Bases de datos

Introducción

Las bases de datos son un elemento fundamental en el entorno informático hoy en día y tienen aplicación en la práctica totalidad de campos. Concebidas con un propósito general, son de utilidad para toda disciplina o área de aplicación en la que exista una necesidad de gestionar datos, tanto más cuanto más voluminosos sean estos. Además, el trabajo con un SIG presenta una serie de características (uso múltiple de los datos, necesidad de acceso eficiente para análisis, necesidad de indexación, etc.) que hacen que sea recomendable el uso de bases de datos.

Pese a que, como veremos en este mismo capítulo, el uso de las bases de datos en el ámbito SIG no ha sido siempre el actual, hoy en día representan una parte clave para la gestión de los datos geográficos, en especial dentro del marco de proyectos de cierta envergadura. Aunque la realidad es que todavía se efectúa mucho trabajo SIG sin emplear bases de datos (y las aplicaciones SIG así lo permiten, no siendo estrictamente necesario disponer de una base de datos para almacenar la información), la naturaleza propia de los proyectos SIG y la progresiva implantación de los SIG a niveles más allá del uso personal traen ambas consigo un uso cada vez mayor de las bases de datos, y por tanto una mayor necesidad de conocer el funcionamiento de estas.

Fundamentos de bases de datos

Aunque las particularidades de los datos espaciales con los que trabajamos en un SIG han hecho necesarias modificaciones y adaptaciones sobre el esquema de trabajo de las bases de datos genéricas, en esencia los fundamentos de estas siguen constituyendo el elemento primordial sobre el que la arquitectura de gestión de datos espaciales se apoya. En esta sección, veremos de forma introductoria esos fundamentos de bases de datos genéricas, aplicables a cualquier otro ámbito además del de los SIG, para posteriormente poder tratar el caso particular de los datos espaciales.

¿Qué es una base de datos?

Entendemos como *Base de Datos* un conjunto de datos estructurado y almacenado de forma sistemática con objeto de facilitar su posterior utilización. Una base de datos puede, por tanto, constituirse con cualquier tipo de datos, incluyendo los de tipo puramente espacial (geometrías, etc.) tales como los que se utilizan en un SIG, así como, por supuesto, datos numéricos y alfanuméricos como los que constituyen la componente temática de la información geoespacial. Los elementos clave de la base de datos son esa estructuración y sistematicidad, pues ambas son las responsables de las características que hacen de la base de datos un enfoque superior a la hora de gestionar datos.

Podemos ver más claramente las implicaciones de utilizar una base de datos si recurrimos al ejemplo que vimos en el primer capítulo de este libro, relativo a la gestión forestal de un territorio. Para ello, consideremos que el número de usuarios del SIG y de los datos asociados no se limita únicamente al gestor forestal que ha de tomar decisiones o establecer planes de actuación, sino a muchos otros profesionales que puedan ejercer su trabajo en ese mismo área o puedan emplear total o parcialmente esos mismos datos.

Imaginemos, por ejemplo, el caso de un ingeniero encargado de planear la instalación de un tendido eléctrico a través de nuestra zona forestal de ejemplo. Sin duda, deberá emplear datos tales como Modelos Digitales de Elevaciones, capas de zonas protegidas o capas de arbolado para establecer el trazado óptimo y estimar costes de la línea, entre otras tareas. Si en una situación ideal este ingeniero estaría en comunicación con el gestor forestal y ambos compartirían sus conocimientos dentro de un equipo multidisciplinar, también en lo referente a los datos debería existir una comunicación igual que implique, ente otras cosas, un uso compartido y

convenientemente coordinado de ellos. En otras palabras, los datos también tienen ese carácter multidisciplinar y deben dejar de verse como algo propio de un uso particular, para concebirse como un conjunto global del que se benefician diversos usuarios.

Establecer un uso compartido de los datos en una situación como la anterior no parece difícil, ya que simplemente se trata de dos profesionales que realizan tareas relacionadas y que, de un modo u otro, van a tener un contacto directo. El gestor forestal puede sencillamente dar una copia de sus datos al ingeniero y este podrá trabajar después con ellos de forma independiente. Aunque los datos con que trabajan son inicialmente los mismos, en realidad esta práctica da lugar a dos copias aisladas que constituyen dos universos distintos.

La situación real, sin embargo, es habitualmente mucho más compleja, y utilizar un esquema de colaboración como el anterior puede ser imposible, carecer por completo de sentido, o tener un buen número de consecuencias negativas. A medida que aumenta el número de usuarios, resulta menos recomendable que cada uno trabaje con sus propios datos y se los hagan llegar entre ellos a medida que los necesitan (una realidad que, desgraciadamente, se presenta con más frecuencia de lo recomendable). No debe olvidarse que un conjunto más amplio de usuarios que trabajan de esta forma y son ellos mismos quienes gestionan sus propios datos, implica directamente un número también más elevado de aplicaciones informáticas y de formatos de archivo, complicando enormemente el trabajo coordinado en cuanto el equipo tiene un tamaño medio.

Es probable además que existan usuarios dentro de una misma organización (por ejemplo, un organismo público) que aunque requieran para su trabajo datos similares, no tengan contacto alguno entre sí. Aunque los usuarios sean independientes, sus datos no lo han de ser necesariamente, y en una situación ideal deberían acudir a un repositorio único de datos del que cada cual tomaría lo necesario, en lugar de basar su trabajo en un conjunto de datos fragmentado y difícil de gestionar.

Pensemos en un dato que pueda ser de interés a varios usuarios, como por ejemplo una capa de vías de comunicación. A nuestro gestor forestal le será de interés para, por ejemplo, saber qué medios de acceso existen en caso de tener que hacer frente a un incendio. Lo más relevante de esas vías será su trazado, es decir su geometría, y tal vez el tipo de vía de que se trata, para poder conocer la velocidad a la que se pueden desplazar los medios de extinción. Otros usuarios, por su parte, pueden necesitar parámetros distintos como el volumen de tráfico medio de cada vía. Si todos ellos tienen una capa de vías con los parámetros asociados que necesitan para su trabajo, nos encontramos con una innecesaria redundancia de la componente espacial (las geometrías), y una dispersión de la componente temática, que resultaría más conveniente mantenerla agrupada.

Pensemos ahora que el gestor forestal detecta un error en el trazado de una de las vías y lo corrige. Esa corrección no estará disponible para los restantes usuarios, que pueden a su vez efectuar modificaciones similares que no redundarán en una mayor calidad de los datos con los que trabaja el gestor forestal, ya que, pese a utilizar datos similares, trabaja con su propio conjunto de datos. Incluso si en algún momento todos estos usuarios deciden poner en común sus datos y unirlos, esta operación puede ser muy compleja o incluso, como sucede frecuentemente, imposible de realizar. Por su parte, otros usuarios pueden añadir una nueva variable temática, como por ejemplo un índice de siniestralidad de la vía, el cual, si bien tal vez no resulte de utilidad inmediata para muchos usuarios, en un futuro sí pudiera serlo. Una vez más, estos nuevos datos no quedan a disposición del resto de usuarios, y en caso de serlo, no lo hacen en conjunto con datos similares, sino como un dato aislado de los restantes.

En definitiva, es complejo gestionar de forma adecuada los datos en el momento en que estos alcanzan un ámbito más allá de lo personal, y las prácticas más habituales basadas en una gestión «manual» de un conjunto de ficheros no son una opción adecuada. La solución para lograr esa necesaria gestión centralizada de los datos son las bases de datos y también, como veremos más adelante, los sistemas gestores de bases de datos, que representan la interfaz entre las bases de datos y los distintos usuarios.

¿Por qué interesa usar una base de datos?

En base al ejemplo anterior, podemos analizar algo más sistemáticamente las ventajas de una base de datos frente a una gestión no organizada de los datos. Algunas ventajas que afectan directamente a los datos son las siguientes:

- **Mayor independencia** . Los datos son independientes de las aplicaciones que los usan, así como de los usuarios.
- **Mayor disponibilidad** . Se facilita el acceso a los datos desde contextos, aplicaciones y medios distintos, haciéndolos útiles para un mayor número de usuarios.
- **Mayor seguridad (protección de los datos)** . Por ejemplo, resulta más fácil replicar una base de datos para mantener una copia de seguridad que hacerlo con un conjunto de ficheros almacenados de forma no estructurada. Además, al estar centralizado el acceso a los datos, existe una verdadera sincronización de todo el trabajo que se haya podido hacer sobre estos (modificaciones), con lo que esa copia de seguridad servirá a todos los usuarios.
- **Menor redundancia** . Un mismo dato no se encuentra almacenado en múltiples ficheros o con múltiples esquemas distintos, sino en una única instancia en la base de datos. Esto redunda en menor volumen de datos y mayor rapidez de acceso.
- **Mayor eficiencia en la captura, codificación y entrada de datos** .

Esto tiene una consecuencia directa sobre los resultados que se obtienen de la explotación de la base de datos, presentándose al respecto ventajas como, por ejemplo:

- **Mayor coherencia** . La mayor calidad de los datos que se deriva de su mejor gestión deriva en mayor calidad de los resultados.
- **Mayor eficiencia** . Facilitando el acceso a los datos y haciendo más sencilla su explotación, la obtención de resultados es más eficiente.
- **Mayor valor informativo** . Resulta más sencillo extraer la información que los datos contienen, ya que uno de los cometidos de la base de datos es aumentar el valor de estos como fuente de información.

Por último, los usuarios de la base de datos también obtienen ventajas al trabajar con estas, entre los que cabe citar:

- **Mayor facilidad y sencillez de acceso** . El usuario de la base de datos se debe preocupar únicamente de *usar* los datos, disponiendo para ello de las herramientas adecuadas y de una estructura solida sobre la que apoyarse.
- **Facilidad para reutilización de datos** . Esto es, facilidad para compartir.

De forma resumida, puede decirse que la principal bondad de una base de datos es la centralización que supone de todos los datos con los que se trabaja en un contexto determinado, con las consecuencias que ello tiene para una mejor gestión, acceso o estructuración de estos.

Modelos de bases de datos

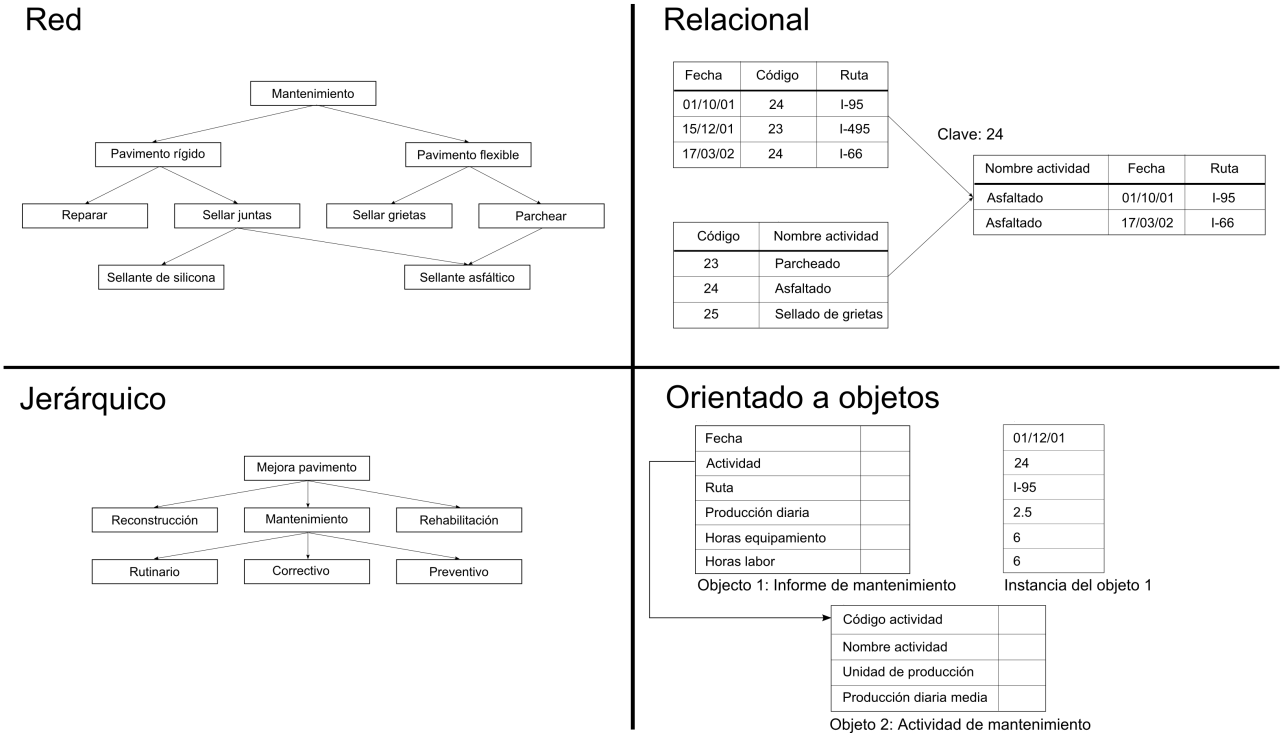
En función de la estructura utilizada para construir una base de datos, existen diversos modelos. El modelo de la base de datos define un paradigma de almacenamiento, estableciendo cómo se estructuran los datos y las relaciones entre estos. Las distintas operaciones sobre la base de datos (eliminación o sustitución de datos, lectura de datos, etc.) vienen condicionadas por esta estructura, y existen notables diferencias entre los principales modelos, cada uno de ellos con sus ventajas e inconvenientes particulares. Algunos de los más habituales son los siguientes:

- **Bases de datos jerárquicas** . Los datos se recogen mediante una estructura basada en nodos interconectados. Cada nodo tiene un único padre, y cero, uno o varios hijos. De este modo, se crea una estructura en forma de árbol invertido en el que todos sus nodos dependen en última instancia de uno denominado *raíz* . Aunque potente, el modelo jerárquico presenta algunas deficiencias, principalmente la escasa independencia de sus registros (el acceso a un registro —un nodo— implica que se ha de pasar por

sus padres, restando flexibilidad a la navegación por la base de datos). Otra grave deficiencia de este modelo es la mala gestión de la redundancia de datos, ya que si un registro guarda relación con dos o más, debe almacenarse varias veces, al no estar permitido que el nodo correspondiente tenga varios padres. Esto tiene consecuencias no solo en el mayor volumen de datos que se almacena, sino también en la integridad y coherencia de los datos. Si se modifica una de las «copias» de ese registro en la base de datos, deben modificarse también las restantes, ya que, aunque no conectadas en la estructura de la base de datos, realmente representan una única realidad y debieran ser idénticas entre sí.

- **Bases de datos en red** . Con objeto de solucionar los problemas de redundancia de las bases de datos jerárquicas, surge el modelo en red. Este modelo permite la aparición de ciclos en la estructura de la base de datos (es decir, no ha de existir un único padre para cada nodo), lo cual permite una mayor eficacia en lo que a la redundancia de datos se refiere. Presenta, no obstante, otros problemas, siendo el más importante de ellos su gran complejidad, lo que hace difícil la administración de la base de datos.
- **Bases de datos relacionales** . Constituyen el modelo de bases de datos más utilizado en la actualidad. Solucionan los problemas asociados a las bases de datos jerárquicas y en red, utilizando para ello un esquema basado en tablas, que resulta a la vez sencillo de comprender y fácil de utilizar para el análisis y la consulta de los datos. Las tablas contienen un número dado de *registros* (equivalentes a las filas en la tabla), así como *campos* (columnas), lo que da lugar a una correcta estructuración y un acceso eficiente.
- **Bases de datos orientadas a objetos** . Se trata de uno de los modelos más actuales, derivado directamente de los paradigmas de la programación orientada a objetos. El modelo extiende las capacidades de las bases de datos relacionales, de tal modo que estas pueden contener objetos, permitiendo así una integración más fácil con la propia arquitectura de los programas empleados para el manejo de la base de datos, en caso de que estos hayan sido desarrollados mediante programación orientada a objetos. Su popularidad crece de forma notable en ciertas áreas en las cuales resultan más ventajosas que el modelo relacional, siendo los SIG una de ellas.

La figura \ref{Fig:Comparacion_BBDD} muestra una comparación esquemática de los anteriores modelos de bases de datos.



Comparación entre algunos modelos de base de datos más frecuentes (adaptado de [USDT2001]). $$\label{Fig:Comparacion_BBDD}$$

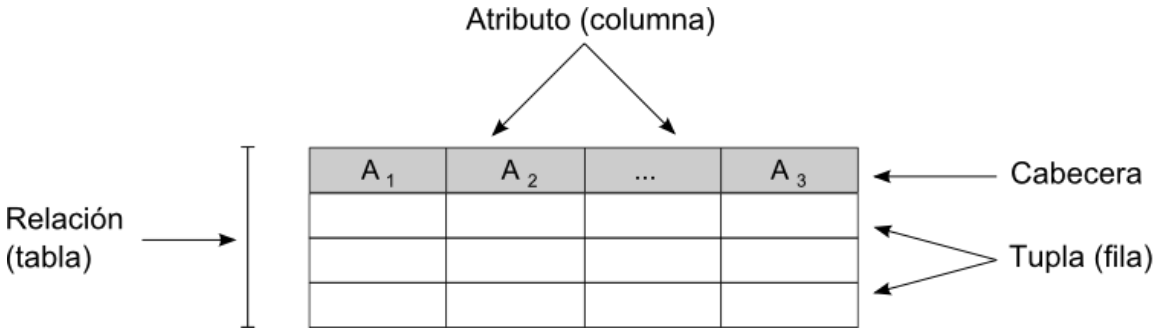
Bases de datos relacionales

Aunque, como ya hemos visto, existen diversos tipos de bases de datos, las más utilizadas con diferencia en la actualidad son las relacionales, que han demostrado su idoneidad en la mayor parte de situaciones. Estas son también las que encontraremos en el ámbito SIG, y resulta por ello necesario añadir algunas nociones adicionales sobre ellas para la correcta comprensión no solo de este capítulo, sino también de otros posteriores que desarrollan temas relacionados.

El modelo relacional fue desarrollado en 1969 por Ted Codd y publicado un año después en un artículo ya clásico [Codd1969ACM], y consiste básicamente en un conjunto de relaciones tabulares. Estas relaciones son tan importantes como los propios datos (las tablas, en este caso), y constituyen una idea central en el modelo relacional, de ahí su denominación. La característica principales que ha convertido a este modelo de base de datos en el más popular en la actualidad es su gran simplicidad, la cual indirectamente le dota de una gran potencia. Paralelamente, el modelo relacional se sustenta en unos fundamentos matemáticos sólidos y sus ideas pueden expresarse mediante conceptos de la teoría de conjuntos, lo que posibilita un análisis formal del mismo.

Además de las denominaciones habituales de *tabla* , *fila* y *columna* , existe una terminología específica empleada al referirse a las bases de datos relacionales. Así, en el modelo relacional los datos se organizan en tablas bidimensionales, cada una de ellas con información relativa a un determinada *entidad* . La tabla en sí se conoce como *relación* , ya que recoge la relación existente entre sus elementos, y constituye así el eje central del modelo relacional. Dentro de la tabla, los datos están organizados a su vez en filas y columnas. Las columnas representan los distintos *atributos* asociados a la entidad, mientras que las filas conforman los distintos *registros* . Una fila se forma con un conjunto de n atributos, constituyendo una *tupla* .

El *esquema de la relación* está formado por los nombres de los atributos y un *dominio* asociado a estos, que delimita el rango de valores posibles para cada atributo. El dominio especifica el tipo de dato a contener en cada columna. Por ejemplo, si se recoge un nombre el atributo será de tipo alfanumérico, mientras que si el atributo es un conteo deberá ser de tipo entero. Además de los tipos habituales (fechas, cadenas de texto, valores reales$. Puede emplearse también la denominación menos formal de *número decimal* o bien *valor de coma flotante* , esta última más común en el ámbito informático y referida a la forma de almacenamiento de este tipo de valores.}, valores enteros, etc.) pueden emplearse en ciertas bases de datos valores más complejos. Esto es de especial interés en el caso de los SIG, ya que permite utilizar geometrías como un tipo de datos más, con la utilidad que esto tiene a la hora de almacenar datos espaciales. El esquema de la relación se recoge en la primera fila de la tabla, conocida como *cabecera* . El número de filas de la tabla sin contar la cabecera (es decir, el número de tuplas) se conoce como *cardinalidad* .



Elementos del modelo relacional.
$$\label{Fig:ElementosModeloRelacional}$$

Las relaciones son, por tanto, un conjunto de tuplas asociadas a un esquema. En una relación, tanto el orden de las filas como el de las columnas son irrelevantes (exceptuando la cabecera, que no es un tupla como tal, sino que define el esquema como hemos visto), pero es importante que cada atributo sea del tipo correspondiente a la columna a la que pertenece. Es decir, que sea coherente con el esquema.

En la figura \ref{Fig:ElementosModeloRelacional} puede verse un esquema de los elementos fundamentales del modelo relacional.

Una forma abreviada de definir las relaciones que forman parte de una base de datos es mediante su nombre y su esquema expresado como una lista de los atributos que lo constituyen. Por ejemplo, podemos definir una relación denominada `PERSONAS` como

```
PERSONAS(DNI, Nombre, Altura, Edad, Ciudad)
```

Una base de datos contiene normalmente más de una tabla, ya que suelen ser muchos los tipos de datos a almacenar y resulta conveniente dividirlos en distintas tablas. Además de las relaciones que la tabla en sí implica, es necesario definir relaciones entre las distintas tablas, y para ello se emplean los denominados atributos *clave* . Un atributo clave es aquel que tiene valor único para cada tupla, pudiendo servir para representar a esta plenamente. Por ejemplo, en una tabla con nombres de personas e información adicional sobre ellas según el esquema anterior, los nombres no pueden ser la clave primaria, ya que puede haber dos personas con un mismo nombre. El número de su Documento Nacional de Identidad, sin embargo, sí que puede servir como atributo clave. Además de su unicidad, una clave debe ser invariable, identificando la misma tupla a lo largo del tiempo. Un esquema de relación puede contener varios atributos clave, que se conocen como *claves candidatas* . Normalmente, de estas se elige una como representante principal de las tuplas, y se conoce como *clave primaria*

{ Por convención, las claves se escriben subrayadas al definir el esquema de la tabla, de tal modo que el de la tabla \texttt{PERSONAS quedaría de la siguiente forma:

```
PERSONAS( DNI , Nombre, Altura, Edad, Ciudad)
```

Si no existe ningún atributo que cumpla los requisitos para ser utilizado como clave, este puede incorporarse al esquema de la relación, añadiendo por ejemplo un nuevo atributo con un código arbitrario. Un ejemplo de esto lo podemos encontrar en el cuadro \ref{Tabla:clavePrimaria}, donde se incorpora un atributo que hace la función de clave a una tabla con información sobre personas pero que no contiene el DNI de estas entre esa información y, por tanto, carece de un atributo adecuado para servir de clave.

En la definición de clave cabe también la presencia de claves compuestas, es decir, formadas por varios atributos cuya combinación es única para cada tupla. No obstante, la utilización de claves simples es preferible generalmente, ya que simplifica gran parte de las operaciones en las que la presencia de una clave es necesaria.

Cuando trabajamos con datos espaciales, es habitual emplear la componente espacial como clave, ya que esta suele ser única. En el caso de almacenar información sobre ciudades, con los nombres sucede de forma similar a lo visto para el caso de personas, ya que existen ciudades con el mismo nombre en distintos lugares. La localización de estas, sin embargo, es única, ya que no puede haber dos ciudades simultáneamente en el mismo lugar.

Tabla a

DNI	Nombre	Altura	Edad	Ciudad
50234561	Juan Gómez	1,85	35	Madrid
13254673	Edurne Montero	1,60	30	Toledo
46576290	Luis Urrutia	1,75	46	Madrid
38941882	Juan Gómez	1, 71	55	Valencia

Tabla b

ID	Nombre	Altura	Edad	Ciudad
001	Juan Gómez	1,85	35	Madrid
002	Edurne Montero	1,60	30	Toledo

ID	Nombre	Altura	Edad	Ciudad
003	Luis Urrutia	1,75	46	Madrid
004	Juan Gómez	1, 71	55	Valencia

Adición de un campo para crear una clave. La tabla a) contiene un atributo único (DNI). La tabla b) no contiene un atributo único entre sus datos, pero se añade el campo ID con un código arbitrario que puede ser empleado como clave. El nombre en este caso no sirve como atributo único, ya que hay dos personas en la tabla con el mismo nombre.
$$\label{Tabla:clavePrimaria}$$

El empleo de estas claves permite relacionar tablas entre sí, siempre que estas compartan algún atributo común. Por ejemplo, pensemos en una base de datos que contenga la tabla anterior y junto a esta otra tabla con el siguiente esquema:

`CIUDADES( Nombre , Población, Superficie)`

Es decir, la base de datos contiene información sobre personas y sobre ciudades.

Es sencillo ver que puede vincularse una tabla a la otra a través del atributo que contiene el nombre de la ciudad. Nótese que este atributo no tiene el mismo nombre en ambas tablas, y que, mientras que en una de ellas representa la clave primaria, en la otra no puede serlo pues existen nombres de ciudades repetidos. Pese a ello, este atributo nos permite establecer una relación entre las tablas, que podríamos denominar «nacido en». A cada tupla de la primera tabla, que representa a una persona dada, podemos vincularla con una de la segunda tabla, que representa una ciudad en particular, ya que toda persona ha nacido en una ciudad y gracias al atributo `CIUDAD` podemos saber exactamente cuál es dicha ciudad.

Las interrelaciones entre tablas pueden ser de distintos tipos en función del número de elementos distintos que se vinculan de cada tabla. En nuestra relación «vive en», una persona puede vivir en una única ciudad, mientras que una ciudad puede tener muchas personas viviendo en ella. Es decir, cada tupla de la tabla `PERSONAS` se relaciona con una única de la tabla `CIUDADES` , y cada tupla de esta última se relaciona con una o varias de la primera. Este tipo de relación se conoce como de *uno a muchos* .

Existen otros dos tipos de relaciones además de esta: las denominadas de *uno a uno* y las de *muchos a muchos* . Un ejemplo de relación de uno a uno podrían ser «casado con», que estableceríamos entre la tabla `PERSONAS` y ella misma (las dos tablas implicadas no han de ser necesariamente distintas). Cada persona puede estar casada únicamente con otra, por lo que la relación es de uno con uno, relacionándose una tupla con tan solo otra distinta, y no con varias.

Es importante reseñar que en algunas relaciones como «nacido en» todos los elementos de una o de las dos tablas se encuentran vinculados de algún modo a través de la relación, mientras que en otros no es así necesariamente. Así, todas las personas han nacido en alguna ciudad, y estarán relacionadas con la correspondiente tupla en la tabla `CIUDADES` , pero no todas las personas están necesariamente casadas.

Un ejemplo de relación *muchos a muchos* la podemos plantear si contamos en nuestra base de datos con, por ejemplo, una tabla con empresas, entre cuya información se incluya una lista de las ciudades en las que cada empresa tiene sede. Una empresa puede tener sedes en distintas ciudades, y una ciudad puede acoger a varias empresas, con lo que tanto ciudades como empresas pueden estar vinculadas a más de una tupla en la otra tabla.

Sistemas gestores de bases de datos

Junto con las bases de datos, el elemento fundamental para el aprovechamiento de estas son los *Sistemas Gestores de Bases de Datos* (SGDB o DBMS, del inglés *DataBase Management System*). Estos sistemas representan un elemento intermedio entre los propios datos y los programas que van a hacer uso de ellos,

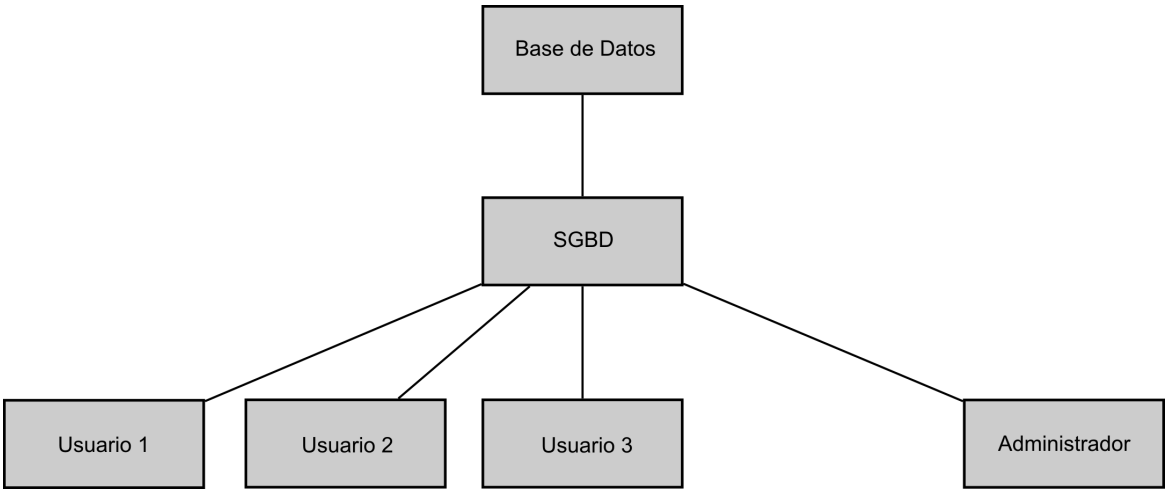
facilitando las operaciones a realizar sobre aquellos. En nuestro caso, son el componente que permite unir el SIG con la base de datos en la que se almacenan los datos espaciales con los que este va a trabajar.

Un SGBD es una pieza de software compleja, ya que las situaciones a las que debe responder son diversas y en muchas ocasiones con requerimientos elevados, por ejemplo en lo que a eficiencia y volumen de datos respecta. Piénsese que una base de datos actual puede tener millones de registros y ser utilizada simultáneamente por miles de usuarios, que a su vez pueden utilizar diversos programas, no todos ellos del mismo tipo. Por ejemplo, una base de datos que contenga números de teléfono, nombres de usuarios, direcciones y coordenadas asociadas a cada línea telefónica, puede ser empleada desde un SIG para crear un mapa que muestre la densidad de usuarios o también desde una aplicación que genere un listín telefónico, o bien desde una aplicación en una página Web que permita localizar el número de teléfono de una persona concreta. Cada una de estas aplicaciones realiza un trabajo distinto, pero todas ellas utilizan la misma base de datos. El SGBD debe proporcionar a todos ellos la metodología adecuada para extraer del conjunto de datos completo cuanto sea necesario en cada caso.

Además, el SGBD es la herramienta utilizada no solo por quienes aprovechan los datos, sino también por aquellos que se han de encargar de la propia gestión y mantenimiento de la base de datos. Administrar una base de datos puede suponer una tarea altamente compleja, por lo que el SGBD debe proveer los útiles necesarios para llevar a cabo ese mantenimiento.

Para ser de verdadera utilidad y responder a todas las necesidades que pueden plantearse en relación con la base de datos, un SGBD debe perseguir los siguientes objetivos:

- **Acceso transparente a los datos** . La base de datos ha de poder accederse de forma transparente, sin que sea necesario para el usuario del SGBD preocuparse por aspectos internos relativos a la estructura de esta u otras características. Esto significa que, por ejemplo, si queremos recuperar un registro de la base de datos, debemos poder hacerlo sin necesidad de saber si dicha base de datos está almacenada en un único archivo o varios, o si el registro que pretendemos recuperar está almacenado a su vez de uno u otro modo. Así, el SGBD debe crear una abstracción de los datos que haga el trabajo con estos más sencillo, ocultando aspectos que no sean relevantes para dicho trabajo. Procedimientos como las consultas que veremos en el capítulo Consultas se realizan a través del SGBD, que es quien se encarga de interpretar dichas consultas, aplicarlas sobre la base de datos y devolver el resultado correspondiente. El SIG no accede a los datos, sino que se comunica con el SGBD y deja en manos de este el proceso de consulta en sí.
- **Protección de los datos** . Si la base de datos almacena información sensible, el SGBD debe controlar el acceso a esta, restringiendo el acceso cuando corresponda (por ejemplo, estableciendo distintos permisos de acceso para distintos tipos de usuarios) e implementando los mecanismos de protección necesarios.
- **Eficiencia** . Acceder a los datos no es suficiente en la mayoría de los casos, sino que se requiere un acceso eficiente. El SGBD debe ser capaz de gestionar de forma fluida grandes volúmenes de datos o de operaciones (por ejemplo, muchos usuarios accediendo simultáneamente), de modo que dé una respuesta rápida a las peticiones de los usuarios de la base de datos.
- **Gestión de transacciones** . Las operaciones sobre la base de datos tales como la adición o borrado de un registro se realizan mediante transacciones. Una transacción es un conjunto de operaciones realizadas por un usuario sobre la base de datos como una única unidad de trabajo, de forma indivisible. El SGBD ha de encargarse de gestionarlas de manera eficiente y segura para que todos los usuarios de la base de datos puedan hacer su trabajo de forma transparente. Aspectos como el acceso concurrente a la base de datos (varias transacciones simultaneas) resultan especialmente importantes, y en su buena gestión se pone gran esfuerzo en el diseño de los SGBD. Se denomina *transaccional* al SGBD capaz de garantizar la integridad de los datos, no permitiendo que las transacciones puedan quedar en un estado intermedio. Esto implica la capacidad de poder volver a un estado anterior en caso de que por cualquier causa (error en el sistema, fallo eléctrico, etc) no haya podido completarse la transacción.



Representación esquemática del papel de un Sistema Gestor de Base de Datos.
$$\label{Fig:SGBD}$$

La figura \ref{Fig:SGBD} esquematiza el papel que el SGBD juega en el manejo y empleo de los datos. Tanto los distintos usuarios (en el caso de nuestro supuesto de gestión forestal pueden ser desde el gestor forestal al cartógrafo encargado de actualizar los limites de las unidades inventariables) como el administrador de la base de datos acceden a esta a través del SGBD. No existe acceso directo a la base de datos.

El SGBD tendrá unas u otras características en función del modelo de base de datos subyacente, ya que debe adaptarse a las características de este para ofrecer las funcionalidades correspondientes en el nivel de usuario.

Diseño y creación de una base de datos

Una vez se toma la decisión de emplear una base de datos, el siguiente paso es el diseño y creación de esta. El diseño implica la definición de la estructura que va a tener la base de datos, que se deberá realizar teniendo en cuenta principalmente el tipo de datos que van a almacenarse y el modelo de base de datos elegido. El diseño debe adecuarse al uso previsto de la base de datos, de tal modo que acomode los datos de la mejor forma posible para cumplir los objetivos enunciados anteriormente en este mismo capítulo. Para ello debe conocerse la naturaleza de los datos que van a almacenarse (no necesariamente datos de los que se dispone en el momento de la creación, sino los que se espera pasen a formar parte de la base de datos a lo largo de su ciclo de vida), así como la de los algoritmos y procesos que van a emplearse sobre ellos.

Posteriormente al diseño, debe procederse a la implementación de la base de datos, esto es, a la creación propiamente dicha, incorporando los datos según los esquemas escogidos en la fase de diseño. Por último, y una vez creada la base de datos, debe procurarse un mantenimiento para que esté continuamente en condiciones de ser utilizada.

Más concretamente, pueden distinguirse las siguientes fases en el proceso global de desarrollo de una base de datos:

- **Diseño lógico** . Independiente del SGBD empleado, es un diseño conceptual que pretende modelizar el contenido de la base de datos.
- **Diseño físico** . Es la adaptación del diseño conceptual a las particularidades del SGBD escogido.
- **Implementación** . Introducción de los datos en la base de datos.
- **Mantenimiento** . Monitorización de la actividad sobre la base de datos.

La primera fase en el diseño de una base de datos implica un análisis de los datos que se van a recoger. Como resultado de ese análisis debe surgir un modelo conceptual que exprese la estructura de la información, siendo dicha estructura susceptible de ser empleada como esquema de partida para la base de datos en cuestión. El modelo conceptual ha de definir básicamente los tipos de datos a tratar y las relaciones existentes entre ellos,

elementos que serán luego expresados en términos del modelo de base de datos elegido (relacional, orientado a objetos, etc.) una vez se pase a la fase de diseño físico.

El modelo conceptual debe estructurar la información de forma que el usuario de la base de datos comprenda de forma sencilla el contenido y forma de esta. Por tanto, debe desarrollarse teniendo presentes las necesidades de los usuarios y el hecho de que estos no necesariamente han de ser especialistas en bases de datos, sino especialistas en los propios datos en sí. Por otra parte, el modelo debe intentar capturar del mejor modo posible la realidad que se pretende modelizar, por lo que el conjunto de tipos de datos y relaciones debe elaborarse de modo similar a dicha realidad para recoger toda la complejidad del sistema. Y, por supuesto, el modelo debe poder ser implementado posteriormente y utilizado en conjunto con el SGBD escogido, ya que de otro modo no presenta utilidad práctica.

Existen diversas metodologías para desarrollar un modelo conceptual. Una de las más extendidas por su sencillez y potencia es la del *modelo entidad--relación* (abreviadamente, modelo E-R).

Denominamos *entidad* a un objeto o concepto del mundo real acerca del cual se recoge información, y que puede diferenciarse de otros objetos, incluso si son de su misma clase (un ordenador, por ejemplo, es un objeto, y puede diferenciarse de otros ordenadores, incluso si son de idénticas características, ya que no son todos el mismo objeto y ese en particular tendrá alguna propiedad distinta, como puede ser el número de serie). La entidad puede tener sentido físico o bien ser una idea abstracta, como un tipo de deporte, una clase de música o una palabra.

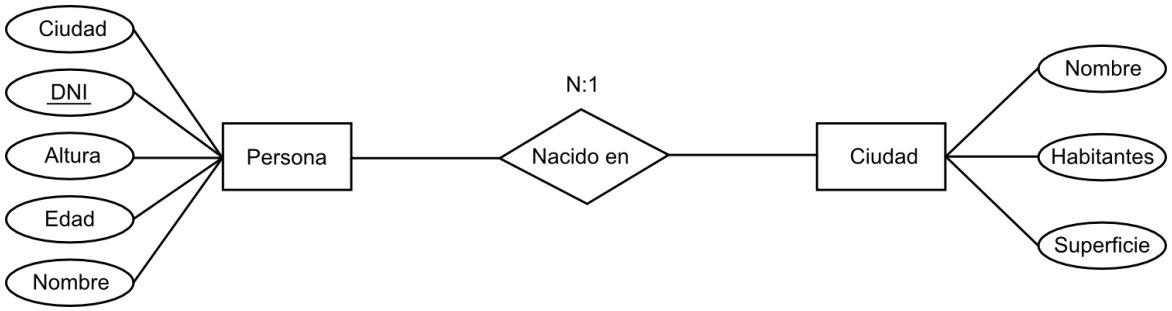
Una entidad se describe mediante una serie de características o atributos, que son las que definen su naturaleza y sus propiedades. Una colección de entidades es un conjunto de entidades distintas (que representan a objetos distintos), las cuales comparten unos atributos comunes. Por ejemplo, un conjunto de ordenadores de los cuales se conocen los atributos *modelo* , *marca* y *procesador* .

Por su parte, una *relación* expresa la dependencia existente entre entidades y permite la asociación de estas. No resulta difícil ver que estos conceptos —entidad, atributos y relación— guardan un notable paralelismo con las ideas del modelo relacional que ya conocemos. Así, y aunque no resulte por completo inmediato, es sencillo traducir un modelo entidad-relación (conceptual) a un modelo relacional, que constituye ya un modelo aplicado a un tipo particular de base de datos. Por ello, el modelo E-R es una herramienta potente para el diseño lógico de la base de datos, especialmente si esta utiliza el modelo relacional.

Para desarrollar el diseño conceptual de una base de datos siguiendo el modelo E-R, estos son los pasos principales:

- Partimos de una descripción textual del problema o sistema que queremos recoger. Esta descripción contiene los requisitos necesarios y ha de formular la pregunta a la que queremos que la base de datos dé respuesta. Para nuestro ejemplo con datos sobre personas y ciudades, el problema podríamos formularlo como «¿qué personas han nacido en cada ciudad?».
- Se toman los verbos y los sustantivos de la descripción textual. Los sustantivos son posibles entidades o atributos, mientras que los verbos son posibles relaciones. En nuestro caso, «persona» y «ciudad» serán entidades y «nacido en» una relación.
- Se analizan las frases y determina la cardinalidad de las relaciones y otros detalles.

Simbología empleada en el modelo entidad--relación.
$$\label{Fig:SimbolosER}$$



Ejemplo de diagrama E-R.
$$\label{Fig:EjemploDiagramaER}$$

El modelo así creado se expresa mediante un diagrama en el que las entidades se representan como cajas rectangulares, las relaciones mediante rombos y los atributos en círculos o elipses, todos ellos con sus correspondientes nombres en el interior. Cuando un atributo es un identificador, se representa con su nombre subrayado, del mismo modo que en la definición de esquemas que ya vimos anteriormente (Figura \ref{Fig:SimbolosER}).

Si el número de atributos es elevado o el diagrama es complejo por existir gran cantidad de tablas e interrelaciones, pueden omitirse los atributos para una mayor legibilidad, describiéndose en un documento adicional.

Como ejemplo de lo anterior, la información sobre personas y ciudades que venimos manejando, así como la relación «nacido en» existente entre ambas, se expresarían según el modelo entidad-relación con un diagrama tal como el mostrado en la figura \ref{Fig:EjemploDiagramaER}.

El modelo E-R presenta algunas limitaciones semánticas, y no es suficiente para expresar con detalle la estructura de algunos tipos de información. Por esta razón, surge el conocido como *modelo E-R extendido* , que amplía el modelo E-R añadiendo nuevos elementos. Con su mayor potencia, el modelo E-R extendido acerca el diseño conceptual a los conceptos de la programación orientada a objetos, incorporando por ejemplo mecanismos de herencia. El enfoque orientado a objetos recoge no solo la estructura del sistema de información, sino también su comportamiento dinámico.

DNI	Nombre	Altura	Edad	Ciudad	Población	Superficie
50234561	Juan Gómez	1,85	35	Madrid	6386932	607
13254673	Edurne Montero	1,60	30	Toledo	80810	232
46576290	Luis Urrutia	1,75	46	Madrid	6386932	607
38941882	Juan Gomez	1, 71	55	Valencia	1564145	134

La información de las tablas `PERSONAS` y `CIUDADES` puede recogerse en una única tabla
$$\label{Tabla:TablaConjunta}$$

Tras el diseño lógico, el diseño físico de la base de datos ha de llevar el modelo conceptual a la práctica y crear un repositorio de datos que pueda ser usado por el SGBD. Debe, asimismo, mantener todas aquellas propiedades del modelo conceptual, de modo que el contenido de la base de datos siga expresando de forma fiel la realidad y su estructura continúe siendo fácil de comprender para los usuarios. Si, siguiendo el enfoque más habitual, optamos por crear una base de datos según el modelo relacional, esto implica la creación de las correspondientes relaciones y los esquemas asociados a cada una de ellas.

La tablas que definamos en la base de datos pueden tener consecuencias directas sobre el uso de esta, afectando a aspectos como el rendimiento de las operaciones que posteriormente se lleven a cabo o al volumen de datos total necesario. Por ejemplo, nuestra base de datos con dos tablas, `PERSONAS` y `CIUDADES` , puede implementarse utilizando únicamente una tabla como la mostrada en el cuadro \ref{Tabla:TablaConjunta}. Esta tabla contiene la misma información que las dos tablas anteriores, y en principio permite realizar operaciones

similares. Si quisiéramos saber la población de la ciudad donde ha nacido una persona en concreto, podríamos hacerlo de igual modo con independencia de cuál de las estructuras mostradas tenga la base de datos. En un caso deberemos acudir a dos tablas y una interrelación entre ellas, mientras que en el otro solo es necesario emplear una tabla, la única que por otra parte contiene nuestra base de datos.

Aunque la funcionalidad sea la misma, el uso de una única tabla tiene efectos poco deseados que se advierten rápidamente, como por ejemplo la redundancia de datos. La población y superficie de Madrid aparecen repetidos en dos ocasiones, y aparecerían más veces si hubiera en la tabla `PERSONAS` más tuplas correspondientes a individuos nacidos en esta ciudad. De igual modo sucedería con otras ciudades. En el esquema basado en dos tablas, sin embargo, estos datos aparecen en una única ocasión y no dependen del número de personas de cada ciudad. En una base de datos de pequeñas dimensiones como la que utilizamos de ejemplo, esta circunstancia puede parecer poco relevante, pero si trabajamos con millones de registros en la tabla `PERSONAS` la diferencia es realmente importante.

Otro aspecto a tener en cuenta en el diseño físico de la tabla es elegir nombres adecuados para los atributos y las tablas. Los nombres deben ser inequívocos y dar una idea clara de la información que contienen, y un usuario debe poder identificar sin dificultades qué tablas y atributos son aquellos a los que debe acudir para efectuar una consulta y dónde se encuentra la información que busca. El atributo `CIUDAD` en la tabla `PERSONAS`, por ejemplo, cumple sin problemas su papel a la hora de establecer la relación entre esta tabla y la que recoge los datos de las distintas ciudades, pero si buscamos exclusivamente información sobre las personas, no es completamente preciso, ya que no aclara si se trata de la ciudad en la que una persona ha nacido o en la que habita.

Siempre que pueda existir alguna duda razonable a la hora de interpretar el contenido de una tabla, debe intentarse solventar esta mediante el uso de nombres claros y concisos. Establecer una sistemática a la hora de nombrar atributos y respetarla a lo largo de todo el conjunto de tablas de una base de datos hará más fácil para los usuarios la comprensión de esta. Por ejemplo, es habitual emplear el prefijo *num* cuando un atributo representa un conteo (y por tanto, su tipo de dato será de tipo entero). Siguiendo esta convención, si quisiéramos añadir un campo a la tabla `PERSONAS` con el número de hermanos de cada individuo, sería más conveniente y más informativo denominar al atributo correspondiente `numHermanos`, en lugar de, por ejemplo, `Hermanos`. Más que seguir unas u otras normas para nombrar atributos y tablas, lo importante es ser consistente y tratar siempre de utilizar nombres que informen y no den lugar a confusiones.

Una vez que se establece un diseño y se implementa en la base de datos, lo normal es que este sea relativamente estable y no varíe a lo largo del tiempo. Las relaciones, por su parte, sí se modifican frecuentemente, ya sea añadiendo tuplas a medida que se incorporan nuevos datos o modificando las ya existentes. No obstante, los SGBD ofrecen también funcionalidades para modificar la estructura de la base de datos, incorporando nuevas tablas o cambiando el esquema de alguna de ellas. Estas funcionalidades no suelen ser accesibles para los usuarios con carácter general, sino pensadas para el mantenimiento de la base de datos por parte de su administrador.

Bases de datos espaciales

Todo cuanto hemos visto en los puntos anteriores constituye el conjunto de ideas fundamentales sobre las que se asienta la creación y uso de bases de datos de cualquier índole. No obstante, no hemos mencionado a lo largo de los ejemplos presentados ningún dato de carácter espacial, a pesar de que sabemos bien que la información geográfica contiene tanto una componente temática como una espacial. Más aún, algunos de los atributos en los sencillos casos mostrados, como puede ser el atributo `CIUDAD`, son fácilmente asociables a elementos geográficos (por ejemplo, un punto que señale el centro de la ciudad o un polígono que recoja su contorno).

Aunque las ideas anteriores no pierden su validez al incorporar datos espaciales, la inclusión de estos no es en absoluto obvia, y presenta una complejidad adicional que requiere de nuevos planteamientos para poder seguir trabajando con la base de datos de una forma similar a como sucede cuando se trabaja con los tipos de datos habituales. Mantener las características propias del SGBD en el contexto de los datos espaciales no es sencillo,

y tampoco lo es integrar esa base de datos dentro de un SIG y permitir que este aproveche la potencia de dicha base de datos de la mejor manera posible.

En lugar de adentrarnos en la complejidad de las bases de datos espaciales (aunque en el capítulo Consultas veremos bastante más en lo que a las operaciones y posibilidades de estas respecta), veremos las distintas etapas que podemos encontrar a lo largo de la historia de los SIG en lo referente a su integración con bases de datos, para de este modo comprender los diversas soluciones que han ido apareciendo.

Evolución del uso de bases de datos en los SIG

Como acabamos de decir, los conceptos que hemos visto en las anteriores secciones representan una gran parte de la realidad actual en cuanto al manejo de datos (espaciales o no) dentro de un SIG. No obstante, el problema del acceso a los datos se ha solucionado de diversas formas a lo largo de la historia de los SIG, y encontramos en las aplicaciones SIG distintos enfoques a lo largo del tiempo. Para concluir este capítulo veremos con algo más de detalle la evolución que ha seguido esta importante faceta de los SIG.

Primera generación. Ficheros

Los primeros programas, entre los cuales se han de incluir los primeros SIG, se caracterizaban en lo que al almacenamiento de datos respecta por una ausencia completa de cualquier tipo de almacenamiento estructurado. En estas aplicaciones, los datos no se veían como un elemento más dentro de un sistema, sino como una parte del propio *software* o, al menos, como algo asociado únicamente a un producto particular. Así, encontramos en esta época como práctica habitual el uso de ficheros con formatos cerrados, pensados para ser leídos y escritos casi de forma exclusiva por la aplicación particular que ha de consumirlos, limitando así el uso compartido y el alcance de los datos a otros ámbitos distintos.

Integrar en el SIG otros datos distintos a aquellos para los que la aplicación se había diseñado no era sencillo, ya que existía una vinculación muy directa entre software y datos. Asimismo, las funcionalidades del software eran también específicas para esos datos, y todas ellas se implementaban directamente en la aplicación. Al no existir un SGBD que se encargara de gestionar las operaciones, era el propio SIG quien debía ser responsable de las funcionalidades de acceso o edición. Otras funcionalidades típicas de un SGBD, sin embargo, no aparecían en estos primeros SIG, ya que no eran necesarias. Por ejemplo, velar por la integridad de los datos en operaciones concurrentes de varios usuarios no era necesario si la aplicación en sí no estaba diseñada para permitir este acceso múltiple.

Las únicas ventajas que pueden encontrarse en este enfoque son las relacionadas con el rendimiento, que podía en ciertos casos ser mayor que el esperable en caso de utilizar un SGBD para canalizar el trabajo con los datos. Esto es así debido a que la propia especificidad de la aplicación permitía una optimización «a medida», aunque todo ello a cambio de sacrificar la flexibilidad de la aplicación, su escalabilidad, o la posibilidad de que los datos empleados pudieran ser utilizados de forma sencilla para alimentar otras aplicaciones.

Segunda generación. Bases de datos relacionales

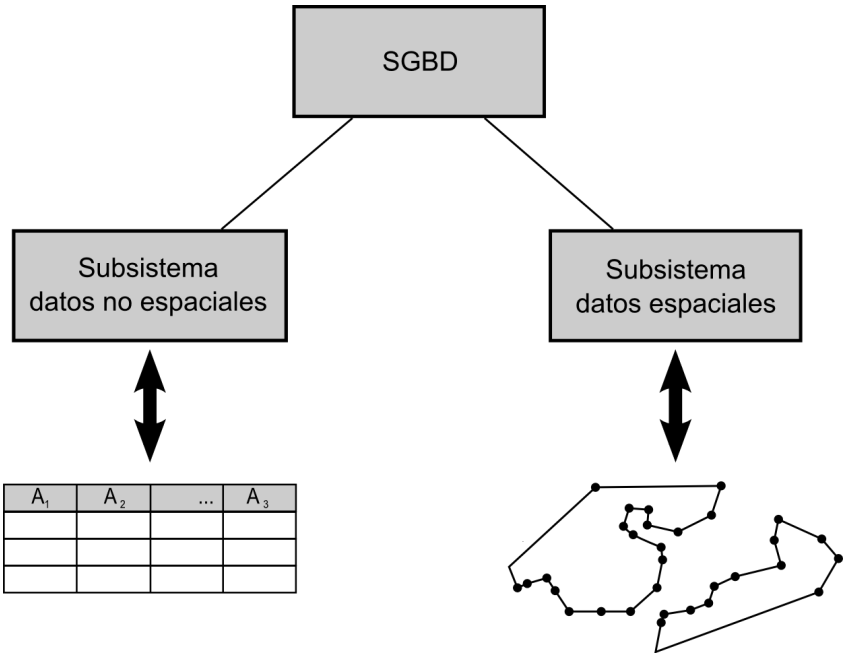
Una vez que las bases de datos comienzan a tomar su papel en el panorama del software, no tardan en encontrar su camino dentro de las aplicaciones SIG. Las bases de datos relacionales, que como ya sabemos son las más empleadas, comienzan a ser utilizadas también para gestionar los datos espaciales con los que se trabaja en un SIG. A partir de esta segunda generación, se empiezan a adaptar las características del modelo relacional y de las bases de datos que lo implementan a las particularidades de los datos espaciales. Las dificultades que aparecen debido a la inherente complejidad de la componente espacial hacen que surjan diversas alternativas para su manejo. Las más reseñables de entre ellas son el uso de una arquitectura dual en la que únicamente la componente temática se gestiona mediante una base de datos y el uso de una arquitectura en capas en el que se da un pleno almacenamiento de la información espacial en la base de datos.

Arquitectura dual

El primer intento de incorporar las bases de datos lo encontramos en el uso de una arquitectura dual en la cual el SGBD se hace cargo únicamente de la componente temática de los datos. Puesto que la dificultad estriba en el manejo de la componente espacial, esta no se incorpora por el momento a la base de datos, que trabajará únicamente con los datos temáticos. Esto permite el uso de sistemas gestores de bases de datos estándar, sin adaptación alguna, ya que estos se encuentran perfectamente preparados para el manejo de esos datos no espaciales, y no requieren elementos adicionales para trabajar sobre ellos.

La componente espacial, por su parte, es gestionada por el propio SIG, en el que se implementan las funcionalidades necesarias. Al igual que sucedía anteriormente con los SIG de primera generación, no todas las funcionalidades de un SGBD han de aparecer necesariamente, ya que el sistema encargado de permitir el trabajo con los datos no es como tal un SGBD. La única diferencia reside en que en este caso esta circunstancia afecta tan solo a la componente espacial de los datos, mientras que la componente temática queda en manos de un verdadero SGBD.

Existen, por tanto, dos subsistemas encargados de la gestión de los datos, cada uno de los cuales se encarga de un tipo de información (Figura \ref{Fig:ArquitecturaDual}). Esta arquitectura en la que datos espaciales y datos no espaciales se encuentran separados tiene ciertas ventajas, puesto que permite reutilizar información ya existente de uno u otro tipo. Por ejemplo, ficheros procedentes de aplicaciones CAD pueden incorporarse en el SIG aunque carezcan de una componente temática, aprovechando, no obstante la información espacial. El resultado de este uso es en su mayoría de tipo gráfico, pero un SIG que presente una arquitectura dual puede trabajar con él y gestionarlo gracias al subsistema encargado de la información espacial, suponiendo ya una mejora respecto al enfoque de los SIG de primera generación.



Arquitectura dual con subsistemas distintos para el manejo de datos espaciales y no espaciales.
$$\label{Fig:ArquitecturaDual}$$

La división entre datos espaciales y no espaciales conlleva, no obstante, una serie de inconvenientes. Por un lado, resulta difícil integrar operaciones en las que se empleen ambas componentes de los datos, que requerirán llamadas a ambos subsistemas y la posterior combinación de sus respuestas. Toda esta labor debe implementarse en el SIG, siendo este un proceso costoso que complica el desarrollo. Si todo el manejo de datos recayera sobre la base de datos, estas operaciones se realizarían de forma transparente, ya que bastaría ejecutar la operación en el SGBD y este se encargaría de realizar las tareas pertinentes y devolver después al SIG la respuesta. Se evitaría asimismo la redundancia en el propio software, ya que al emplear dos subsistemas han de duplicarse una buena parte de funcionalidades, una de ellas en el SGBD externo y otra en el propio SIG.

Aunque una parte importante del SIG descansa ya sobre un SGBD, otra sigue presentando muchas de las deficiencias que caracterizaban a la primera generación. Mientras que la componente temática disfruta de las ventajas de usar un SGBD, la componente espacial no goza aún de esas ventajas, y existe una cierta descompensación que limita las posibilidades y hace más complejo el desarrollo del sistema.

Arquitectura en capas

La otra forma de aprovechar una base de datos relacional para su uso dentro de un SIG consiste en incorporar toda la información dentro de la base de datos, incluyendo la de corte espacial, buscando la manera más adecuada de llevar esto a cabo pese a las limitaciones que la propia base de datos presenta en este caso. Asumiendo que una base de datos relacional en su concepto tradicional no esta diseñada para contener objetos complejos tales como geometrías o imágenes, y que, especialmente, el SGBD correspondiente no presenta las mismas funcionalidades y la misma potencia en el manejo de este tipo de datos que en el de tipos de dato estándar (valores numéricos, cadenas de texto, fechas, etc.), es posible, sin embargo, plantear soluciones que permitan llevar toda la información de un SIG a una base de datos y poder gestionarla por completo a través de un SGBD, con las ventajas que ello conlleva, y que ya conocemos.

Dos son las alternativas existentes: un almacenamiento *transparente* y un almacenamiento *opaco* . Ambos se distinguen en la forma de almacenar la información y también las operaciones sobre los datos, que vienen condicionadas por la estrategia empleada para el almacenamiento de estos.

En el almacenamiento transparente se emplean los propios tipos de datos del SGBD, y las operaciones se implementan en el lenguaje de consulta de este. Es decir, se intenta implementar toda la funcionalidad deseada empleando los elementos básicos del SGBD de la misma forma que haríamos si los datos a almacenar no fueran de tipo espacial. La componente espacial de los datos se almacena empleando tuplas, variando según la implementación la manera en que esto se lleva a cabo. Una geometría como tal no se ajusta a ningún tipo básico de datos, pero en realidad esa geometría no es sino un conjunto de coordenadas que definen una serie de puntos, y dichas coordenadas sí que son un tipo básico susceptible de almacenarse en un SGBD común.

En el almacenamiento opaco se emplean objetos binarios para almacenar la información y las operaciones se implementan externamente en la herramienta SIG. Al no utilizar los tipos de datos del SGBD, tampoco pueden emplearse las operaciones de consulta de este, y es necesario implementar los algoritmos correspondientes en el SIG.

La ventaja más directa de utilizar una arquitectura en capas, ya sea mediante un almacenamiento transparente o uno opaco, es la facilidad para reutilizar un SGBD existente. Con poco esfuerzo pueden incorporarse los datos espaciales a un SGBD estándar, existiendo en la actualidad numerosas alternativas sobradamente probadas y con una amplia gama de funcionalidades. Esta es la opción más empleada hoy en día en los SIG, principalmente por esa sencillez, que permite una conexión sin muchas dificultades de una aplicación SIG con la mayoría de los SGBD de uso habitual fuera del ámbito SIG.

Existen, aun así, inconvenientes y aspectos mejorables, achacables a la nula especialización de los SGBD para el manejo de información espacial. En el caso del almacenamiento opaco, no poder emplear el lenguaje de consulta del SGBD constituye un grave inconveniente. Por su parte, en el almacenamiento transparente sí que puede emplearse, pero no todas las operaciones necesarias para el trabajo con datos espaciales pueden implementarse con un lenguaje de consulta no adaptado a las particularidades de los datos espaciales, por lo que la funcionalidad es limitada.

Asimismo, la eficacia es menor, ya que en un caso los algoritmos son externos al SGBD y en el otro las consultas suelen ser complejas y operan sobre un elevado número de tuplas, necesario para recoger la información espacial.

Tercera generación. Bases de datos extensibles

En la actualidad, las bases de datos presentan arquitecturas extensibles que permiten adaptarse a la naturaleza de los datos con los que trabajan, de tal forma que enfocan sus funcionalidades hacia la tipología particular que se maneje. Los tipos de datos clásicos conviven con nuevos tipos de datos que pueden ser definidos, y con operaciones específicas para estos.

Un caso particular de estas bases de datos extensibles son las *bases de datos orientadas a objetos* , que ya fueron comentadas al presentar los distintos modelos de bases de datos. A pesar de que este tipo de bases de datos no ocupan una porción significativa en el mercado global de las bases de datos, y son las de tipo relacional las más extendidas, existen algunos sectores en los que han logrado una mayor penetración, entre ellos el del SIG. Por sus características, las bases de datos orientadas a objetos resultan ventajosas para el manejo de datos complejos que no puedan recogerse con facilidad utilizando los tipos de datos clásicos de una base de datos relacional. En este grupo pueden incluirse las primitivas geométricas que utilizamos en un SIG para recoger la componente espacial de un dato espacial, las cuales resulta más adecuado considerar como objetos de un tipo dado (punto, línea o polígono), aprovechando así las ventajas que un enfoque orientado a objetos proporciona.

La principal ventaja de una base de datos orientada a objetos es su mayor eficiencia en el acceso a datos, lo que se traduce en consultas más rápidas en comparación con una base de datos relacional (veremos más sobre consultas en bases de datos espaciales en el capítulo Consultas). Por el contrario, carece de la base matemática de esta, por lo que el soporte para esas consultas es menos robusto. Para saber más sobre bases de datos orientadas a objetos, puede consultarse [Marques2002BBDD].

Los SGBD actuales presentan en su gran mayoría extensiones dedicadas al manejo de datos espaciales, los cuales contienen todo lo necesario para el manejo óptimo de estos, la realización de ciertas operaciones fundamentales y la optimización de las consultas y operaciones. Esta optimización es posible ya que el tipo de datos espacial está plenamente integrado en la base de datos y es considerado de la misma manera que cualquiera de los tipos de datos estándar como puede ser una cadena de texto o un valor numérico. La eficiencia que se obtiene de este modo es muy elevada.

Resumen

En este capítulo hemos visto los conceptos básicos sobre bases de datos. Una base de datos constituye un sistema que permite un manejo adecuado de los datos, garantizando la seguridad e integridad de estos y permitiendo el acceso a distintos usuarios de forma transparente. La base de datos está formada por los datos en sí, organizados de forma estructurada, mientras que las operaciones las provee el sistema gestor de base de datos (SGBD).

Existen diversos modelos para el almacenamiento de datos, siendo el modelo relacional el más habitual en la actualidad. En el modelo relacional la información se organiza en tablas relacionadas entre sí. Cada fila de una base de datos conforma una tupla, que contiene la información correspondiente a una entidad dada.

El diseño de la base de datos es de gran importancia, y conlleva el diseño de un modelo conceptual, el diseño de un modelo físico, la implementación y el mantenimiento. Herramientas como los diagramas E-R son de ayuda en las fases de diseño, cuyo principal objetivo es crear una estructura de la base de datos que facilite la interpretación de la información contenida y permita el máximo aprovechamiento de esta.

En lo que a los SIG respecta, las bases de datos se han ido incorporando paulatinamente a la gestión de los datos espaciales. Partiendo de una situación inicial en la que no se empleaban sistemas gestores de bases de datos, estos han ido integrándose en los SIG de diversas formas. En la actualidad, se emplean bases de datos relacionales adaptadas para poder almacenar datos espaciales y realizar operaciones sobre ellos. Los SGBD extensibles representan la ultima tendencia, y en ellos puede integrarse plenamente la información geográfica de forma óptima.
Portada