más de un elemento
- Tienen nombre

⁶¹ Decimos “*no cierto*” en lugar de *falso* porque incluye el “*quizás*” en el caso de admitirse nulos y ser “*quizás*” el resultado de la evaluación.

B 2.2.2) De condición específica

- Son opciones proporcionadas por el propio modelo
- No se especifica ninguno de los componentes relativos a una restricción (ni la operación, ni la condición, ni la acción)
- Son poco flexibles
- El optimizador puede tomarlas en consideración
- Su ejecución puede ser más fácilmente optimizada que las de condición general