TIPOS DE BASES DE DATOS.

Al igual que cuando se habla, p.ej., de coches no existe un único modelo, ni una sola marca, ni siquiera una sola tecnología sobre su funcionamiento, cuando se trabaja con bases de datos ocurre una cosa parecida: no existe una sola marca, sino varias, y además cada marca puede tener diferentes productos cada uno de ellos apropiado a un tipo de necesidades.

Sin embargo, la división que vamos a hacer aquí de las bases de datos será en función de la tecnología empleada en su funcionamiento. Hablando de coches tenemos los tradicionales de motor a gasolina, los de gasóleo, los turbodiesel, los que funcionaban con gasógeno, y mucho menos frecuentes los coches solares o incluso los de propulsión a chorro; pues bien, hablando de bases de datos tenemos que las más utilizadas son la bases de datos relacionales, las más antiguas son las jerárquicas y en red, y las más avanzadas son las orientadas a objetos, y las declarativas. Estas se diferencian como hemos dicho, en la forma de trabajar con los datos y en la concepción o mentalidad que el usuario debe adoptar para interactuar con el sistema.

Al igual que en el caso de los coches, unos sistemas consumen más recursos que otros. P.ej., los sistemas declarativos consumen tanta memoria y tiempo de funcionamiento como queroseno un coche de propulsión a chorro; una base de datos en red puede resultar tan penosa de manejar como un coche antiguo con gasógeno. En el término medio podemos decir que lo más empleado actualmente (aunque algunos pueden decir que lo más contaminante) es el sistema relacional, al igual que los coches de gasolina o gasóleo.

Para describir cada uno de los modelos o paradigmas en que se basan las bases de datos, vamos a seguir un criterio histórico, estudiando primero los sistemas más antiguos para pasar por último a los sistemas más avanzados.

Modelos tradicionales.

Estudiaremos en este epígrafe los sistemas de bases de datos más utilizados hasta el momento, aunque quizás en pocos años, los sistemas orientados a objeto deban ser incluídos en este epígrafe.

No obstante, para ser realistas, hay que recordar que no es verdad que la práctica totalidad de las empresas dejen descansar sus datos sobre bases de datos de alguno de estos tres tipos. De hecho, la triste realidad es que muchas entidades, especialmente los bancos, por el hecho de haber sido las primeras en informatizarse, siguen teniendo todos sus datos sobre ficheros electrónicos simples, manejados directamente a través del S.O., y no con la intervención de un S.G.B.D. que facilite su gestión y mantenimiento.

No por ello debe pensar el estudiante que, en tal caso, el emplear una base de datos es algo inútil y falto de esencia. De hecho, la realidad que circunda a Málaga, no es una realidad de grandes empresas, excepto quizás los bancos, sino un mundo de pequeñas y medianas empresas, que por haberse incorporado más tardíamente al mundo de los ordenadores, sí que han adoptado las bases de datos como responsable del almacenamiento de sus datos. Por otro lado, los sistemas de reserva de billetes, y otros sistemas que tienen en común una enorme cantidad de usuarios dispersos efectuando peticiones al sistema central, también utilizan bases de datos, debido a que el concepto de transacción que éstas facilitan hace mucho más fácil su programación, asegurando además el buen funcionamiento

global de toda la red de terminales de usuario.

Modelo jerárquico.

El sistema jerárquico más comúnmente conocido es el sistema IMS de IBM. Esta base de datos tiene como objetivo establecer una jerarquía de fichas, de manera que cada ficha puede contener a sus vez listas de otras fichas, y así sucesivamente. P.ej., una ficha de clientes puede contener una lista de fichas de facturas, cada una de las cuales puede contener a su vez una lista de fichas de líneas de detalle que describen los servicios facturados.

Una base de datos jerárquica está compuesta por una secuencia de bases de datos físicas, de manera que cada base de datos física se compone de todas las ocurrencias de un tipo de registro o ficha determinada.

Una ocurrencia de registro es una jerarquía de ocurrencias de segmento.

Cada ocurrencia de segmento está formada por un conjunto de ocurrencias o instancias de los campos que componen el segmento.

P.ej., en la figura siguiente tenemos una ocurrencia del tipo de registro Curso, de manera que como cabeza principal tenemos una instancia del segmento curso, de la cual dependen una o varias instancias de los segmentos Requisito y Oferta; a su vez, de Oferta dependen otros que son Profesor y Estudiante.

Cabe distinguir en este punto entre el concepto de tipo de registro, y ocurrencia o instancia de registro. El tipo define la estructura general que debe poseer, o sea, los campos de cada uno de sus segmentos, y la estructura jerárquica entre ellos. Una instancia es un valor de un tipo de registro. Para que quede más claro, un tipo de registro es como un tipo de persona: blanco, negro, amarillo, aceitunado, etc., mientras que una instancia es una persona concreta perteneciente a uno de estos tipos: Pablo Picasso, Nelson Mandela, Mao Tse Tung, Toro Sentado, etc.

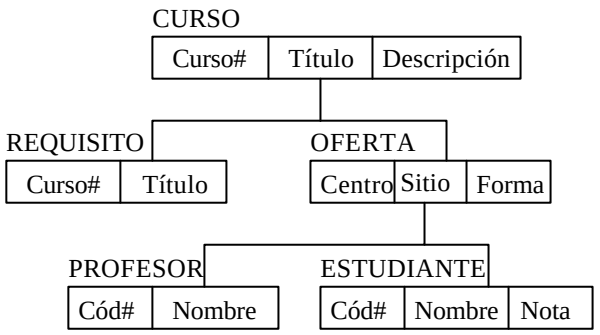


Figure 1. Ejemplo de tipo de registro. Los tipos de segmento son CURSO, REQUISITO, OFERTA, PROFESOR, y ESTUDIANTE. CURSO es el tipo de segmento raíz.

De esta forma, al segmento que se halla a la cabeza de un registro, se le llama segmento padre, y se llama segmentos hijo a los que dependen de él.

Para movernos por un registro de estructura jerárquica lo que se hace es posicionarse inicialmente en la raíz de una instancia, e ir navegando por sus hijos según nos convenga consultando o modificando los datos pertinentes.

Una base de datos de este tipo, no permite el acceso directo a las instancias de un segmento hijo, si no es seleccionando previamente las instancias de los padres de los que depende. P.ej., no se

puede seleccionar un estudiante si no es previa selección de una oferta y de un curso.

Las instancias de un mismo segmento que dependen de una misma instancia padre se llaman instancias gemelas. en el ejemplo, las instancias:

1 J. Toro 9
2 F. Mora 7
3 A. Gil 3

son ocurrencias gemelas, pues todas dependen de la instancia

UMA Clase Pizarra

del tipo de segmento Oferta.

Nótese que si el administrador decide ocultar a determinados usuarios ciertos segmentos (debido a que no tienen por qué tener conocimiento de su existencia), hay que eliminar también todos los segmentos hijos que dependen de él. P.ej., si alguien no debe tener acceso a las ofertas, sólo podrá acceder a los Cursos y a los Requisitos, pero tampoco a los profesores ni a los estudiantes.

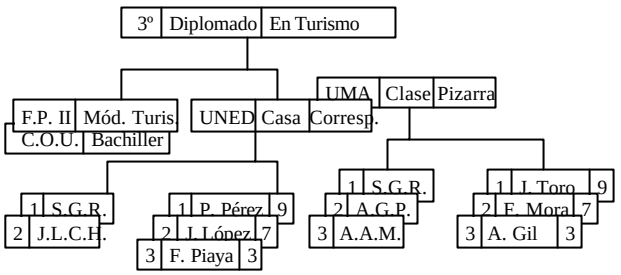


Figure 2. Instancia de un registro.

No profundizaremos más en este sistema; tan sólo indicar algunos de sus problemas:

- La jerarquía existente entre los tipos de objetos que se manipulan (Cursos, Estudiantes, Profesores, etc.), y las dependencias existentes, hacen que sea imposible el acceso directo a instancias de cada una de ellos, con lo que se pierde en independencia y facilidad de uso.
- Si un mismo segmento debe participar en varios tipos de registro, deben incluirse mecanismos que eviten la repetición de datos. Es más, en el ejemplo anterior se ve que una instancia del segmento Profesor:

1 S.G.R.

aparece dependiendo de la oferta de la UNED, y de la UMA. Está claro que los datos no se deben repetir, ya que ello puede provocar que posteriormente se modifique una de las instancias pero no la otra, con la consiguiente inconsistencia entre ambas copias de los mismos datos.

Modelo en red.

Podemos considerar al modelo de bases de datos en red como de una potencia intermedia entre el jerárquico y el relacional que estudiaremos más adelante. Su estructura es parecida a la jerárquica aunque bastante más compleja, con lo que se consiguen evitar, al menos en parte, los problemas de aquél.

Los conceptos fundamentales que debe conocer el administrador para definir el esquema de una base de datos jerárquica, son los siguientes:

- Registro: Viene a ser como cada una de las fichas almacenadas en un fichero convencional.
- Campos o elementos de datos. Son cada uno de los apartados de que se compone una ficha.
- Conjunto: Es el concepto que permite relacionar entre sí tipos de registro distintos.

Podemos imaginar los registros simplemente como fichas de un fichero. Para ilustrar el concepto de conjunto, supongamos que tenemos un tipo de registro de clientes, y un tipo de registro de vuelos de avión, y supongamos que queremos asociar ambas informaciones, de manera que para cada vuelo queremos saber cuáles son los pasajeros que viajan en él. La forma de hacerlo es a través de un conjunto. Un conjunto relaciona dos tipos de registro. Uno de ellos es el registro propietario del conjunto, y el otro es el miembro. Veamos el diagrama de la figura siguiente quenos aclarará las cosas un poco más. Son los diagramas de Bachman.

Cada tipo de conjunto, posee, a su vez, una serie de ocurrencias de conjunto, donde cada ocurrencia está formada por una instancia del tipo propietario, y una, varias o ninguna instancia del tipo miembro. P.ej. una ocurrencia de conjunto puede ser:

IB-763 Málaga Helsinki 27/8/97 17:00
33387698-K Juan Linares
83698637-H Pedro Hernández
24885764-G Luis Caro
64653627-J Pablo Mármol

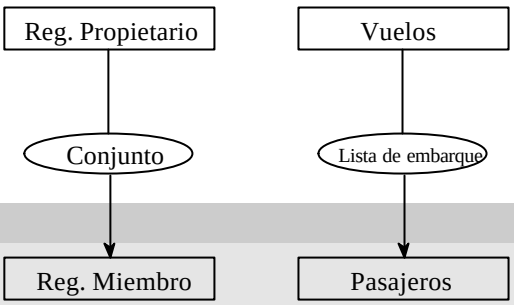


Figure 3. A la izquierda podemos ver el formato general de un conjunto, y a la derecha, el conjunto concreto que nos soluciona saber la lista de embarque de cada vuelo.

Una restricción bastante importante de este modelo, es que una ocurrencia de registro miembro puede pertenecer como máximo a una sola instancia de un determinado conjunto, aunque puede participar en varios tipos de conjuntos distintos.

Este modelo en red es más potente que el modelo jerárquico, ya que aquél puede simularse, aplicando una jerarquía de conjuntos en varios niveles. P.ej., el ejemplo jerárquico del punto anterior quedaría ahora como:

Por otro lado, en un conjunto concreto, el tipo de registro propietario no puede ser, a su vez, el mismo que el tipo de registro miembro, o sea, un mismo tipo de registro no puede intervenir en el mismo conjunto como propietario y como miembro a la vez.

Para ilustrar por qué el modelo en red es más potente que el modelo jerárquico, basta con observar un conjunto como el siguiente:

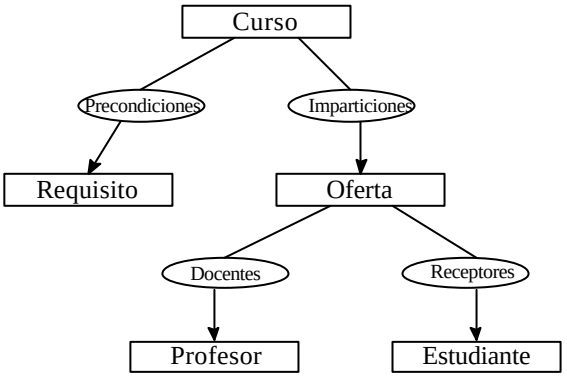


Figure 4. Cómo simular el ejemplo jerárquico mediante el modelo en red.

Aquí, un elemento de A puede poseer varios de B, mediante el conjunto A-B; a su vez, los de B pueden poseer a los de A, mediante B-A, y así sucesivamente cuantas veces se quiera. Este ejemplo no se puede hacer en el modelo jerárquico, pues el número de niveles varía dinámicamente.

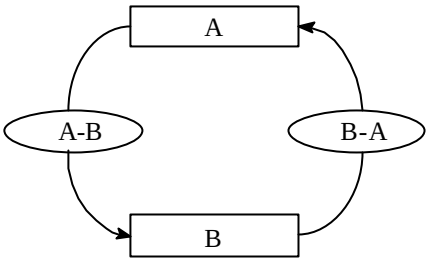
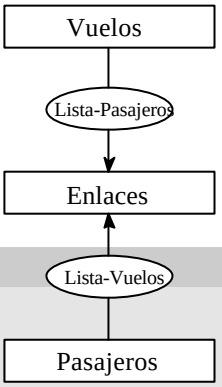


Figure 5. Este diagrama representa relaciones entre tipos de registro que no pueden expresarse según el esquema jerárquico.

Que una misma instancia de registro miembro no pueda aparecer en más de una instancia de conjunto, hace que sea difícil de expresar algunas situaciones. P.ej., en el caso de las lista de embarque, está claro que no sólo cada vuelo lo componen varios pasajeros, sino que, además, un mismo pasajero ha podido embarcar en varios vuelos a lo largo de su vida. ¿Cómo representar esta situación?.

La solución a este problema es algo artificiosa, y pasa por la creación de tipos de registro llamados enlaces. La figura siguiente ilustra la solución:



Así, cada pasajero se relaciona con una lista de vuelos, que viene dada por una serie de códigos, y cada vuelo se relaciona con una lista de pasajeros que vendrá dada por otra serie de códigos. P.ej., para representar la siguiente estructura:

IB-763 Málaga Helsinki 27/8/97 17:00
33387698-K Juan Linares
83698637-H Pedro Hernández
24885764-G Luis Caro
64653627-J Pablo Mármol

IB-722 Málaga Zurich 21/9/97 7:00
63553572-K Alfredo Sánchez
24746928-G Antonio Fernández
64653627-J Pablo Mármol

que es ilegal en el modelo en red, ya que una misma instancia de pasajero, a saber:
64653627-J Pablo Mármol
aparece en más de una instancia del conjunto lista de embarque, se emplearían las siguientes ocurrencias de conjuntos:
Lista-Pasajeros
IB-763 Málaga Helsinki 27/8/97 17:00
1

2
3
4
IB-722 Málaga Zurich 21/9/97 7:00
5
6
7
Lista-Vuelos
33387698-K Juan Linares
1
83698637-H Pedro Hernández
2
24885764-G Luis Caro
3
64653627-J Pablo Mármol
4
7
63553572-K Alfredo Sánchez
5
24746928-G Antonio Fernández
6

Con lo que el pasajero Pablo Mármol se enlaza con 4 y con 7, esto es, con los vuelos a Helsinki y a Zurich, solucionándose el problema.

Modelo relacional.

En este apartado tan sólo daremos unas nociones iniciales sobre este modelo, ya que todo nuestro trabajo se basará en él, y será estudiado con mucho mayor detalle en capítulos posteriores.

Este modelo intenta representar la base de datos como un conjunto de tablas. Aunque las tablas son un concepto simple e intuitivo, existe una correspondencia directa entre el concepto informático de una tabla, y el concepto matemático de relación, lo cual es una gran ventaja, pues permite efectuar formalizaciones de una forma estricta mediante las herramientas matemáticas asociadas, como pueda ser el álgebra relacional en el ámbito de las consultas.

Gracias a Dios, no será necesario enfrentarnos con todos estos formalismos propios de los matemáticos, sino que dispondremos de unas herramientas fáciles de manejar que nos permitirán interactuar con la base de datos.

Los conceptos básicos del modelo relacional son:

- Registro: Es algo así como cada ficha de un fichero convencional.
- Tabla: Es un conjunto de fichas de un mismo tipo.

Con estos dos conceptos es posible crear cualquier tipo de datos, y asociarlos entre sí, sin las restricciones propias del modelo jerárquico o en red. P.ej., si necesitamos diseñar una base de datos

para una agencia de alquiler de coches, necesitaremos una tabla en la que se guarde información sobre los coches, como puede verse en la figura.

De esta forma, vemos que cada tabla está compuesta por filas, también llamadas tuplas o registros, cada uno de los cuales posee una serie de campos en los que se almacenan los datos básicos. El esquema de una tabla nos indica los nombres de cada uno de los campos que contiene, así como el tipo de información que debe contener.

Marca	Modelo	Color	Matrícula	Situación
Lamborghi.	Diablo 630	Amarillo	MA-2663-BC	En renta
Ferrari	F-40	Rojo	MA-8870-BC	Disponible
Sbärro R.	Decade	Blanco	VD-870-GTH	Disponible
De Tomaso	Pantera	Blanco	ML-7890-B	En renta
Pontiac	Trans-Am	Negro	KNIGHT	En taller
Austin M.	S3'40	Marrón	CA-5647-AB	Disponible
Jaguar	Destructor	Verde	AD-768-TTY	En renta

Figure 7. Ejemplo de tabla relacional.

Una tabla es para nosotros un conjunto de registros; por tanto, los registros no pueden repetirse.

Para poder acceder a un registro concreto, es necesario hacer una consulta a través de algún campo que identifique a dicho registro, como puede ser p.ej. el número de la matrícula. A este campo especial que identifica cada registro se le llama clave del registro. La figura siguiente ilustra una tabla de clientes.

En el modelo anterior disponíamos de los conjuntos para asociar información entre sí; ¿cómo nos las apañamos para indicar ahora qué cliente se hace responsable de cada coche alquilado? Fácilmente, a través de una nueva tabla que relaciona los clientes con los coches. Para ello dado que cada registro queda identificado por su clave, nos basta con incluir en esta nueva tabla a las claves de ambas tablas, en lugar de todos sus campos. Así, podemos obtener una nueva tabla de alquileres que contenga la matrícula del coche, y el D.N.I. del cliente, tal como se ve en la figura siguiente.

Apellidos	Nombre	D.N.I.	Edad
González Aranda	Javier	75836934	27
Beato Apóstol	Antonio	28836746	43
Campos Ortega	Adriano	82665358	36
Ruíz Rojo	Juan	83667228	35

Figure 8. Tabla de clientes de la agencia de alquiler de coches.

En esta última tabla podemos observar varias cosas interesantes. Por un lado, un cliente se puede responsabilizar de más de un coche, o sea, puede alquilar más de un coche, pues vemos que Javier González Aranda ha alquilado tanto el Lamborghini como el Jaguar. Pero, a su vez, más de una persona puede hacerse cargo de un coche: Javier González Aranda y Adriano Campos Ortega comparten el alquiler del Jaguar.

De esta forma, el modelo relacional soluciona el problema que se planteaba en el caso de las listas de embarque mediante enlaces artificiosos, y lo soluciona de una manera intuitiva a través de las tablas, y eliminando el concepto de conjunto.

Matrícula	D.N.I.
MA-2663-BC	75836934
ML-7890-B	83667228
AD-768-TTY	75836934
AD-768-TTY	82665358

Figure 9. Tabla que relaciona los coches en renta con los clientes que se responsabilizan de ellos.

Este método de expresar los datos facilita además las consultas, que se realizan ahora a través de estas tablas especiales que relacionan a otras tablas. P.ej., si queremos saber los coches que ha alquilado González Aranda, basta con buscar su clave

en la tabla de clientes (75836934), y a continuación ver que matrículas tiene asociadas en la tabla de alquileres (MA-2663-BC, y AD-768-TTY); a continuación, buscamos en la tabla de coches cuales son los coches que poseen esas claves, y obtenemos como resultado: Lamborghini y Jaguar.

Por otro lado, además de los modelos propios de base de datos existentes en la realidad, existen los llamados modelos semánticos, que permiten expresar relaciones entre los datos, independientemente del tipo de base de datos que se emplee finalmente. Uno de estos modelos, el modelo Entidad-Relación, que estudiaremos en el capítulo siguiente, tiene grandes similitudes con el modelo relacional, siendo esta otra gran ventaja del modelo relacional, esto es, se pueden expresar las relaciones entre los datos a través de diagramas fáciles de comprender y de modificar, y, posteriormente, pasar el resultado a un esquema relacional.

Modelos avanzados.

Las bases de datos relacionales han sido y siguen siendo ampliamente utilizadas para una extensa gama de aplicaciones. Sin embargo, el aumento de potencia de los ordenadores personales, ha hecho aparecer nuevas aplicaciones potentes que requieren la utilización de datos complejamente relacionados o con necesidades de consultas muy particulares, como puedan ser p.ej., los sistemas de información geográficos, el diseño de circuitos electrónicos por ordenador, etc.

Otro de los problemas que poseen los sistemas relacionales es el uso de los lenguajes de manipulación y definición de datos, que, aunque son muy simples de manejar directamente por un usuario, son difíciles de insertar en un lenguaje de programación convencional, lo que da lugar a un problema de impedancia o resistencia de un lenguaje a ser utilizado junto con otro.

Otros problemas se refieren a la inclusión del concepto de orden en los registros almacenados. Dado que una tabla es un conjunto de registros, y un conjunto no permite ni repeticiones de sus elementos, ni establece un orden entre ellos, es imposible representar ciertas características de datos muy particulares.

Todos estos problemas han hecho que los investigadores estén buscando alternativas fiables a las bases de datos relacionales, como puedan ser las deductivas, las persistentes, las funcionales, o las orientadas a objetos, pasando por una gama de bases de datos históricas, espaciales, etc.

Dos de ellas son las que están sufriendo mayor empuje por parte de la comunidad informática. Pasamos a describirlas.

Modelo orientado a objetos.

Actualmente, la creación de programas más grandes y complejos, ha hecho avanzar los métodos de programación hacia nuevas formas que permiten el trabajo en equipo de una forma más eficaz y en la que se disminuyen los problemas de coordinación. Uno de estos métodos consiste en la programación orientada a objetos (POO), que trata los problemas desde un punto de vista realista, y modelándo cada uno de ellos como si se tratase de un conjunto de elementos u objetos que interrelacionan entre sí para solucionar el problema.

Para entender mejor esta filosofía, podemos pensar en ella como en el funcionamiento de un reloj de cuerda. Un reloj de cuerda posee numerosos elementos que interactúan entre sí para obtener

como resultado final una determinada posición de las manecillas, que son interpretadas por una persona como la hora actual. Cada uno de estos objetos es un elemento. Cuando un engranaje, por ejemplo, gira, no lo hace por capricho, sino para obtener como resultado el movimiento de otro engranaje, de una cremallera, o de la propia manecilla. De esta forma, cuando el usuario da cuerda a la maquinaria, lo que está haciendo realmente es modificar el estado de un objeto del reloj, normalmente la espiral de la cuerda cuya energía potencial mueve la corona haciendo que un oscilador avance el segundero. A su vez el movimiento del segundero hace avanzar el del minutero, que hace avanzar el de la hora. Si el reloj es de cuco, cada hora se activará la portezuela del cuco que saldrá un número determinado de veces según la hora. De esta manera, una modificación del estado de un objeto por parte de un usuario, desencadena una serie de acciones cuyo objetivo final es solucionar un problema al usuario: darle a conocer la hora exacta. Así, la programación orientada a objetos pretende ser una simulación de los procesos de la realidad.

De este ejemplo podemos sacar varios conceptos útiles:

- **Clase.** Cuando hay varios objetos semejantes, pueden agruparse en una clase. De hecho, todo objeto debe pertenecer a una clase, que define sus características generales.. P.ej., nuestro reloj posee varios engranajes. Serán diferentes, puesto que cada uno de ellos posee un diámetro y un número de dientes distinto, además de poder ser o no helicoidal. Pero al fin y al cabo todos son engranajes. De esta manera cada engranaje pertenece a la misma clase, a pesar de tener unas características particulares que lo diferencian de los demás.
- **Estado.** Son las características propias de cada objeto. Siguiendo con el caso de los engranajes, su estado puede ser el número de dientes, el tamaño, etc. El estado se utiliza especialmente para guardar la situación del objeto que varía con el tiempo. En nuestro caso almacenaríamos la situación en un espacio tridimensional, y la posición o postura en que se encuentra.
- **Encapsulación.** Cada objeto es consciente de sus propias características. El engranaje «sabe» que si recibe una fuerza en uno de sus dientes, debe girar, y lo sabe porque obedece a unas leyes físicas. En el caso de un programa, es el programador el que debe indicarle al objeto como comportarse ante cada estímulo del exterior o de otro objeto. Los demás objetos simplemente se limitan a indicarle al engranaje las fuerzas que le hacen, y ya sabrá el engranaje para dónde se ha de mover, y a qué otros objetos modificar.
- **Mensaje.** Es cada uno de los estímulos que se envían a un objeto.
- **Herencia.** Para facilitar la programación, se puede establecer toda una jerarquía de tipos o clases. P.ej., podemos declarar una clase Engranaje con las características básicas de los engranajes. De ella podemos derivar otras tres: Eng. fijo, Cremallera, y Eng. helicoidal. Cada una de estas clases especializa la clase general, con la ventaja de que las características comunes a los tres tipos de engranajes sólo hay que decirlas una vez.

El avance de la programación orientada a objetos ha llegado hasta los programas de gestión y que requieren el uso de bases de datos. El problema surge en el momento en que dos filosofías entran en conflicto: la filosofía orientada a objetos, y la de la base de datos que se pretende usar, fundamentalmente relacional. El conflicto principal es el problema de la impedancia, es decir, es difícil hacer encajar una programación orientada a objetos con las consultas y accesos propios de la base de datos, realizados en un lenguaje de manipulación y acceso a los datos, lenguaje que suele ser de otro tipo, normalmente no procedural. Asimismo, los datos retornados por la base de datos están en un

formato incomprensible para el lenguaje orientado a objetos, por lo que es necesario un paso de conversión que haga inteligibles esos datos.

Una solución factible a este problema consiste en hacer bases de datos cuyo sistema gestor tenga una interfaz orientada a objetos. Cuando hablamos de interfaz nos referimos a que tenga una capacidad tal que los programas sean capaces de interactuar con él según la filosofía orientada a objetos. Esta solución puede ser aproximada, a su vez, según varios métodos:

- Extender el modelo relacional. Consiste en añadir a una base de datos relacional la posibilidad de hacer cosas orientadas a objeto.
- Modelo de objetos persistentes. Consiste en declarar cierto tipo de objetos com persistentes. Un objeto es persistente si queremos que se guarde en la base de datos.
- Modelo integrado semántico. Añade también ciertas capacidades de consulta sin necesidad de programación externa.

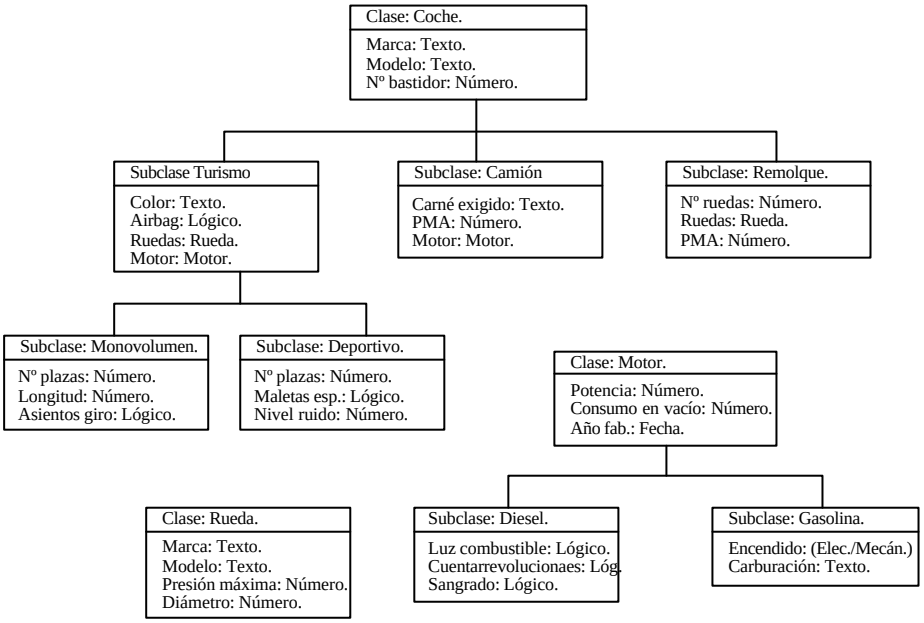


Figure 10. Esquema de clases para almacenar información sobre coches.

De estos métodos el más empleado actualmente es el de objetos persistentes, ya que es el que mejor se adecúa a la metodología de programación orientada a objetos.

El esquema de la figura ilustra la estructura que podría tener la clase Coche. De esta forma, cada objeto de tipo Coche que manejemos, será almacenado automáticamente en la base de datos si se declara como objeto persistente. Vemos que un objeto Coche puede ser, a su vez, un Turismo, un Camión o un Remolque; un Turismo puede ser Monovolumen o Deportivo. Dependiendo del lenguaje que se emplee, podremos tener objetos que sean simplemente Turismos sin necesidad de pertenecer a Monovolumen o a Deportivo, o sea, podemos tener tanto objetos de clases finales como de clases intermedias. Nótese que con esta metodología, vamos describiendo un objeto como integrado por otros más pequeños, llegando al nivel de refinamiento que la solución de nuestro problema requiera. Aquí indicamos que un Turismo o un Camión posee un Motor, y a continuación se describen las

características de un motor. Nótese que el campo Motor no se especifica en la clase Coche, ya que también consideramos que un Remolque es un Coche y éstos carecen de Motor.

Este tipo de esquemas define una jerarquía desde dos puntos de vista. Por un lado especifica un refinamiento en cuanto a conceptos: Un Deportivo es un refinamiento de un Turismo, que a su vez es un refinamiento de un Coche. Así, podemos decir que todo Turismo es un Coche, y que todo Deportivo es un Turismo, pero en ningún caso que todo Coche es un Deportivo, ya que hay casos de Coches, como p.ej. Remolques, que no son Deportivos, ni siquiera Turismos. Así, existe una jerarquía en base a la especialización o generalización (según se vea) de los objetos del problema. Hay casos, como el de las Ruedas, en los que no es necesaria especialización alguna.

La otra jerarquía es la jerarquía de composición. Cada objeto está compuesto de campos, que pueden ser, a su vez, otros objetos. P.ej., vemos que un Deportivo posee campos que indican sus características particulares: su número de plazas, si viene con maletas especialmente diseñadas a la forma del maletero, y el nivel de ruido del motor. Pero, además, por el hecho de ser un Turismo, también posee otra información, tal como el color y si dispone de airbag o no; por otro lado los campos Rueda y Motor, son, a su vez, objetos, cada uno de ellos con sus características propias. Esta jerarquía supone un refinamiento en cuanto a las características de cada objeto.

Estas dos jerarquías conjuntas dan una gran potencia a la programación orientada a objetos.

Desde el punto de vista de una base de datos, los datos se almacenan de una forma parecida al sistema relacional, de manera que existirá un tabla por cada clase o subclase de nuestro esquema. Quizás la única diferencia sustancial es que cada objeto tiene asignado automáticamente un número (OID- *Object Identifier*) que sirve para poder ser referenciado por los objetos de los que forma parte. El concepto de OID sustituye, en parte, al de clave en el sistema relacional.

Modelo declarativo.

El enfoque de las bases de datos declarativas es sumamente intuitivo para el usuario, y le permite abstraerse de los problemas de programación inherentes a otros métodos. Este modelo suele usarse para bases de conocimiento, que no son más que bases de datos con mecanismos de consulta en los que el trabajo de extracción de información a partir de los datos recae en realidad sobre el ordenador, en lugar de sobre el usuario. Estos mecanismos de consulta exigen que la información se halle distribuída de manera que haga eficiente las búsquedas de los datos, ya que normalmente las consultas de este tipo requieren acceder una y otra vez a los datos en busca de patrones que se adecúen a las características de los datos que ha solicitado el usuario. Sin embargo, no hablaremos de la organización de los datos, sino sólo de las formas de las consultas.

Antes de comenzar, aclararemos que, cuando se vea el lenguaje SQL sobre las bases de datos relacionales, diremos que este es un lenguaje no procedural, en el sentido de que el usuario especifica qué es lo que quiere, pero no cómo. No se debe confundir este aspecto del SQL con un lenguaje puramente declarativo, ya que éstos, amplían la filosofía de la base de datos, de manera que el usuario no es consciente de los métodos de búsqueda que se realizan internamente, y la forma en que se manejan los datos también es muy distinta; además, en el caso de las funcionales, es necesario complicar soberanamente los métodos utilizados si se quiere mantener la pureza de la metodología funcional. Además, la teoría que subyace en ambos modelos difiere radicalmente.

Entre las bases de datos declarativas podemos citar fudamentalmente dos: las deductivas, y las funcionales. Ambas extienden paradigmas o métodos de programación (al igual que ocurre con la programación orientada a objetos) a las bases de datos, de manera que ambos, programa y base de datos puedan cooperar más eficientemente en la resolución del problema.

Las bases de datos funcionales extienden el modelo de programación funcional, que se basa especialmente en el concepto de transparencia referencial. Este concepto viene a indicar que todo objeto computacional se debe comportar como una función, de manera que ante las misma entradas responde siempre con la misma salida. Este hecho, puede no ser cierto en otros paradigmas, especialmente el orientado a objetos, en el que la salida de un objeto no depende sólo de sus entradas, sino también del estado interno en el que se hallaba. Así, el modelo funcional elimina el concepto de estado.

Sin embargo, una base de datos, identifica precisamente el estado de los datos que la empresa necesita o posee en un momento determinado. Dado que bases de datos y estados tienen una relación bastante directa, es difícil hacerla encajar con el modelo funcional. Por ello, las dejaremos a un lado, y continuaremos con el siguiente modelo: las bases de datos deductivas.

Una base de datos deductiva puede ser considerada también como integrada por un conjunto de tablas. Sin embargo, nuestro punto de vista varía esencialmente. A veces es necesario ver una misma cosa (un problema, una situación, etc.) desde distintos puntos de vista, ya que ello ayuda a compararlo con distintas cosas que ya conocemos y permite adoptar soluciones que, de otra forma, serían difíciles de comprender. Algo así ocurre con las bases de datos deductivas. Aquí una tabla no se considera como un conjunto de tuplas, sino como un conjunto de hechos de un tipo concreto. De hecho, una base de datos deductiva, pretende deducir qué hechos son ciertos o no, y en qué circunstancias. Toda la base de datos gira en torno a esa filosofía.

Por ejemplo, si queremos tener información sobre el horario de trenes en la provincia de Málaga, podemos dar los siguientes hechos:

```
tren(Málaga, Fuengirola, 18:00, 1:00, Metro).
tren(Málaga, Bobadilla, 7:00, 1:30, Picasso).
tren(Bobadilla, Archidona, 11:15, 0:30, Antequerano).
tren(Bobadilla, Ronda, 12:00, 1:00, Rondeño).
tren(Ronda, Fuengirola, 13:45, 2:00, Ojalá).
```

El primer hecho nos indica que el tren llamado Metro sale de Málaga a las 18:00 y llega a Fuengirola 1:00 horas más tarde, y así sucesivamente. No complicaremos demasiado el problema incluyendo el concepto de paradas intermedias. También podemos tener información sobre qué poblaciones de Málaga son de interés turístico:

```
interés(Antequera).
interés(Málaga).
interés(Marbella).
interés(Ronda).
interés(Vélez-Málaga).
```

Esto que aquí se indica son los hechos de cuya veracidad, la máquina tiene una certeza absoluta. Según la hipótesis del mundo cerrado, el ordenador presupone que todo aquello que no es un hecho, o que se puede deducir, es falso. Esta hipótesis, un poco soberbia por parte del ordenador, simplifica mucho la lógica interna de los procesos de consulta.

Hasta ahora, hemos dado hechos de forma directa, al igual que en el modelo relacional se almacenaba información en las tablas. Sin embargo, la potencia de este método radica en que se puede dar información de manera indirecta. P.ej. para indicar que dos poblaciones están conectadas por tren, no es necesario especificarlas todas una por una, lo cual sería un verdadero tedio en el caso de tener varios miles de poblaciones en nuestra base de datos. Basta decir que dos poblaciones están conectadas si existe un tren que las une, o pueden unirse haciendo trasbordos, o sea, si hay alguna o algunas intermedias con la/s que ambas están conectadas. Esto se indica mediante una cláusula condicional, de la forma[1]:

```
conectadas(X, X).
conectadas(X, Y) :- conectadas(Y, X).
conectadas(X, Y) :- tren(X, Y, _, _, _).
conectadas(X, Y) :- conectadas(X, Z) and conectadas(Z, Y).
```

Estas cláusulas indican lo siguiente:

* Toda población está conectada consigo misma.
* Si una población X está conectada con otra Y, entonces se supone que Y también está conectada con X.
* Dos poblaciones X e Y están conectadas si hay un tren que las une directamente. El carácter '_' indica que nos da igual el valor de ese campo.
* Dos poblaciones X e Y están conectadas si hay una intermedia Z, a la que ambas están conectadas.

Estas cláusulas dan información indirecta sobre la base de datos. Nótese el uso de la palabra *si* en las explicaciones anteriores: en realidad cada cláusula define una condición para que se cumpla algo. La parte izquierda de la cláusula será verdad *si* se cumplen las condiciones de la derecha. Veamos ahora la facilidad con que pueden hacerse consultas.

La consulta más fácil es ver si un hecho es cierto o no. P.ej., ¿hay un tren entre Málaga y Bobadilla?:

```
:- tren(Málaga, Bobadilla, _, _, _).
```

a lo que la máquina responderá:

Sí.

En realidad, lo que preguntamos con la cláusula de consulta anterior, es si tren(Málaga, Bobadilla, _, _, _) se deduce de lo que la base de datos contiene, lo cual es efectívamente cierto.

Sin embargo, este sistema es aún más potente. Podemos preguntar qué trenes parten de Málaga:

```
:- tren(Málaga, _, _, _, Y).
```

a lo que se responderá con los valores que puede tomar Y para que el resultado se deduzca de lo que tiene la base de datos:

Y := Metro.

Y := Picasso.

Pero podemos hacer que el sistema «piense» un poco: es lo que se llama inferencia de valores.

[1] Las reglas que se dan obedecen a una expecificación algebraica, más que a unadescripción lógica comprensible computacionalmente. No obstante, se ha optado por dicha especificación a efectos aclaratorios sobre el cometido de las reglas.

Preguntemos si están conectadas Málaga y Ronda:

```
:- conectadas(Málaga, Ronda).
```

Sí.

El sistema ha deducido que sí, ya que según la regla 4ª existe una población intermedia, Bobadilla, en la que poder hacer trasbordo.

También podemos hacer preguntas más complejas, que tengan en consideración hechos de varios tipos. P.ej. ¿qué trenes parten de poblaciones con interés turístico?

```
:- tren(X, _, _, _, Y) and interés(X).
```

a lo que se responderá:

X := Málaga, Y := Metro.

X := Málaga, Y := Picasso.

X := Ronda, Y := Ojalá.

De esta forma, vemos que en una base de datos deductiva, la información se puede indicar como hechos, o como secuencia de deducción. Aunque a la hora de la verdad, la parte deductiva es mucho más compleja de lo aquí expuesta, suponen un avance interesante para las aplicaciones de Inteligencia Artificial.

No profundizaremos más en los métodos de resolución ni en la teoría que hay por debajo de este modelo, ya que escapa a los objetivos de la asignatura.