Informatización de organizaciones con Microsoft Access ®

Colección Textos Universitarios

Informatización de organizaciones con Microsoft Access ®

· Ernesto Ponsot Balaguer

PUBLICACIONES
VICERRECTORADO ACADÉMICO
CODEPRE

COLECCIÓN
Textos Universitarios

Publicaciones
Vicerrectorado
Académico

Informatización de organizaciones con Microsoft Access®
Primera edición, 2008

- *Concepto de colección y diseño de portada*
  Kataliñ Alava
- *Corrección*
  Luis Paniagua
- *Diseño y diagramación*
  Freddy Parra Cepeda
  Lilyan Carolina Matos P.
- *Impresión*
  Centro Editorial Litorama C.A.

HECHO EL DEPÓSITO DE LEY
Depósito Legal: LF23720086101418
ISBN: 978-980-11-1155-9

Universidad de Los Andes
Av. 3 Independencia
Edificio Central del Rectorado
Mérida, Venezuela
publicacionesva@ula.ve
http://viceacademico.ula.ve/
publicacionesva

- Los trabajos publicados en la Colección Textos Universitarios han sido rigurosamente seleccionados y arbitrados por especialistas en las diferentes disciplinas.

Impreso en Venezuela
Printed in Venezuela

Dedicado a *Morella Briceño Avila*, mi amada esposa,
sin cuyo aliento y comprensión esta obra
sencillamente no hubiese sido posible

Introducción

He querido presentar este material producto de varios años de estudio y dedicación, como un aporte práctico y metodológico al tema de la informatización de organizaciones y desarrollo de sistemas de información de pequeña y mediana escala, con enfoque de bases de datos relacionales.

Mucho del contenido aquí expuesto es producto de la reflexión surgida año a año a partir del estudio de diversos autores y el ejercicio docente de la materia, combinados con la experiencia extraída del ejercicio gerencial al frente de las direcciones de Organización y Sistemas, y Servicios de Información Administrativa de la Universidad de Los Andes, tanto coordinando proyectos informáticos como programando e implementando aplicaciones concretas.

Espero que sea especialmente útil a mis estudiantes de la Licenciatura en Estadística de la Universidad de Los Andes, sin cuyo esfuerzo permanente y crítica inteligente no habría valido la pena llevar al papel estas líneas.

Escogí un producto en particular para las exposiciones computacionales del texto: *Microsoft Access*. En él están implementados todos los ejemplos y con su código *Visual Basic for Applications* se elaboraron todos los programas. Debo confesar que la selección no me costó mucho esfuerzo. Suelo sentirme atraído fácilmente hacia el *software* que me reta y exige dedicación sostenida. Así ocurrió cuando vi por primera vez la versión 97 del producto y la he seguido hasta el mo-

mento con cierta fidelidad. No obstante, no es azarosa ni arbitraria la escogencia. *Access* debe ser el producto de bases de datos relacionales que cuenta con mayor capacidad instalada en el mundo, se integra a la suite de aplicaciones más popular en la actualidad, *Microsoft Office,* y ofrece prestaciones que dejarían pasmado a cualquier experto, por menos de 50 dólares americanos. Aunque se trata de un sistema gestor de bases de datos en toda la extensión de la frase, hay que reconocer que no es muy hábil tratando con enormes volúmenes de información ni con ambientes de red muy exigentes, así que si, como en efecto, aceptamos estas limitaciones, quede claro que hemos de circunscribirnos a sistemas de pequeña y mediana escala, aun cuando los conceptos y la práctica que abordaremos sean válidas en ambientes de desarrollo de exigencia superior.

Este texto pretende seguir la secuencia lógica que imprime la enseñanza del tema a estudiantes que lo enfrentan por primera vez. Afortunadamente, es también la secuencia lógica en que habría de acometerse un proyecto de informatización de organizaciones orientado hacia la creación de bases de datos y, sobre éstas, sistemas de información.

He tratado de mantener el material en un nivel introductorio sin trivializar las explicaciones. He procurado tocar los temas con la rigurosidad debida, presuponiendo que el lector tiene cierto dominio de la programación de computadores y obviando tópicos básicos (como las estructuras de la programación, tipos elementales de datos y similares) que se suponen conocidos y que, por lo demás, escapan al nivel introductorio que pretendo en esta obra. Para facilitar la lectura y comprensión de los contenidos, los ejemplos dados a lo largo del texto cubren exhaustivamente todas las etapas de la informatización organizacional.

Pretendo además motivar especialmente a los "profesionales del azar" para que utilicen la herramienta en toda su extensión y sin complejos. El mundo que les espera será muy diferente de aquel que enfrentaron sus antecesores. En el futuro no habrá que salir a buscar datos a la calle, éstos estarán a disposición en medios computacionales y, de no estarlo, será tarea prioritaria llevarlos allí. La idea es hacerlo de la mejor manera posible y obtener el máximo provecho medido en términos de información.

Agradezco a la Universidad de Los Andes la oportunidad que me ha brindado para producir este material al otorgarme un año sabático con este propósito.

Acompañan en un CD-ROM al texto, los apéndices a los que se hace referencia, la base de datos "MisLibros.mdb" con todos los programas que fueron necesarios y también el sitio Web del Sistema de Registro de Libros, material todo que he utilizado intensivamente para discutir y aplicar los contenidos expuestos.

capítulo 1

Los sistemas
Panorama general de sistemas

El mundo que nos rodea está repleto de sistemas. De hecho, él mismo es un sistema. Cada uno de nosotros inmerso en él, también lo es. Nuestra familia, la sociedad, la universidad, son sistemas sin duda. Ahora bien, formalmente ¿qué es un sistema?, ¿por qué y para qué hablamos de sistemas?, son interrogantes que no siempre tienen la debida respuesta, previo inicio de una discusión que utilizará el término copiosamente. Iniciamos pues el texto con este capítulo dedicado a los sistemas, cuyo objetivo es mostrar, sin pretender mucha rigurosidad, respuestas concretas a estas preguntas en el entendido de que no siempre están suficientemente claras la acepción e implicaciones del término.

La idea de sistemas ha rondado la humanidad desde tiempos inmemoriales. Ya Aristóteles en la Grecia antigua, sin acuñar precisamente el vocablo, afirmaba: "El todo es más que la suma de sus partes". Podría pensarse que en aquel momento cumbre de la reflexión humana, en el cual se produjeron muchas de las ideas que conforman la esencia misma de la civilización occidental, esa sencilla frase pasaría fácilmente desapercibida. Y así fue por varios siglos, cuando menos en lo formal. No obstante, allí comienza la preocupación filosófica y práctica por los sistemas, aun sin bautizar como tales.

Esta afirmación de Aristóteles desnuda un importante argumento: La simple delimitación, identificación, separación, estudio de sus partes y la posterior unión de ellas, como si se tratara de un rompecabe-

zas, no puede reconstruir el todo, no conduce a su comprensión. Faltan ingredientes básicos, pocas veces evidentes, como las interrelaciones de sus componentes entre sí y con el entorno.

Si entendemos el todo del que habló Aristóteles como cualquier objeto sobre el que se tenga especial interés, digamos el cuerpo humano, la familia, la empresa, el clima y una larguísima lista, entenderemos que su planteamiento, a primera vista de gran abstracción, resulta por el contrario muy concreto y, genialmente, de gran utilidad práctica para el pensamiento moderno.

Bertalanffy (1), precursor de la Teoría General de Sistemas, escribe en el prefacio a la edición revisada de su célebre libro:

> Están ingresando en la esfera del pensamiento científico entidades de naturaleza esencialmente nueva. En sus diversas disciplinas –ya fueran la química, la biología, la psicología o las ciencias sociales–, la ciencia clásica procuraba aislar los elementos del universo observado –compuestos químicos, enzimas, células, sensaciones elementales, individuos en libre competencia y tantas cosas más-, con la esperanza de que volviéndolos a juntar, conceptual o experimentalmente, resultaría el sistema o totalidad –célula, mente, sociedad–, y sería inteligible. Ahora hemos aprendido que para comprender no se requieren sólo los elementos sino las relaciones entre ellos –digamos, la interacción enzimática en una célula, el juego de muchos procesos mentales conscientes e inconscientes, la estructura y dinámica de los sistemas sociales, etc.

Sorprende la lucidez y visión de futuro en estas palabras, cuanto más porque fueron escritas hace ya casi 30 años, y esto, en materia de sistemas, es una era. Comparada la evolución del estudio de sistemas con la de la ciencia o el método científico, la cuestión está en pañales, pero es evidente que está teniendo y tendrá tal impacto en la sociedad como el que ha producido la ciencia. De hecho, la idea de sistemas y la práctica de sistemas, junto con, a la par de y complementando la ciencia en su sentido clásico, representan en la actualidad el marco conceptual que orienta la evolución del pensamiento y el desarrollo de nuestra civilización.

Mucho se ha escrito y se escribirá sobre el concepto de sistemas. Como suele suceder, no hay consenso absoluto entre los autores acerca

de su significado (no podría haberlo en realidad), sin embargo, y en vista que no es el propósito de este texto dilucidar tal asunto, mucho menos agotar la discusión, sino colocar el tema en perspectiva, adoptaremos una definición operativa y ampliamente utilizada: *Un sistema es una unidad o entidad con un propósito en alguna medida claro y fronteras suficientemente definidas, que se constituye gracias a la interrelación e interacción entre sus partes componentes.*

Resulta interesante a la luz de la definición, que son innumerables los conjuntos a nuestro alrededor que unitariamente considerados clasifican como sistemas: el Universo, la Vía Láctea, el Sistema Solar, el Planeta Tierra, el ecosistema, la humanidad, nuestro país, la sociedad, nuestra ciudad, nuestra familia o también el Estado, el sistema legal, el sistema de salud, la universidad, la facultad, el departamento, además, una empresa, una fábrica, la casa que habitamos, el vehículo en que nos movilizamos, la máquina que fabrica tornillos, el computador, nosotros mismos, nuestro sistema circulatorio o respiratorio, cada una de nuestras células, amén de una infinidad que dejamos fuera de la lista.

Sin embargo, el hecho de que prácticamente cualquier cosa puede ser entendida o vista como un sistema, aunque resulte extremadamente útil en el dominio de lo teórico, luce muchas veces escabroso en la práctica. Puede llegar a distraer hasta tal punto la atención de quien indaga, que representa un verdadero reto "descubrir", en el contexto del problema planteado, cuál es realmente el sistema.

Pues bien, un sistema, desde la perspectiva que nos ocupa, es aquel que nos interesa primordialmente. Debe ser considerado desde un punto de vista orientado hacia la solución del problema planteado, y delimitadas sus fronteras de manera que su comprensión contribuya al asunto que indagamos. Esto es importante puesto que en todos los sistemas hay tal variedad y complejidad, que resulta difícil centrar la atención de manera precisa. Para colmo de males, esta complejidad se acrecienta cuanto mayor es el grado de intervención humana en el asunto.

Afortunadamente, no hay imposibles en esta materia. Lo único que necesitamos es un poco de sentido común, y en lo que corresponde al problema de información (punto central aquí), una aproximación práctica y sencilla será de mucha ayuda.

Propongamos, como ejemplo, estudiar una biblioteca casera. Puede ser nuestra propia biblioteca integrada principalmente por libros (y no revistas o periódicos), seguramente pequeña, pero precisa en con-

tenidos de nuestro particular interés. ¿Es tal biblioteca un sistema como lo hemos definido? Antes de contestar, ahondemos un poco:

1. ¿Desde qué punto de vista? Si fuera desde el punto de vista físico o químico nos interesarían las partículas que componen el papel de los libros, las moléculas que forman la madera de la estantería, sus propiedades de elasticidad, durabilidad, resistencia, las fuerzas que intervienen para soportarla, su peso y cosas por el estilo. Si fuera desde el punto de vista histórico o socio-antropológico nos interesarían fechas importantes en su evolución, qué hombres o qué civilización la crearon y mantuvieron o actualizaron, cuáles eran los temas de interés de esos hombres y, por ende, qué visión del mundo tenían inducida a partir de su contenido, entre otras cuestiones. Cualquiera de estos puntos de vista es válido, pero cada uno de ellos nos conduce a plantearnos la biblioteca de manera diferente.

   Supongamos que en realidad nos interesa la biblioteca desde el punto de vista del control de sus libros. El problema que da origen a nuestro interés por ella se deriva de que ha ido creciendo, hemos venido incorporando nuevos textos y ha llegado el momento en que cuando necesitamos hacer una consulta, nos cuesta mucho trabajo ubicar el libro preciso, no contamos con un esquema de clasificación que rápidamente nos permita capturarlo y esto nos preocupa. La biblioteca se nos ha ido de las manos. La situación es tal, que hemos llegado a preguntarnos: ¿para qué tantos libros si cuando los necesito no los encuentro?, ¿habrá valido la pena adquirirlos si cuando preciso de ellos no están disponibles?, ¿cómo puedo saber si tengo o no libros del tema que ahora me ocupa? En otras palabras, miraremos la biblioteca desde el punto de vista de la información general que nos permita registrar y ubicar oportunamente un ejemplar de la colección. En consecuencia nos interesan, a grandes rasgos, cada libro, sus autores, el tema que trata, el espacio que ocupa o su ubicación en la estantería, las estanterías disponibles, en qué lugar de la casa se encuentran, etc.
2. Decidido el punto de vista, ¿se pueden enumerar y describir los aspectos que le son inherentes como un todo y aquellos que no? La respuesta a esta pregunta es mucho más fácil en el primer sentido que en el segundo, es decir, resulta inoficioso ponerse a

elucubrar sobre todos los aspectos que no le son inherentes. Mucho más productivo es pensar en aquellos que sí le tocan, aun cuando por descarte vendrán a la mente muchos que no. Esto no quiere decir que la pregunta sea inútil en su segunda parte, por el contrario, seguramente encontraremos aspectos cuya pertinencia no se establece claramente en las primeras de cambio. El método es entonces ir refinando sucesivamente nuestras consideraciones.

Volviendo a la pregunta, la respuesta parece ser afirmativa: establecido el punto de vista como en 1 en una primera aproximación -que deberá ser revisada repetidas veces-, podríamos considerar formando parte de lo que hemos dado en llamar "la biblioteca", los siguientes aspectos: sobre los libros, la adquisición, el registro, su clasificación, codificación o identificación, su ubicación en la estantería, luego, la recuperación para consulta, su préstamo, su destrucción o desaparición. Sobre la estantería, su ubicación en la casa, su capacidad, su clasificación, codificación o identificación; sobre las personas usuarias, su identificación, dirección y otras formas de ubicación. En síntesis, parece que decidimos considerar los libros, las estanterías y las personas que la utilizan. Entonces, por descarte, queda fuera todo lo demás.

Resulta trivial señalar el color de la casa en donde se encuentra la biblioteca o el modelo del carro en que trajimos el libro, como aspectos que no son inherentes. Mucho más útil será decidir sobre aquellos más estrechamente vinculados con nuestro punto de vista que dejaremos fuera. Así, por ejemplo, el texto completo de los libros podría tener mucho que ver con la biblioteca, pero desde el punto de vista de su registro y control, ¿tendrá sentido preocuparnos por el contenido de cada libro en toda su extensión? Podría ser que sí, pero si así fuera estaríamos hablando no de la biblioteca como la hemos conceptuado, sino del conocimiento contenido en ella a partir de cada libro exhaustivamente considerado. La decisión es por consiguiente dejar fuera el contenido en extenso de cada libro, excepto cuando sea útil para clasificarlo. Veamos otra consideración interesante: no resulta obvio dejar fuera de la biblioteca los aspectos relativos a quien nos facilitó o vendió el libro, pero ¿queremos recabar información sobre el

proveedor? Una vez más la respuesta perfectamente podría ser afirmativa, pero decidimos que no nos interesa.

3. ¿Es posible identificar las interrelaciones entre sus componentes? Establecimos en 2 que le son inherentes los libros, las estanterías y los usuarios. También enumeramos aspectos más específicos en cada caso que podrían interesarnos, ahora bien, ¿cómo se asocian estos actores? La respuesta a esta segunda interrogante resuelve ambas cosas, pero una vez más tiene el carácter subjetivo impreso por quien responde: los usuarios se rozan con los libros al adquirirlos, clasificarlos, requerirlos para consulta; los estantes y libros en cuanto a que los contienen; los usuarios y estantes en cuanto a que para poder obtener los libros deben conocer su ubicación, etc. En fin, podríamos poblar mucho más nuestro razonamiento, pero valga por ahora decir que efectivamente identificamos al menos algunas, si no todas, las interrelaciones que nos interesan entre sus componentes.

Retomando la pregunta original, la respuesta es entonces afirmativa: la biblioteca es un sistema. La consideración 1 nos lleva a descubrir el propósito de este sistema y ayuda a encontrar sus fronteras. La consideración 2 refuerza la comprensión de las fronteras y apoya el descubrimiento de las partes componentes del sistema. La consideración 3 conduce a comprender la interacción entre los componentes principales del sistema.

Todo sistema lo es propiamente cuando representa para quien lo define un objeto de interés, un "problema", en el sentido positivo del término, que lo mueve a indagar y profundizar en consideraciones reales y razonamientos lógicos. De aquí se deriva una cuestión muy importante: un sistema, a menos que haya sido expresamente diseñado y construido por el hombre, no existe en la realidad física como tal, existe en la mente del observador en tanto que este *le aprecie como tal,* y esto significa que una misma situación puede derivar en sistemas muy diferentes bajo la lupa de distintos observadores, incluso bajo miradas de un mismo observador desde distintos ángulos. También significa que un sistema se materializa como un "modelo" de la situación real cuando es descrito por el observador, quien le imprime su punto de vista (lo que en el contexto de la Teoría General de Sistemas se conoce como *weltanschauung* (2), palabra de origen

alemán que se refiere al modo de ver las cosas, de conceptuar el mundo que nos rodea[1]).

Por otra parte, todo objeto o situación real digna de ser indagada, es compleja en mayor o menor grado. Justamente por esta razón, ya no tanto desde el punto de vista filosófico sino más bien práctico, el enfoque de sistemas resulta tan útil. El hombre, desde que está sobre el planeta, al enfrentar la complejidad ha utilizado una táctica que generalmente da buenos resultados: "divide y vencerás". Esta tesis, aplicada apropiadamente en el plano de la investigación y la ciencia, posibilita enfrentar problemas y resolverlos exitosamente subdividiéndolos en partes más pequeñas, operando sobre detalles ya no tan complejos y luego, sin olvidar las interacciones que daban características unitarias al problema original, acometer las acciones hacia su solución ahora con mayor comprensión de sus características. En términos del enfoque de sistemas se trata de descubrir (o determinar) el sistema, los subsistemas que lo componen y el suprasistema que lo contiene (figura 1).

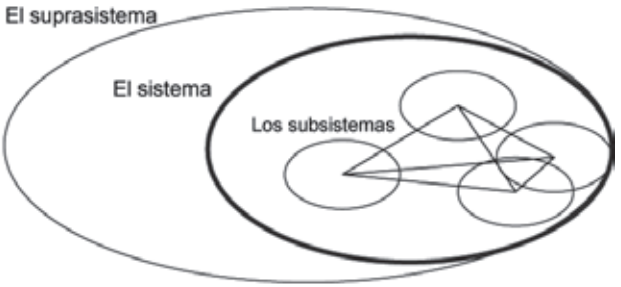


Figura 1. Esquema recursivo de los sistemas

La figura 1 muestra el esquema "recursivo" de los sistemas. Dado que todo sistema está inmerso en un suprasistema que lo contiene, es relevante determinarlo aunque no necesite describirse completamente sino sólo en términos de la relación de dependencia entre ambos. Por otro lado, en todo sistema pueden descubrirse o determinarse subsis-

[1] Gratificante resulta la lectura del libro "Pensamiento de sistemas, práctica de sistemas", de Peter Checkland, al que alude la referencia, en especial la revisión histórica que el autor expone en la primera parte, en la que sintetiza cómo surge la idea de sistemas y su relación con el pensamiento científico.

temas de complejidad inferior que sí necesitan ser descritos completamente, tanto ellos mismos como su relación con aquél.

La idea de recursividad la tomamos prestada de la computación. Un objeto recursivo es aquel que para su definición hace uso de sí mismo. En la programación de computadores, una función es recursiva si para que complete su misión debe invocarse a sí misma (por lo general, un número no preestablecido de veces).

El asunto cobra importancia cuando caemos en cuenta de que al ser el suprasistema, el sistema y los subsistemas, sistemas al fin, les aplica la misma definición dada originalmente y por ende pueden ser estudiados con las mismas herramientas conceptuales, sin importar qué tan complejos sean los unos comparados con los otros. También es consecuencia de esta propiedad que sea posible e incluso necesario materializar la estrategia "divide y vencerás".

En nuestro ejemplo de la biblioteca hemos definido en una primera aproximación muy general, tanto el sistema como los subsistemas. Falta escudriñar el suprasistema que le contiene. Nuestra biblioteca se encuentra inmersa en un lugar físico, por ejemplo, la casa. El hecho de que radique allí, como ya mencionamos, no parece interesante, excepto que al encontrarse en nuestra casa forma parte de nuestro hogar. La palabra "hogar", de significado mucho más amplio que "casa", comprende también a la familia. Así pues, podemos llegar a la conclusión de que nuestra biblioteca está contenida en un suprasistema más amplio que, para los propósitos señalados, decidimos será nuestro hogar, entendido como el recinto físico y las personas que lo habitan formando una familia.

Si nos interesara por ejemplo el sistema de facturación de una empresa, seguramente modelaríamos a la empresa misma como el suprasistema. Si nos interesara el departamento en que se desarrolla nuestra carrera universitaria como sistema, la Facultad podría definirse como el suprasistema que le contiene y así en todos los casos.

1.1 Algunas propiedades útiles de los sistemas

En la práctica de sistemas es conveniente conocer la existencia de dos importantes propiedades que auxilian al investigador en su tarea: el *isomorfismo* y la *emergencia.*

La primera de ellas, el *isomorfismo*, se refiere, como sugiere su nombre, a la "equivalencia en la forma" (*iso*, igual; *morfo*, forma). En una buena proporción de los casos, sistemas que aparecen ante el observador muy diferentes resultan teniendo formas equivalentes o, mejor aún, estructuras equivalentes. La utilidad de este hecho y la conciencia de su existencia ofrece al que indaga una ventaja al enfrentar el estudio de sistemas: no importa qué tan complejo parezca un problema particular de sistemas, generalmente será posible compararlo con otro que, sin ser igual a aquél, le es isomorfo, con la salvedad de que éste ya ha sido conceptuado o bien resuelto y, por tanto, buenas ideas podrán ser tomadas de su solución y aplicadas equivalentemente al que le ocupa. Se habla de isomorfismos y no equivalencias, pues la cuestión de la igualdad no se establece en toda la extensión del sistema, sólo en su estructura o forma. Se sugiere entonces la búsqueda de isomorfismos siempre que se enfrente un nuevo reto de sistemas.

Al comenzar nuestro ejemplo hemos podido haber hecho consideraciones como "Tal vez haya literatura que describa algo parecido a una biblioteca", o bien en algún otro momento nos hemos rozado con un problema que, aunque esencialmente distinto, nos luce parecido en su estructura. Por allí se empieza. En nuestro caso podríamos haber reflexionado que una biblioteca, tal como la hemos definido, es en realidad un inventario. ¡Sí, un almacén de artículos que se apilan para protegernos de la escasez! Sólo que aquí no son estanterías de cualquier cosa, sino de libros, y los libros no se consumen (y en este caso, no los pensamos vender, aunque podríamos). La escasez, problema por vencer en el inventario, es aquí escasez de material o contenido académico. No se trata de un inventario que ofrece sus bienes a cualquier cliente, aquí el cliente es uno mismo, que acude a la biblioteca para encontrar un libro. Entonces, tal vez ahora veamos el asunto con mayor claridad, sobre todo si antes hemos enfrentado un problema de inventarios y contamos con el isomorfismo entre aquél problema y el nuevo que se nos presenta.

La segunda propiedad, la *emergencia*, por supuesto no se refiere a una situación imprevista que revista gravedad (como sería una emergencia médica, por ejemplo), el término se utiliza en el sentido de emerger, nacer, salir a flote. Se trata de la emergencia o el descubrimiento de algo que no estaba antes y ahora sí.

Como hemos dicho, en un sistema podremos en general identificar subsistemas. Éstos, suponemos, son de complejidad inferior al

principal y por tanto podríamos clasificarlos en un nivel más bajo. Pues bien, cuando estudiamos los subsistemas, encontramos propiedades que les caracterizan y nos ayudan a describirlos y comprenderlos, sin embargo, en algunas oportunidades, cuando tratamos de deducir de sus propiedades alguna otra que habíamos notado antes en el sistema, resulta no ser posible; entonces, ¿de dónde sale esta propiedad del sistema que no se deriva de la combinación de las propiedades de sus subsistemas?... simplemente emergió. Es una propiedad emergente, es decir, una propiedad que no se deriva del ensamblaje de los componentes de inferior complejidad, sino que surge, nace, sale a flote, sólo cuando todos ellos componen el sistema y no cuando fueron aislados y luego relacionados.

En nuestro problema de la biblioteca podríamos identificar muchas de estas propiedades emergentes. Veamos sólo una: resulta que un libro tiene ciertas propiedades intrínsecas, digamos el número de páginas, los autores, el tema, entre otras. Es fascinante que, por más que busquemos, no tiene entre sus propiedades intrínsecas la ubicación. El libro es tal sin importar en donde se encuentre físicamente. Luego, su ubicación es una propiedad emergente. Sólo surge cuando se combinan libros y estantes formando una biblioteca. Entonces, y sólo entonces, comienza a tener sentido hablar de la ubicación del libro y podemos adjudicarle, ahora considerado en el marco de nuestra biblioteca, una propiedad que en sí mismo no tiene, su ubicación.

Para el analista de sistemas, la importancia del conocimiento de esta realidad estriba en que cuenta con una explicación racional sobre aspectos que a primera vista parecerían carecer de lógica, previniéndole también sobre la trascendencia de considerar el sistema en su totalidad y no como simple compuesto de partes, so pena de fallar en su análisis.

Otro ejemplo muy ilustrativo de las propiedades emergentes de los sistemas es la molécula de agua. El agua, en su aspecto molecular, se compone de dos átomos de hidrógeno y uno de oxígeno (H_2O). Si consideráramos estos átomos como los subsistemas del sistema agua y pensáramos un poco, entre muchas cosas notables del agua notaríamos lo siguiente: Por un lado, el hidrógeno es un elemento químico sumamente inflamable, arde con mucha facilidad y fuerza; por otro, el oxígeno es un acelerante de la combustión y su presencia es necesaria para que se produzca, sin embargo, el agua actúa como inhibidora del fuego. ¿Suena paradójico? La cualidad de inhibidora del fuego que tiene el agua

es una propiedad emergente. Proviene no tanto de las características químicas de sus componentes como de sus características morfológicas cuando se combinan, formando la molécula, dos átomos de hidrógeno y uno de oxígeno. Esta propiedad emerge o nace sólo cuando el agua es agua y no cuando es hidrógeno u oxígeno por separado.

1.2 Caracterizaciones de sistemas

Para comprender el enfoque de sistemas puede resultar útil describir las formas en que éstos pueden caracterizarse. Una primera clasificación gruesa de sistemas, los divide simplemente en sistemas *duros* y sistemas *blandos* (figura 2). Esta sencilla caracterización conduce al analista de sistemas en la dirección correcta, para comenzar.

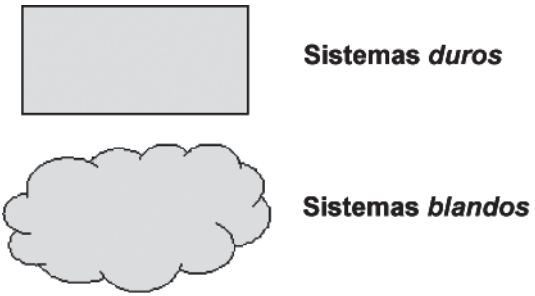


Figura 2. Clasificación general de los sistemas

Los sistemas *duros* son aquellos en que las cosas se ven con mayor claridad, aquellos que tienen firmemente definido su propósito y fronteras. A éstos resulta relativamente fácil encontrarles subsistemas (aunque ello no significa que los subsistemas resulten fáciles, de hecho pueden llegar a ser muy complejos). Hablamos de sistemas que aparecen claros a la luz del investigador.

Se trata de situaciones con muy poca o ninguna intervención humana. Sistemas cuyo propósito es puntual y conocido, por ejemplo, un torno: sistema–máquina con un motor y un eje rotatorio que da vueltas a un objeto para que un maestro tornero lo moldee a su gusto; un computador: mecanismo de funcionamiento electromecánico que pone a

disposición del hombre una serie finita y conocida de características de procesamiento y memoria; una línea de producción o ensamblaje de vehículos, y muchos más. En todos estos casos, los sistemas pueden llegar a ser muy complicados en su interior, pero si lo pensamos bien, no lo son en su conceptualización.

Los sistemas *blandos,* por otra parte, son aquellos en los que su definición y propósito nos mueven a la reflexión profunda. Aquellos en los que las fronteras son esquivas para el observador y no es posible definir certera y absolutamente su misión. Consecuentemente la búsqueda de subsistemas también se complica, y aunque algunos puedan parecer incluso mucho más sencillos que los sistemas duros, resultan, por su capacidad de cambiar rápidamente, más complejos que aquéllos. Hablamos de sistemas con alto grado de intervención humana.

El hombre, con su conciencia y libre albedrío, decide qué hacer en momentos determinados, no se ciñe a las reglas estrictamente, y en situaciones similares, en momentos diferentes del tiempo, puede actuar de manera diametralmente opuesta. Imprime una dinámica al sistema que complica las relaciones entre sus componentes y hace difícil su interpretación o descripción para el análisis.

Se trata de aquellos sistemas en los que el hombre interviene formando una organización. Pongamos un ejemplo: nuestra familia. ¿Qué propósito podríamos adjudicarle a nuestra familia?, ¿podríamos fácilmente decir que los tíos, primos, el compadre, la amiga del alma, están afuera?, ¿o adentro?, ¿será posible descubrir todas las respuestas que dará mi familia a situaciones que se le presenten?, ¿se conservarán estas respuestas a través del tiempo de manera que situaciones idénticas en el futuro produzcan la misma reacción de mi familia? Como habrá imaginado el lector, las respuestas a estas interrogantes no son evidentes. Luego, estudiar este segundo tipo de sistemas resulta una tarea mucho más complicada que la de hacerlo con aquellos del primer tipo.

La mayoría de los problemas de sistemas no son puramente blandos ni puramente duros. Una posición práctica, dada la complejidad de los sistemas blandos, procura limitar o simplificar el problema llevándolo al extremo duro de la cuerda tanto como sea posible, sin perder de vista los efectos blandos de la situación. Hay que limitarlo en su complejidad de alguna manera para que se ponga a nuestro alcance y así mantenerlo dentro de parámetros manejables. Al final siempre po-

dremos volver a él una y otra vez, en una espiral de refinamiento, estudiándolo cada vez con mayor profundidad y amplitud.

Nuestro ejemplo de la biblioteca podría ser clasificado entre los extremos. Es un sistema blando en cuanto interesa a las personas que la utilizan y son ellas las que deciden como manipularla; duro en cuanto los libros son objetos inanimados cuya presencia física ocupa un espacio que puede ser controlado sin ambigüedad.

Otra clasificación más fina en la escala, pero que también incluye ambos extremos del espectro, nos la proporciona Checkland:

- *Sistemas naturales*
- *Sistemas humanos*
  - Sistemas físicos diseñados
  - Sistemas de actividad humana
  - Sistemas abstractos diseñados
- *Sistemas trascendentales*

Los *sistemas naturales* son aquellos que proporciona la naturaleza. Están preestablecidos para los hombres y aunque actuemos para transformarlos (a veces atentando contra nuestra propia supervivencia), siguen cumpliendo su destino como si estuviera escrito en alguna parte. Ejemplos son el Sistema Solar, el clima, el sistema hidrológico, el cuerpo humano y otros. Aunque los seres humanos no los pusimos allí, sí podemos describirlos y estudiarlos en términos de sistemas, bien para intentar controlarlos, bien para conocer sus efectos y características. Así, estudiamos el sol, la luna y los planetas, la meteorología, las mareas y corrientes hidráulicas, nuestro organismo y muchos otros.

Los *sistemas humanos* son aquellos que hemos construido con un propósito determinado. Responden a un diseño surgido de la inteligencia y necesidad de la humanidad y procuran incrementar nuestra propia capacidad civilizadora. Sobre estos tipos de sistemas, las opiniones se dividen: por un lado están algunas corrientes naturalistas o conservadoras, que se oponen, muchas veces con razón, al avance del hombre en su afán de construir cada vez más sofisticados sistemas para su propio beneficio. Argumentan que en reiteradas ocasiones, estos sistemas, lejos de acrecentar o perfeccionar nuestra civilización, alteran la naturaleza de las cosas sin control alguno y producen que la humanidad se deshumanice cada vez más.

Sobran los ejemplos de este tipo de sistemas de dudoso beneficio: sistemas de armas nucleares, sistemas que alteran el comportamiento de la atmósfera o el equilibrio hidrológico en pos de un pírrico beneficio, alteraciones genéticas que afectan la vida sobre el planeta, etc. Muchas veces, estos sistemas, aunque no necesariamente perjudiciales por sí mismos, lo son en extremo utilizados inapropiadamente.

Desde otra óptica, llamémosla modernizadora, los sistemas humanos se conciben de forma optimista en términos de acrecentar el nivel de vida de los hombres, lo cual hará, hipotéticamente, ganar tiempo a la humanidad para moldearla cada día más humana, dejando las tareas no reflexivas o rutinarias en manos de sistemas específicamente concebidos. Esta forma de ver el "vaso medio lleno" en lugar de "medio vacío", propone que se emplee la inteligencia para alcanzar un nivel de vida más racional, aunque menos natural. De este lado del abanico está la tecnología, y en especial la que toca este texto: la tecnología de la información.

Estos asuntos nos llevan al terreno de la ética y la propia conciencia. Una vez más, se trata de encontrar una posición intermedia entre los extremos, una postura en equilibrio: procurar sistemas humanos que mejoren nuestras condiciones de vida sin desmejorar las de otros seres humanos ni las del resto de los seres vivos que comparten con nosotros el planeta. Debe profundizarse en el estudio de la tecnología con visión crítica en cuanto a los aspectos de inequidad social, dominio de un grupo sobre otro, desigualdad de oportunidades y sus implicaciones sobre los cambios culturales y educacionales necesarios para resolver estos problemas.

De este grupo de sistemas humanos, los primeros en ponerse en práctica, cronológicamente hablando, fueron los físicos-diseñados. Ellos comprenden todos los que, gracias a la intervención humana sobre los recursos naturales, resultan en herramientas a las cuales el hombre no cambia su composición molecular, pero que transforma convenientemente para darles un destino o propósito esencialmente diferente al que tenían en la naturaleza. Un martillo, una casa, la rueda, un corral, un hacha, una carreta, un arco y una flecha, entre infinitos más, son ejemplos de este tipo de sistemas. La flecha, por mencionar uno cualquiera, es un pedazo de madera unido a un trozo de material metálico o piedra. Sus componentes no han sido alterados en su estructura atómica, sólo se ha ideado una manera diferente de combi-

narlos o se les ha dado una forma tal que ahora sirven a un propósito definitivamente distinto del original. La flecha ayudó a la humanidad al mejorar las condiciones en que los primeros humanos cazaban presas a distancia, perfeccionando así sus mecanismos de alimentación. La rueda hizo posible movilizarse con mucha mayor rapidez. El corral disipó el enigma de la domesticación de animales. Y así muchos otros. Desde el punto de vista de la complejidad, éstos son los menos complicados. En cuanto a la clasificación general, están en el extremo duro del segmento.

Los siguientes en aparecer fueron los sistemas de actividad humana. El hombre, enfrentado a los retos de la supervivencia en un mundo hostil, busca formas de organización e interrelación con otros hombres y constituye sistemas que establecen reglas de convivencia cada vez más sofisticadas. Propone tareas a cada uno de los integrantes de la organización y logra en consecuencia sumar voluntades en torno a un propósito común. Al principio de la humanidad se dan formas muy precarias de organización, pero a medida que ha avanzado la historia, y por ende nuestra comprensión de la fuerza que este concepto da a la civilización, surgen más y mejores organizaciones que llegan en nuestros días a ser de complejidad a veces abrumadora y propósitos diversos. El Estado, un sindicato, una empresa, un grupo de estudio en la universidad, la Iglesia, el Ejército, por mencionar algunas, pertenecen a esta categoría. En general les llamaremos organizaciones para acentuar que se trata de sistemas principalmente integrados por seres humanos en busca de un propósito supuesto compartido. Desde el punto de vista de la complejidad, éstos son los más complicados. Se encuentran en el extremo blando de la clasificación general.

Los últimos en aparecer fueron los sistemas abstractos diseñados. Se trata de sistemas que no tienen presencia física, pero que no por ello, son menos reales. Surgen de la mente de los hombres generalmente para explicar su tiempo y circunstancia. Aquí entran las construcciones que ha aportado la ciencia, y ahora, mucho más recientemente, las tecnologías de la información. Son ejemplos de estos sistemas un modelo matemático o estadístico, una ley de la física, una ecuación de la química, un sistema de información. Nadie podrá jamás encontrar en la naturaleza algo como $F = m \cdot a$, pero si F representa la fuerza, m la masa y a la aceleración, cualquiera sobre nuestro planeta puede verificar esta igualdad (no sin rendir honores a su descubridor, *sir* Isaac Newton).

De igual forma, un sistema de información, por ejemplo, el que resuelve nuestro problema de la biblioteca casera, no podrá ser visualizado como un objeto tangible en la realidad, pero existe en cuanto es útil para llevar el control de nuestros libros.

Estos tipos de sistema son modelos o construcciones abstractas que toman lo esencial de una realidad física que quieren explicar, resolver o ayudar a mejorar y se materializan en forma de leyes y reglas, o bien herramientas "virtuales", imitando a la realidad en lo estructural, si bien omitiendo los detalles hasta donde sea posible. En cuanto a su complejidad, caminan entre los dos extremos del segmento según qué tan cerca o lejos están de tocar lo físico y lo organizacional.

Como mencionamos antes y ya habrá descubierto el lector, la temática que aborda este libro se mueve en esta última categoría. La informatización, en cuanto aplicación de las técnicas de la informática, es clasificada dentro de los sistemas abstractos diseñados. Mostraremos cómo se puede lograr, de la conceptuación a la implantación, un sistema de información cuyos pilares estén constituidos por otro sistema (de ideario mucho más concreto) que denominaremos sistema de bases de datos relacionales.

El tercer grupo, los *sistemas trascendentales*, engloba aquellos sistemas que trascienden la comprensión racional del ser humano o de la ciencia. Aquí se encuentran nuestras creencias, valores morales, constructos teológicos o metafísicos y asuntos por el estilo. No ahondaremos en ellos, pues escapan del ámbito expuesto en esta obra.

1.3 Análisis de sistemas de información

La palabra "análisis" evoca reflexión, descomposición, comprensión. Pues bien, el análisis de sistemas no es más que una herramienta metodológica que contribuye a la formalización del enfoque de sistemas. Su propósito es materializar la idea ya mencionada "divide y vencerás". Cuando se analiza cualquier situación, y particularmente una en términos de sistemas, la idea es desmenuzar, desagregar, tratar de reducir la complejidad para aumentar el grado de comprensión que sobre la situación se tenga.

De esta forma, analizar un sistema es en realidad describirlo, descubriendo en el intento el suprasistema y los subsistemas principales

o estructurales, aquellos que son esenciales para conocerlo en profundidad. Tanto los subsistemas como el suprasistema deben hacerse explícitos y sus relaciones con el sistema que analizamos deben quedar plasmadas de la forma más precisa posible. Hacer análisis de sistemas es además descubrir sus fortalezas y debilidades, proponiendo de manera formal una suerte de función objetivo que mida el resultado, es decir, el grado de cumplimiento de los propósitos, tanto del sistema como lo estamos concibiendo, como del sistema una vez hayamos intervenido en él.

Así pues, no se trata de un simple juego de palabras, es en realidad una técnica poderosa. Valga destacar también que cuanto más *duro* resulta el sistema, más poderoso es el análisis de sistemas como herramienta metodológica. Por el contrario, cuanto más *blando*, menos efectiva la técnica.

Sobre lo que nos resulta en este texto particularmente interesante, el problema de la información, se aplican las ideas del análisis de sistemas en un contexto aún más restringido. De hecho, la secuencia metodológica para el desarrollo de sistemas de información, que por supuesto parte del análisis de sistemas reales, es bien conocida y ha sido profusamente aplicada por años para tratar el asunto. Antes de exponerla aclaremos algunas ideas importantes que no son evidentes y que lamentablemente, sin embargo, no es común verlas abordadas:

1. Postulamos que un sistema de información no es el sistema que inicialmente nos ocupa. El sistema que nos ocupa inicialmente es aquel, real, tangible, existente, sobre el cual y para el cual buscamos desarrollar un sistema de información. De hecho, analizamos el sistema que nos ocupa para tratar de comprenderlo con la mayor exactitud posible, captando el fondo de su accionar justamente para poder construir un modelo que lo represente fielmente, y a partir de ese modelo proponer un nuevo esquema de organización y de información que sirva al propósito de mejora formulado sobre el sistema original.

   Para ilustrar estas ideas tomemos nuevamente el caso de la biblioteca. En este sistema, lo que inicialmente nos ocupa es la biblioteca misma, sus libros, sus estantes, sus usuarios, es decir, la real, la de verdad. Por otro lado la estamos viendo desde el punto de vista de la información, puesto que buscamos comprender su funcionamiento y captar su esencia para lograr un sistema de información útil a la hora de controlarla, manejarla. En el fondo

suponemos que el sistema real, la biblioteca, tal como es al inicio, resulta susceptible de ser mejorado, y que la incorporación de un sistema de información al sistema real, obrará tal mejora.

El *sistema biblioteca* y el *sistema de información biblioteca* no son el mismo. El primero existe, el segundo es un modelo (en términos de información y datos) que implementa las funciones más importantes para el registro y control de la biblioteca real. El sistema biblioteca puede vivir, y de hecho vive, sin el sistema de información biblioteca; por el contrario, este último pierde todo sentido sin aquél.

2. De lo dicho en 1 surge una cuestión interesante: Una vez que el sistema de información se completa, pasa a formar parte del sistema real, se convierte en uno más de sus subsistemas y, en consecuencia, el análisis que hiciéramos antes del sistema de información es diferente del que haríamos ahora, luego del sistema de información.

   El sistema de información imprime nuevas características al sistema real, emergen nuevas propiedades, comienzan a preocupar aspectos que antes no inquietaban, en fin, el sistema cambia. Por supuesto, ya que dijimos que enfrentábamos la tarea de desarrollar un sistema de información para mejorar el sistema real, no debe extrañarnos que éste cambie luego de la incorporación del sistema desarrollado. Eso era precisamente lo que pretendíamos. Lo que no es automáticamente cierto es que el sistema real haya cambiado para bien. Existe la posibilidad de que la introducción de este nuevo factor –que podríamos llamar de perturbación-, afecte negativamente el sistema real, en cuyo caso, el ansiado objetivo de mejora no se ha logrado. De hecho, si esto ocurriera, nos habríamos topado con un efecto contrario al deseado.
3. Ahora, con plena conciencia de que el sistema real y el sistema desarrollado no son la misma cosa, y de que el segundo afecta dramáticamente al primero, es fácil comprender la importancia del análisis de sistemas. Si en el proceso de estudio del sistema real no se logra asir la esencia de la cuestión, si no se profundiza y se descubren las relaciones importantes entre sus subsistemas, si no se logra desnudar la realidad de su funcionamiento y describir sus características verdaderamente influyentes, difícilmente se puede llegar a proponer un nuevo sistema desarrollado a partir

de bases falaces, de bases engañosas. En tales circunstancias nos espera inevitablemente el fracaso.

El análisis del sistema es por ende una tarea importante. No es la única tarea importante, pero sí una de ellas. A grandes rasgos, durante el análisis deben considerarse los siguientes aspectos: ¿cómo es el sistema real?, ¿qué objetivo de mejora se persigue con el sistema a desarrollar?, ¿cómo medir el grado de cumplimiento del objetivo de mejora?, ¿cómo impactará el nuevo sistema desarrollado sobre el sistema real?, ¿qué habrá de cambiar y cómo deberá actuar el sistema real cuando tenga inmerso el nuevo sistema desarrollado?

O dicho con otras palabras, se parte de un sistema que existe. Generalmente, este sistema actúa o se comporta de una manera que debe ser descubierta y hecha explícita. Esta forma de hacer las cosas en el sistema actual es objeto de estudio, de comprensión. Surge la crítica y las consecuentes propuestas de mejora. Sobreviene un nuevo sistema desarrollado que subsana las deficiencias (esa es la creencia) y hay que decir ahora cómo se comporta éste. Obviamente, la diferencia entre el sistema original y el nuevo desarrollado (éste último con el sistema de información inmerso en él), es lo que medirá aquella función objetivo mencionada y representa el grado de mejora alcanzado.

1.4 ¿Por qué sistemas de información?

Esta es la primera pregunta que el analista de sistemas debe responder con propiedad. La respuesta no es trivial, ni obvia. Muchas organizaciones, especialmente en nuestro medio aún subdesarrollado, hipertrofiado en el área de servicios públicos y el comercio, en contraposición a lo industrial o transformador, desconocen o desconfían del uso de la tecnología como apoyo a sus procesos. A no pocos gerentes les parece costoso y hasta inútil emprender proyectos de sistemas de información (SI), especialmente cuando para ello deben disponer recursos que creen mejor empleados en otras áreas. Lamentablemente, algunos de nuestros gerentes, especialmente públicos pero privados también, temen ser desplazados o controlados por la vía de los SI. Sin embargo, son muchas y variadas las ventajas que los SI aportan a las organizaciones.

La más importante de todas las bondades que trae consigo la implantación de SI en una organización proviene del proceso mismo de su desarrollo. En efecto, el proceso de desarrollo de un SI requiere escudriñar a fondo la organización, descubrir junto con ella la manera en que hace las cosas, documentar prolíficamente su estructura organigramática, sus procesos de toma de decisiones, sus formas de operación y sus prácticas administrativas.

En el camino, el analista de sistemas y todos los que se involucran en el desarrollo del SI aprenden y comprenden mucho mejor a la organización, de lo que lo hacían al comenzar. Esta es por sí misma una inmensa oportunidad para que la organización reflexione de forma crítica sobre su quehacer y busque mejorar sus prácticas cotidianas.

Otra muy buena razón por la que emprender el desarrollo de SI proviene de considerar la información como un recurso valioso que debe ser controlado. En este sentido, pensar la organización en términos de los flujos de información que en ella se generan y utilizan, tanto para la operación normal o cotidiana como para la toma de decisiones, estructurar lo más que sea posible esta información, capturarla y recuperarla en medios computacionales ágiles, eficientes y versátiles, suele contribuir enormemente a la estandarización de los procesos tanto productivos como de servicios o administrativos, y la estandarización de procesos representa una meta bien conocida y anhelada por toda organización que tenga propósitos de trascendencia.

La estandarización y la documentación de sus prácticas disminuye el grado de arbitrariedad en las decisiones (algunas veces confundido con heurística), brinda a la organización flexibilidad en el manejo de sus recursos humanos (haciendo que desaparezcan las personas imprescindibles, pues con el entrenamiento apropiado podrían ser sustituidas por otras igualmente calificadas) y da tiempo extra para mejorar continuamente (pues la forma en que se acomete lo rutinario está claramente establecida y es bien conocida por todos, lo que libera tiempo de reflexión que puede dedicarse a profundizar en problemas más complejos).

Otra buena razón para emprender desarrollos de SI es mantenerse al nivel de los competidores. En situaciones de competencia, especialmente en organizaciones empresariales con fines lucrativos, organizarse mejor que el rival hace a menudo la diferencia requerida para ganar la carrera, lo que en este caso significa posicionarse mejor que

otros en el mercado a que se aspira. Disminuir tiempos de respuesta y costos de operación libera recursos que pueden emplearse en otras áreas, por ejemplo, mercadeo, inventarios, publicidad y otras, los cuales, si la competencia no los cuenta también, pueden representar una ventaja decisiva.

Por otro lado, en términos ahora meramente económicos, el analista de sistemas debe estar consciente de las inversiones que acarreará el desarrollo de SI, tanto como de los retornos esperados, en los tiempos previstos. Es precisamente aquí donde las cosas aparecen menos claras. Este aspecto, el económico, es el más problemático que enfrenta un analista cuando intenta convencer a una organización de desarrollar el SI.

El asunto de cuánto cuesta desarrollar e implantar un sistema de información versus cuánto del incremento en la ganancia podrá atribuírsele, es materia de estudios avanzados de costos, puesto que la mayoría de las contribuciones que un SI aporta a las finanzas de una organización, es intangible. Mucho de lo que un sistema aporta en lo financiero proviene de la comparación con lo que sucedería de no contarse con él. Otro tanto proviene de los ahorros que se logran por tener información oportuna y confiable sobre aspectos claves de la operación. Una gran parte de los beneficios económicos se desprende del incremento que proporciona en las capacidades de planificación y control de la organización, en fin, siendo que los SI en general no se desarrollan para una organización con el propósito de su venta directa, no es asunto sencillo determinar la relación costo–beneficio que traen consigo, lo que de ningún modo significa que no se obtengan ganancias económicas de ellos, bien por la vía del incremento de ingresos gracias a su contribución, o bien por la vía del ahorro sistemático y sostenido que aporten en el tiempo.

1.4.1 ¿Qué es un sistema de información?

Es un conjunto de procedimientos y acciones humanas, así como de elementos computacionales interrelacionados entre sí, cuyo fin es la producción de información confiable y veraz que apoye la toma de decisiones (rutinarias o no) necesarias para la marcha y el control de la organización de la que forma parte.

En la infraestructura de sistemas informáticos de una organización, los sistemas de información ocupan el lugar que se esquematiza en la figura 3.

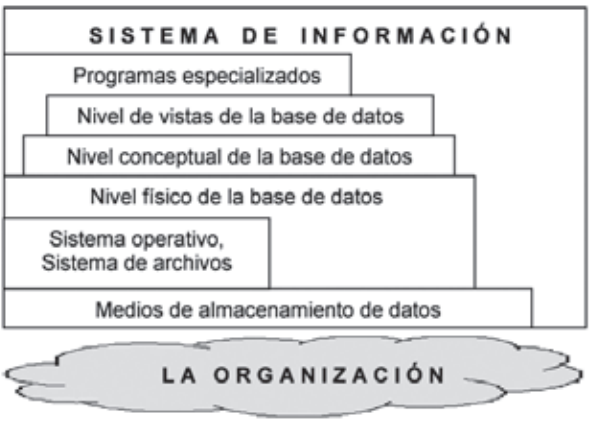


Figura 3. Lugar que ocupan los SI en la organización

La figura 3 muestra que los sistemas de información se construyen sobre una organización particular, pero también sobre los datos almacenados en medios computacionales, sobre la base de datos y, posiblemente, sobre programas de aplicación especializados. Todos estos elementos constituyen las bases sobre las que descansan los sistemas de información, así como las bases de un edificio son sus fundaciones.

1.4.2 Los datos, la información y los niveles de la información

Un *dato* es una característica atómica o elemental del objeto en estudio. Es la unidad básica de información. Por su parte, la *información* es una colección o agrupación de datos, conceptualmente relacionados entre sí, que al ser vistos integralmente enriquecen el **conocimiento** que teníamos sobre el objeto.

Luego, la diferencia entre *información* y *datos* estriba en que la primera incrementa el conocimiento de quien indaga, mientras que los segundos no lo hacen necesariamente. Podría argumentarse que al ser los datos el insumo básico de la información, éstos contribuyen a la ganancia de conocimientos; sin embargo, el conocimiento se gana sólo

cuando los datos se disponen de tal manera y se contextualizan hasta el grado en que aparecen revelando algo que no se conocía. La ganancia de conocimientos a partir de un conjunto de datos, lo cual los transforma en información, podría considerarse una propiedad emergente, tal como se planteó en secciones previas.

La idea central de los sistemas de información es capturar datos, procesarlos, disponerlos y recuperarlos de tal forma que se obtenga información. De allí su nombre y de allí la importancia de ver los flujos de datos de la organización, así como la información que se necesita, bien sea para operar, bien sea para tomar decisiones.

Por otro lado, ¿qué tipo de información necesita el ayudante de contabilidad para calcular la nómina de la empresa?, ¿qué tipo necesita el gerente de Planificación para tomar decisiones respecto de la nómina de la empresa?

La primera situación requiere información *operativa*, es decir, información necesaria para las tareas rutinarias de la organización, para mantener a la organización "operando" plenamente. Otros ejemplos de este tipo de información podrían ser las horas trabajadas por los empleados, las solicitudes al inventario, etc.

Figura 4. Triángulo de la información

La segunda situación requiere información *directiva*, es decir, información necesaria para tomar decisiones no triviales que afectarán el desempeño de la organización a largo plazo y que generalmente están bajo la responsabilidad de los "directivos". Ejemplo de este tipo de información es aquella que responde a interrogantes como ¿cuánto producir el año entrante, o dentro de tres años?, ¿qué estrategias serán

necesarias para entrar con éxito en un determinado segmento del mercado?, ¿cuándo y cuánto ordenar al proveedor para optimizar el costo del inventario?, etc.

La figura 4 representa el triángulo de la información y sugiere que la información *directiva* (y por tanto los sistemas de información orientados hacia la dirección de la organización), se basa en la información *operativa* (proporcionada por los sistemas de información orientados a las operaciones).

La operación cotidiana de una organización requiere y produce información de corte operativo. Esta información, resumida, agrupada, de ser posible convertida en gráficas, estadísticas o pronósticos, es la base de la información de mayor nivel que se utiliza para tomar decisiones. En consecuencia, en una organización no informatizada, el primer paso siempre será emprender el desarrollo de sistemas de información operativos para poder luego desarrollar sistemas de información directivos. Como comprenderá el lector, realmente no son unos más importantes que los otros, ambos son igualmente importantes en la infraestructura informática de una organización.

1.5 Metodología ADDI para la elaboración de sistemas de información

Como asomamos en la sección anterior, todo proyecto de sistemas de información parte de la definición del sistema real y el modelado del sistema que será desarrollado. La primera tarea, entonces, tiene que ver con el análisis de sistemas.

Para hacer estas ideas y las que siguen un tanto menos abstractas y proponer un esquema de trabajo que oriente (sobre todo al que se inicia) en el abordaje de problemas de esta naturaleza, propondremos en lo sucesivo un esquema metodológico simple. No reproduciremos en este texto todas las corrientes ni todos los puntos de vista por considerar el tema fuera de alcance en un nivel introductorio. Baste decir aquí que se han desarrollado un sinfín de experiencias en la aplicación de diferentes metodologías de desarrollo de sistemas de información, con mucho, aceptable, escaso o ningún éxito en distintas circunstancias. El consenso general en nuestros días, luego de ver correr bastante "agua bajo el puente", es que con las herramientas tecnológicas ya desarrolla-

das, sea cual fuere la metodología preferida o el enfoque adoptado, éste tiene que ser flexible, adaptable, dinámico y no como se suponía en los albores de esta ciencia, dogmático, inflexible o seguido a pie juntillas.

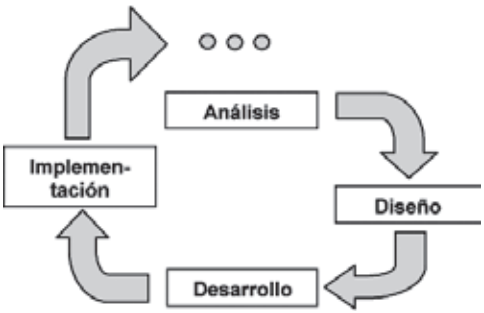


Figura 5. Metodología para el desarrollo de sistemas de información (ADDI)

Desconocemos si existen estudios formales publicados sobre este tema aplicados a nuestra realidad; no obstante, un interesante estudio crítico publicado por Middleton (3) aborda este aspecto en el desarrollo de sistemas de información para el sector público en el Reino Unido, y prueba que las metodologías rígidas con alta inclinación hacia los detalles y excesiva rigurosidad en los pasos a seguir, no han conducido en general al éxito. Por el contrario, allí queda claro que aquellas organizaciones que optaron por seguir este camino como un dogma, encontraron que sus soluciones estaban siempre retrasadas, y en una significativa proporción, el resultado final fue decepcionante.

Así pues, los lineamientos que siguen deben tomarse como una orientación y no como reglas o normas a cumplir al pie de la letra. Suele ser útil, sobre todo cuando aún no se tiene práctica en el asunto, echar mano de estas ideas para comenzar. A medida que el practicante adquiere experiencia, debe necesariamente ver la metodología con enfoque crítico y, por así decirlo, "descubrir" su propia forma de hacer las cosas. Esto no significa en lo absoluto que no deba existir un método, menos aún que no deba seguirse una cierta metodología, lo que significa es que todo método por sí mismo no es garantía de éxito, y menos cuando se aplica ciegamente, acríticamente.

A grandes rasgos, metodológicamente hablando es posible identificar cuatro etapas en el ciclo de desarrollo de un sistema de información. La figura 5 muestra estas cuatro etapas y, para continuar la tradición de los computistas de poner acrónimos a casi todo, las haremos un conjunto al que nos referiremos como la metodología ADDI (por **A**nálisis, **D**iseño, **D**esarrollo e **I**mplantación).

Nótese que la flecha que parte de la última de las etapas no llega precisamente a la primera de ellas, sino un tanto más arriba. Este detalle de la figura pretende resaltar el hecho cíclico y a la vez incremental de la metodología. En realidad, el estudio de sistemas de información es una práctica que nunca acaba, una rutina en espiral que se constituye en un ciclo virtuoso de mejoramiento continuo. Partiendo de una organización inicial se implanta un sistema de información que impacta la organización original, transformándola en otra parecida pero algo mejor. Ahora ya no es la original y por tanto el análisis vuelve a ser necesario, considerando las nuevas condiciones. La idea es que en cada paso del espiral, la organización mejore su desempeño por la vía de perfeccionar incrementalmente su sistema de información.

Dentro de cada una de estas etapas hay pasos concretos que deben darse, a saber:

1. En la etapa de análisis ocurre, por supuesto, el análisis del sistema. Esto es, el estudio detallado del sistema real y la descripción precisa de sus principales subsistemas y del suprasistema que lo contiene. No necesariamente el sistema que luego se diseñe, incorporará de un solo golpe todas las mejoras posibles. Deberá tratar de hacerlo al máximo, pero estará restringido por dos circunstancias: el grado de adaptabilidad del sistema real a los cambios y el tiempo con que se cuente para llegar a una primera solución. No obstante, mencionar con la mayor profundidad y alcance posibles los aspectos susceptibles de ser mejorados, siempre será una guía valiosa para las siguientes etapas.
   Un resultado muy importante del análisis es la explicitación de los requerimientos de información que deberá satisfacer el nuevo sistema. También, como producto de esta etapa, se espera un documento que contenga información sobre el sistema estudiado, sus definiciones, supra y subsistemas descubiertos, su propósito, sus objetivos, cómo se organiza para cumplirlos, cuáles procesos se desencadenan en su interior, cómo se llevan a cabo éstos y

los problemas de información que confronta. Este último aspecto implica ver de forma crítica el comportamiento del sistema estudiado y pasar revista a todos aquellos puntos que a juicio del analista sería menester mejorar. Podría decirse que en el caso de los sistemas de información, los problemas se convierten en buena medida en la función objetivo de mejora.

2. En la etapa de diseño, un primer asunto que debe dilucidarse es la factibilidad de llevar adelante el proyecto de sistemas. A la luz de lo expuesto en el análisis, el diseñador concibe *grosso modo* el nuevo sistema e imagina las condiciones que deberán cumplirse para echarlo a andar. Estas condiciones comprenden aspectos de *hardware*, *software*, pero sobre todo humanos y de organización.
   Una segunda tarea en la etapa de diseño es el modelado de la organización en términos de datos y sus flujos, consecuencia directa de ahondar en el análisis de requerimientos de información. El diseño de la base de datos representa una actividad de corte técnico y también creador, muy importante dentro de la metodología ADDI. Desmenuzando la información requerida por la organización hasta dar con los datos precisos de interés, el analista propone un modelo conceptual que luego será llevado al computador y conocido como la base de datos del sistema. Aun cuando hay muchos, en este texto nos concentraremos en los modelos *Entidad–Asociación* y *Relacional*, como herramientas para la elaboración del diseño de la base de datos.
   También debe diseñarse con cuidado la interfaz hombre–máquina: pantallas de entrada y salida de datos, consultas, reportes a imprimirse, menús de opciones e iconografía, entre otros elementos de la herramienta computacional. Los procesos, procedimientos y mecanismos de uso que deberán seguirse para emplear correctamente el nuevo sistema de información, tanto desde el punto de vista del operador como desde el punto de vista de aquellos que generan o utilizan la información (que pueden o no ser operadores), son también objeto de diseño en esta etapa.
   El plan de informatización, esto es, el plan de incorporación de la informática en la organización, finaliza la etapa de diseño haciendo explícitos no sólo los aspectos técnicos mencionados antes, sino también los de transformación organizacional imperativos

para que el nuevo sistema alcance el éxito (y con él, por supuesto, la organización). Cambios en la estructura organigramática, simplificación de procesos administrativos e integración del nuevo sistema en las labores habituales de los integrantes de la organización, pueden ser incorporados como parte del diseño del plan de informatización.

De estas actividades se esperan los siguientes productos: (1) El nuevo *manual de organización* luego de la implantación del sistema, si resultare que éste ha afectado la estructura de la organización. (2) El nuevo *manual de procesos* que, partiendo de los procesos que la organización adopta, incluya los nuevos procedimientos y tareas humanas necesarias en adelante. (3) El *diseño de la base de datos*, es decir, un documento que contenga todos los detalles de forma y fondo sobre los datos a ser almacenados y que servirán de soporte al nuevo sistema, y (4) El *documento de especificaciones de la interfaz*.

Aun cuando en nuestros días siguen siendo de mucha utilidad los documentos escritos, los productos (3) y (4) suelen presentarse y discutirse con éxito en el ámbito de un "prototipo del sistema", en lugar de sólo papel. La elaboración de tal prototipo, esto es, un programa computacional que contenga sólo algunas de las características del sistema y de la base de datos, generalmente sin incluir ninguna validación ni algoritmo sofisticado, suele dar excelentes resultados como apoyo al diseño del nuevo sistema de información, sobre todo cuando se trata de discutir con la organización las características deseables del nuevo sistema.

3. En la etapa de desarrollo, los documentos elaborados y las discusiones sostenidas, tanto en la etapa de análisis como de diseño, toman cuerpo en un producto de software concreto que debe ser construido. El desarrollo de un sistema comprende la programación en código objeto, es decir, en código que un computador puede entender y obedecer, de todos los aspectos del producto de *software* que será el responsable de capturar los datos, almacenarlos eficientemente en bases de datos y recuperarlos en forma de información cuando se les requiera.

   Los programas de computadora que sean desarrollados deben atender a las especificaciones que se obtuvieron en la etapa de diseño. En esta etapa, la base de datos se materializa temporal-

mente en un servidor de base de datos de pruebas, sólo para efectos de la elaboración de los programas que accederán a ella para insertar, modificar, eliminar o consultar información en todas las diversas formas descritas en los documentos de diseño.

La interfaz hombre–máquina para la inserción de datos o presentación de consultas e informes debe quedar completamente programada, desde las ventanas a las que el usuario accede al interactuar con el sistema de información hasta los menús de opciones y otros elementos de la interfaz que le sean de utilidad. Todas las validaciones sobre los datos que hayan sido especificadas deben ahora transformarse en reglas programadas para el acceso del sistema a la base de datos.

Una vez programado, el producto software debe ser sometido a pruebas exhaustivas, de preferencia por parte de otras personas distintas a las que lo desarrollaron. Este tipo de pruebas ocurren en un ambiente controlado por los desarrolladores. Un juego de pruebas diferente, en el que las condiciones no son controladas del todo, ocurre en la etapa de implementación.

Por último, en esta etapa y una vez programado el *software*, deben elaborarse dos nuevos documentos de importancia capital: El *manual del usuario* y el *manual del administrador del sistema*. El primero de los documentos describe en forma precisa cómo se utiliza la herramienta programada, en estrecha relación (y posiblemente vinculado mediante enlaces de hipertexto) con el *manual de procesos*. En él deben quedar por escrito las acciones a realizarse en la herramienta de *software*, en respuesta a los procesos y procedimientos que se llevan a efecto cotidianamente. El segundo documento contendrá toda la información necesaria para mantener el nuevo sistema, incluyendo las especificaciones de *hardware* y *software* que amerite su ejecución, los índices, claves y demás especificaciones de diseño de la base de datos que se hayan materializado y los parámetros de configuración de todos los componentes del nuevo sistema.

4. En la etapa de implementación, ya probado el sistema (hasta donde es posible hacerlo desde el punto de vista de la programación computacional), su producto de *software,* así como todos los documentos producidos, son entregados a la organización y puestos en operación en un ambiente de trabajo real.

El sistema de información pasa entonces a utilizarse y, dentro de un lapso razonable que generalmente depende de la envergadura del proyecto, es sometido a nuevas pruebas de funcionamiento, ahora contrastando las nuevas acciones con los resultados que se obtienen de él contra las acciones y resultados anteriores a su existencia. Un sistema de información correctamente desarrollado debe convertirse en una herramienta de uso habitual sin mayores traumas.

En esta etapa se afinan detalles que podrían aparecer y se corrigen problemas de organización del trabajo que puedan ocasionar inconvenientes a la nueva organización, hasta el punto de dejar funcionando completamente a satisfacción del usuario todos los aspectos del producto entregado.

En los tiempos que corren, los sistemas de información son elaborados por profesionales del área. Pueden formar parte integrante de la propia organización a la que pertenecerá el sistema o pueden ser consultores externos, contratados para tal propósito. En ambos casos, los productos, documentos y prototipos generados en el proceso son de gran importancia, cuanto más si los desarrolladores conforman un equipo multidisciplinario integrado por varias personas, cada una con sus competencias. Por ejemplo, un equipo de desarrollo puede incluir analistas de sistemas, analistas de organización y métodos, programadores, analistas de pruebas y redactores de documentos.

El sistema de información entregado, completamente documentado y operando a plenitud es, como asomamos antes, un peldaño más en la escalera de mejoramiento continuo de la organización. Constantemente debe ser objeto de revisión y análisis crítico, lo cual casi siempre determina el planteamiento de un nuevo proyecto de sistemas que abre camino a un nuevo ciclo ADDI, y así *ad infinitum*. Gracias a este hecho, los profesionales de sistemas que pueden ser considerados buenos son aquellos que permiten con su profesionalismo este salto cualitativo y continuado en el tiempo, es decir, aquellos que hacen buenos sistemas, en el sentido de que los sobreviven y no los necesitan para funcionar, o malos cuando se enclavan en un sistema programado convirtiéndose ellos mismos en parte de la solución, y por ello en parte del problema, al no permitir e incluso frenar el avance incremental que propone la metodología ADDI.

1.6 El papel de las bases de datos

Hasta entrada la década de los años 1970, el enfoque imperante para desarrollar sistemas de información en las organizaciones seguía un esquema divisionista, isomorfo con la propia estructuración interna de la organización. Si una organización tenía departamentos de nómina, contabilidad, inventarios o producción en su organigrama, y contaba con sistemas de cómputo para sus operaciones, seguramente tenía distintos sistemas de información, uno para cada departamento (figura 6).

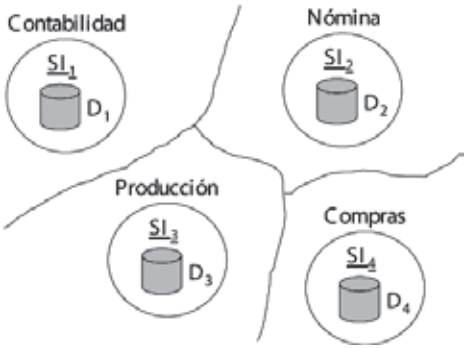


Figura 6. Enfoque "divisionista"
para los sistemas de información

Cuanto mayor la organización o más caótica su estructura (ejemplos de caos estructural son la mayoría de nuestras organizaciones gubernamentales, incluidas por cierto las propias universidades del Estado), las posibilidades de que dichos sistemas hubiesen sido diseñados e incluso construidos por personas diferentes, sobre plataformas de *hardware* y *software* incompatibles, eran considerables.

"Pedro Pérez" de Nómina no era el mismo "Pedro perez" de Contabilidad ni el mismo "pedro Péres" de Producción, aunque sí era el único "Pedro Pérez" que trabajaba en la organización. "María Rivas" tenía una calificación definitiva en su facultad que no coincidía con la calificación registrada en el sistema central de la universidad, ni con la que de ella tenía la oficina de grados a la que llegó cuando le tocó graduarse, lo que la puso a dar preocupantes carreras.

La entrada triunfal de los microcomputadores y computadores personales o de escritorio, el abaratamiento y la masificación de sus tecnologías, la marcha a paso redoblado hacia una verdadera industria del *software* para computadores y las redes telemáticas, abrieron camino para la solución de una gran variedad de problemas que este enfoque divisionista conllevaba.

Comenzó entonces a fines de la década de los años 1970 la discusión sobre el papel que desempeñan los datos. Se estableció rápidamente la idea de que los datos son un recurso valioso, y de que como tales deben ser resguardados, administrados y bien aprovechados por todos los integrantes de la organización.

Resultó evidente, por ejemplo, que si en la nómina estaba buena parte de la información del personal, no había necesidad de tener también esta información en las cuentas por cobrar de la contabilidad, o en donde se guardaba lo producido en el día.

El enfoque divisionista, centrado en las funciones que el sistema de información debía apoyar, dio paso al enfoque de bases de datos, centrado primero en los datos que nutren a los sistemas y luego en las funciones para las que se les necesitan (figura 7).

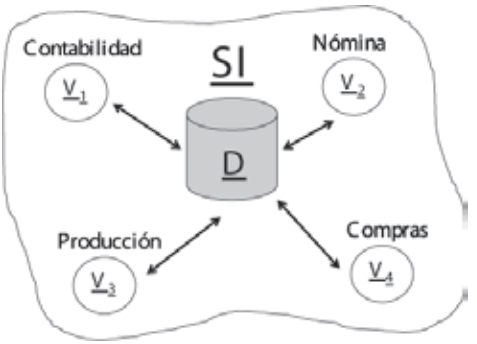


Figura 7. Enfoque de base de datos
para el sistema de información

Los modelos de datos, la visión del banco de datos primero, la base de datos después, la interconexión de computadoras ya no tan costosas, la asignación de responsabilidades especializadas en la admi-

nistración de los datos como activos de las organizaciones, lograron en corto tiempo que la idea de sistemas aislados para cada departamento cediera el paso a la de sistemas interconectados, compartiendo datos de un mismo almacén.

En consecuencia, ya no se habla de los sistemas en plural sino, mejor, del sistema, en singular. *Una organización, un sistema de información*, es el lema. No importa la estructura interna de la organización o las distintas aplicaciones computarizadas, módulos, o como quiera llamarse a las divisiones antiguas de sistemas, estos son sólo sub-sistemas de un único sistema de información y una única base de datos desde la que parten y a la que llegan todos los datos necesarios.

1.7 Informatización de organizaciones

El título de este texto tal vez parezca algo extraño al lector. Posiblemente sienta curiosidad por determinar qué quiere decir exactamente "informatización" o qué tiene que ver ello con las organizaciones y, más allá, con un producto comercial de *software* de base de datos como *Microsoft Access*.

Seguramente podrá preguntarse el lector por qué no simplemente "bases de datos", o "sistemas de información" u "organizaciones", por ejemplo.

La intención ha sido enfatizar en que el problema del manejo, uso y aprovechamiento de la información en una organización pasa en nuestros tiempos por estos cuatro conceptos: sistemas, organización, informatización y bases de datos.

No es frecuente encontrar en la literatura el enfoque integrador que pretenden dejar estas líneas, sin embargo, la experiencia de quien escribe indica sin lugar a dudas que, lo que comúnmente se conoce como programación de *software*, desarrollo de sistemas de información, bases de datos y demás disciplinas relacionadas, de carácter eminentemente técnico, poco solucionan sin el estudio de las organizaciones o, si se prefiere, de los sistemas que les enmarcan.

Todo sistema de información pasa primera y primordialmente por el estudio de la organización que le utilizará, y debería culminar necesariamente con la descripción del impacto que dicho sistema produce en ella. Por eso se enfatiza la informatización de organizaciones

en lugar de implementación de sistemas de información, bases de datos u otros. De esta manera se resalta a la organización como el sujeto de estudio y no al sistema de información contenido en ella o pretendido para ella, aun cuando este último sea el producto tangible deseado.

Una organización es primordialmente un sistema. Más específicamente un sistema humano y social. Por supuesto, existen organizaciones de muy diversa naturaleza; sin embargo, las que interesan aquí, las que hacen relevante el tema de la información para la toma de decisiones, son aquellas que se integran con seres humanos (y no máquinas mecánicas o computadores, por ejemplo).

Una organización, entonces, es un sistema en el que un equipo de personas, delimitando funciones tanto individuales como colectivas, con ánimo de trascendencia en el tiempo, hacen causa común para el logro de propósitos preestablecidos.

Un grupo de trabajadores que se organiza para discutir con sus patronos y mejorar las condiciones salariales y de trabajo con que cuenta es una organización, en este caso una organización sindical. Un grupo de individuos que se organiza para producir un bien que luego llevará al mercado y por el que espera un beneficio económico a cambio, es también una organización, ahora una empresa con fines de lucro. Un grupo de profesores que se organiza para impartir las asignaturas que componen el programa de estudios de una carrera universitaria, conforma también una organización, por ejemplo, llamada departamento docente.

Una condición necesaria pero no suficiente para que un grupo de individuos conforme una organización, es precisamente, como la palabra sugiere, que se organicen. Condiciones necesarias y suficientes son que, además de organizarse, el grupo de individuos pretenda objetivos comunes y procure la sostenibilidad y perdurabilidad del equipo en el tiempo.

Como cabe suponer, para alcanzar un mismo propósito seguramente existen múltiples formas de organizar un equipo de individuos. Precisamente por eso, las organizaciones son cada vez más, en el mundo contemporáneo, materia de estudio y reflexión profesional.

La especialización y la división del trabajo junto con el sistema de producción preponderante, surgidos a partir de la revolución industrial, son los aspectos conceptuales o filosóficos que más han influido en la forma que adoptan las organizaciones modernas. Tratándose de sistemas humanos, es precisamente la participación humana

en sus diversas formas, desde lo manual a lo intelectual, el factor clave en toda organización.

Una vez claros los propósitos, los procesos que deben ejecutarse y las tareas que deben cumplirse para el logro de estos propósitos en la forma más expedita y menos costosa posible, sugieren en buena medida la estructura que adoptará la organización.

La estructura de la organización, explícita generalmente en un diagrama especial llamado "organigrama", desempeña un papel relevante para la comprensión del sistema, a lo interno. No obstante, son los individuos que ocupan las posiciones descritas en el organigrama los verdaderos sujetos de interés, pues son ellos quienes finalmente llevarán mejor o peor cada responsabilidad asignada. Este hecho, por cierto, es olvidado con frecuencia en el análisis organizacional, lo que conduce a graves equivocaciones.

1.7.1 La misión, la visión

La organización vista desde la óptica de los sistemas necesita explicitar su papel y propósito social. Todos en la organización deben comprender y compartir los propósitos que persiguen como colectivo, en primer término, para luego entender el papel individual que cumple cada uno para el logro de estos propósitos. Igual aplica para el analista de sistemas que estudia la organización. Está claramente demostrado, a partir de las ideas de calidad total y mejoramiento continuo surgidas en el Japón de la posguerra, que la participación de todos los integrantes de la organización en la definición de sus propósitos, contribuye de manera sustantiva a crear la *sinergia* necesaria para su logro.

Así pues, hacer explícitas la misión que se asigna a la organización y la visión que de ella se tiene como colectivo, tanto en el presente como para el futuro, son tareas mucho más que declarativas. No sólo cumplen el objetivo, loable sin duda, de documentar o plasmar por escrito estas importantes características, sino que, y lo que es más importante, el proceso que conduce a estas definiciones a partir de la reflexión de los propios integrantes de la organización a todos los niveles, debe producir el consenso requerido y hacer llegar la información mínima necesaria para que todos en el equipo se sientan y actúen formando

parte del sistema y comprendan la significación de su papel en la materialización de tales ideas.

La misión de una organización es la definición clara y precisa de su propósito real, fáctico, incluyendo adjetivos que indiquen el grado de calidad que se espera en dicho propósito. La visión, en cambio, es la definición de la organización desde el punto de vista de lo deseable, de lo utópico, de lo supremo. La visión representa el ideal hacia el que debe ir la organización según sus integrantes, representa el reto permanente, difícil de alcanzar, pero que debe guiar los pasos de quienes la integran.

La misión es el ahora, lo real; la visión es el mañana, lo posible, lo deseable.

Veamos a manera de ejemplo cómo quedaron redactadas la misión y visión de una organización muy particular, la Dirección de Servicios de Información Administrativa de la Universidad de Los Andes (4), dedicada al quehacer informático:

Misión

Proporcionar a la Administración Universitaria, desde los niveles operativos hasta los gerenciales, coordinadamente y con unidad de criterios, las herramientas técnicas, servicios y productos necesarios para la organización del trabajo en ambientes con alto control automatizado de la información.

Visión

La Dirección de Servicios de Información Administrativa es una estructura integrada, actualizada, con alto valor técnico y ético, creativa, dinámica, comprometida, responsable, efectiva y eficiente, en todo lo que se relaciona con el desarrollo y/o implantación de herramientas organizacionales y teleinformáticas útiles para capturar, mantener, controlar y producir información administrativa que contribuya a facilitar la ejecución de los procesos y la toma de decisiones en distintos niveles de la Universidad de Los Andes. Asimismo, tiene como preocupación de primer orden, satisfacer las necesidades de los usuarios como parte integrante de los desarrollos que se emprendan y focalizar su atención hacia el logro de la independencia entre éstos y los diseñadores, como meta de calidad de alto valor estratégico para la institución.

Nótese que la misión es concreta, precisa, mientras que la visión es amplia, podría decirse incluso, utópica. La combinación de ambas entrega en pocas líneas una idea excelente del quehacer de la organización y sirve como punto de partida al analista de sistemas que se interna en su *praxis*.

En cualquier caso, siempre es preferible echar mano a estas definiciones para comenzar todo trabajo de sistemas, pero de no contarse con ellas, bien sería buena idea proponer su redacción a la organización o bien construirlas someramente a partir de los juicios que pueda formarse el analista, si lo primero no es posible.

1.7.2 Tipología desde la óptica de los sistemas de información

Desde el punto de vista de los sistemas de información, las organizaciones podrían tipificarse como sigue:

1. No informatizadas: ninguno de sus procesos se encuentra recibiendo apoyo de sistemas de información, las cosas se hacen manualmente, existen muy pocos o ningún documento sobre su estructura organigramática, procesos, normas, procedimientos.
2. Medianamente informatizadas. Algunas áreas de la organización reciben apoyo vía sistemas de información o simplemente aplicaciones computarizadas de propósito general, pero existen áreas que no reciben ningún tipo de apoyo informático. La documentación de sus procesos es escasa y las relaciones en términos de información entre sus departamentos o unidades, son incompletas e insatisfactorias.
3. Altamente informatizadas: Se cuenta con un sistema de información que integra todas las áreas de la organización, que se entienden entre sí apropiadamente. Hay documentación sobre su quehacer y las decisiones se toman ágilmente con base, al menos en parte, en lo que arroja su sistema.

Dependiendo del tipo de organización frente a la que esté el analista, variará su proceder en el intento de emprender proyectos de desarrollo de SI.

Frente a organizaciones no informatizadas, el principal problema será vencer el desinterés que posiblemente encontrará por estos temas. Sin embargo, si se da el caso en que la organización esté convencida de la necesidad de incluir SI, resultan éstas los tipos más atractivos para el desarrollo de sistemas. En este caso, el analista opera en terreno fértil, tiene total libertad creativa y le resultará muy fácil producir mejoras visibles en la organización, pues compite contra la labor meramente manual.

Frente a organizaciones medianamente informatizadas, el analista tendrá el mayor número de problemas. Una organización que no ha completado sus desarrollos informáticos o que cuenta con algunos precarios e incompletos, no es precisamente una organización que confíe en la informática como disciplina que pueda apoyarle. Sin lugar a dudas, este tipo de organizaciones consideran importante la informática, pero como ésta no les satisface, han perdido buena parte de la confianza que podrían haber tenido en la tecnología. Ganar otra vez su confianza para que estén en disposición de invertir nuevamente en esta materia, suele ser asunto difícil. Por otro lado, cuando de sistemas se trata, es muy probable que este tipo de organizaciones antepongan el factor económico ante cualquier otro. Seguramente son organizaciones inclinadas a adquirir productos informáticos de bajo costo y de propósito general, sacrificando la integración y el flujo constante de información como valor agregado. Si el analista logra convencer con su proyecto de sistemas, deberá tener especial cuidado en hacer el trabajo lo mejor posible, en el menor tiempo posible y al más bajo costo posible para recuperar la confianza de la organización.

Frente a organizaciones altamente informatizadas, el punto focal cambia notablemente. Ahora ya no es necesario hacer entender a la organización las bondades de los SI, pues ya las conoce y utiliza. El punto entonces se transforma en la búsqueda de mejoras incrementales a lo establecido. Aquí, la organización, generalmente está abierta a nuevos retos, siempre con el cuidado de no estropear lo que ya funciona bien. En estas organizaciones, un cambio completo de sus sistemas se avizora improbable, por lo cual la aproximación correcta debe ser la mejora de aspectos puntuales aprovechando la experiencia pasada y ofreciendo como paradigma la actualización tecnológica.

En los tres casos es de suma importancia que el analista de sistemas actúe apegado a la ética profesional. Bajo ninguna circunstancia debe anteponer su particular interés económico por el proyecto frente

al interés de la organización que espera servir. Hacer lo contrario puede redituarle beneficios inmediatos, pero que verá mermados en el mediano y largo plazo.

La informática es muy útil, quién lo duda, pero tampoco es una panacea. No es cierto que la simple inclusión de sistemas de información mejore la situación de una organización, en especial cuando ésta en realidad no lo necesita o su escala económica no es suficiente para hacer frente a los costos que la tecnología le acarrearía. Un analista de sistemas debe actuar en primer lugar como un profesional, y debe, en consecuencia, asesorar apropiadamente a sus usuarios, aunque a veces ello implique desechar el proyecto de sistemas de información sobre el que tiene interés particular. Una secretaria que recibe o transcribe 10 cartas por mes y que tiene un archivero convencional, tal vez no necesite un avanzado sistema de información de correspondencia que cueste ingentes sumas de dinero a la empresa cuando con las herramientas actuales se las arregla perfectamente. Pues bien, un analista, actuando éticamente, debe ser capaz de decir esto sin complejos y buscar otras áreas de la organización donde puede brindar una contribución verdadera.

Si un sistema funciona muy bien, aún cuando el sistema no haya sido elaborado por el analista, éste debe reconocerlo sin egoísmo ni prejuicios. El no hacerlo, el tratar de imponer el propio por razones meramente económicas, siempre terminará en fracaso. Puede ser que el nuevo sistema reporte ganancias a su realizador, pero al no aportar nada nuevo y al llevar a la organización a procesos de migración tecnológica costosos e inútiles, dará al traste con la opinión que de él se tenga, haciendo así mella en su reputación profesional. Mantener una postura ética no es sólo un asunto de decoro, sino también de dividendos, más allá de lo que algunos logran imaginar.

Otro problema de relevancia que enfrenta el analista de sistemas es la postura inmediatista de la gerencia frente al desarrollo de SI. Especialmente en nuestro medio, son frecuentes las situaciones en las que los gerentes opinan que el desarrollo de sistemas es tarea sencilla y, por tanto, puede ser hecha en corto tiempo o con inversiones irrisorias. En muchos casos no están dispuestos a ceder parte de su tiempo para las tareas de desarrollo, pero exigen que éstas se completen en tiempos muy ajustados e incluso imposibles. Quienes así opinan, seguramente han dejado pasar mucho tiempo sin emprender la informatización como norte en su ámbito de acción y se encuentran desesperados por

contar inmediatamente con ella, por regla general, debido a presiones externas más que propias. También es frecuente encontrar entre ellos personas que creen que la adquisición de computadores o la instalación de redes telemáticas son sinónimo de informatización o son capaces, por sí solas, de mejorar la situación organizacional.

El analista debe lidiar con este problema de la forma más profesional que le sea posible, explicando con todo detalle los diversos pasos que deben darse para completar un desarrollo exitoso de SI. Debe exponer de manera clara los inconvenientes que puede traer establecer tiempos o costos irreales y explicar que las computadoras y las redes son sólo la infraestructura necesaria para que la informatización sea viable, pero que nada de ello por sí mismo obrará el milagro si no hay sistemas de información que se instalen sobre dicha infraestructura y sean operados, de acuerdo con procedimientos establecidos, por usuarios entrenados, y su producto, es decir, la información, utilizada efectivamente en el proceso de toma de decisiones.

Por supuesto, el analista debe procurar que tanto los costos como los tiempos sean reales, factibles, los mínimos posibles. Sobre ello pueden "negociarse" salidas intermedias, siempre y cuando no traspasen la línea en la que el desarrollo corra peligro por la imposibilidad de cumplir razonablemente con el tiempo o las inversiones previstas inicialmente. En casos extremos, al enfrentar estas situaciones será mejor dar marcha atrás antes de comprometerse en un proyecto cuyo riesgo de fracasar sea demasiado alto. Es mejor decir que un verdadero profesional no es capaz de hacerlo en tan corto tiempo o a tan bajo costo que emprenderlo a sabiendas de que no culminará exitosamente.

1.8 Los procesos en la organización

Como hemos dicho antes, toda organización es un sistema y, como tal, tiene un propósito (o varios) que dan sentido a su existencia. Las organizaciones que nos interesan aquí, son aquellas cuyo propósito está lo suficientemente claro como para que la idea de sistemas de información pueda ayudarles o apoyarles en su labor. Para determinar esta claridad de propósitos es necesario comprender bien el quehacer de la organización y, para ello, la idea de procesos es de primera importancia.

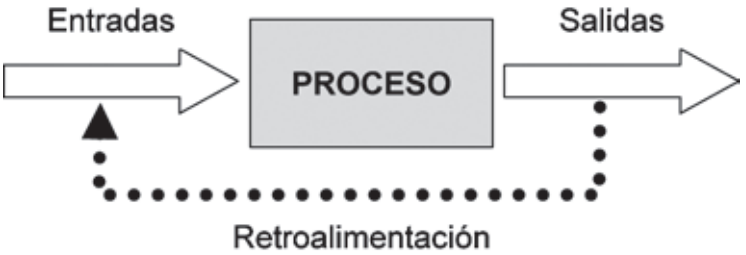


Figura 8. Esquema "Entradas-Proceso+Salidas" de los sistemas

Un proceso, como muestra la figura 8, es el nombre con que se puede caracterizar una actividad compleja, de alto nivel, que toma unas entradas y las transforma convirtiéndolas en unas salidas. En el camino puede o no haber retroalimentación, la cual simplemente es la información que aportan las propias salidas al proceso.

Por ejemplo, supóngase una organización merideña dedicada a la fabricación de medicamentos, llamada PROULA–Medicamentos (5).

Su principal propósito es producir medicamentos genéricos para la población venezolana a partir de insumos farmacológicos adquiridos y transformados mediante fórmulas farmacéuticas previamente desarrolladas. Para ello cuenta con inventarios de insumos, inventarios de productos terminados y, por supuesto, con laboratorios en los que se producen los fármacos. Claramente podemos describir su principal actividad, la fabricación de un medicamento, en términos de procesos.

En el proceso productivo, el objetivo es elaborar un medicamento, las entradas son los insumos, excipientes y principios activos farmacológicos con los que se elabora tal medicamento. Las salidas son los medicamentos producidos a partir de la combinación apropiada de las entradas, y el proceso en sí mismo es la labor humana y mecanizada requerida para transformar los insumos, mediante la fórmula farmacéutica, en medicamentos aptos para consumo humano. Una de las muchas líneas de retroalimentación que de hecho tiene este proceso en PROULA–Medicamentos, está constituida por el control de calidad que recibe cada fármaco producido. En efecto, al término del proceso se estudia la calidad de lo producido en cuanto al cumplimiento de las especificaciones farmacológicas, y a partir de esta información se ajusta el proceso para mejorar los resultados en caso que sea necesario.

Nótese que la idea de visualizar las actividades a nivel macro de una organización, en términos de procesos, es en realidad la construcción de un modelo sistémico de tales actividades utilizando una "plantilla" que los ve en términos de entradas, proceso y salidas y, posiblemente, retroalimentación.

Todo proceso, como modelo de la realidad que es, como sistema que es, no necesita ser descrito en profundo detalle; generalmente es suficiente describirlo someramente sin obviar lo esencial. Asimismo, todo proceso se refiere a una actividad en un nivel macro, por lo cual en la mayoría de los casos será posible identificar sub-procesos internos contenidos y super-procesos externos que lo contienen, exactamente de la misma forma como en todo sistema existen sub-sistemas y supra-sistemas.

Así pues, la tarea de describir los procesos de una organización es siempre de primer orden en todo estudio de sistemas, y en particular en cualquier estudio de sistemas de información. Encontrar los procesos, sus sub-procesos contenidos recursivamente hasta tocar procedimientos y tareas que no puedan ser descompuestos más allá, resulta en un modelo completo del quehacer de la organización que, obviamente, incluirá los flujos de información y datos que le son propios.

A partir de este mapa, el analista de sistemas tendrá una idea clara de lo que su sistema de información debe hacer para apoyar a la organización en su labor, así como de las prioridades y relaciones entre diversas actividades que deben abarcarse. Todo ello se compila en un documento del ciclo ADDI que se denomina *manual de procesos*.

En el menor nivel de complejidad de los procesos, esto es, cuando se les ha descompuesto hasta el punto en que no se encuentran más sub-procesos (no triviales), se está en presencia de un procedimiento. Llegado este punto, el procedimiento es entonces descrito como si fuera una secuencia de pasos que deben seguirse para completar una actividad de baja complejidad o precisamente determinada. Todo manual de procesos incluye en sus páginas, desde luego, el inventario de procedimientos asociados con los niveles de menor complejidad de cada proceso.

Resulta muy útil presentar un diagrama de procesos (también de procedimientos) que incluya información sobre las unidades, departamentos o dependencias responsables de llevar a efecto tales procesos y

procedimientos, especialmente cuando las salidas de unos pueden ser las entradas de otros.

Un diagrama de este tipo se constituye en una gráfica de dos entradas en la que, por ejemplo, verticalmente se ubican las dependencias que ejecutan el trabajo, horizontalmente se ubican los procesos o procedimientos y en su interior se describen las responsabilidades de cada unidad, conectando las actividades mediante líneas o flechas.

En general, todo este material debe ser presentado y discutido con la organización hasta el punto en que reciba aprobación formal, como indicación que es una síntesis apropiada de su quehacer. A partir de él se conforman las primeras ideas concretas de lo que tiene que lograrse con el sistema de información propuesto.

Bibliografía

(1) Bertalanffy, L. (1976). *Teoría general de los sistemas*. 1ª Edición. Fondo de Cultura Económica. Madrid, España.
(2) Checkland, P. (1993). *Pensamiento de sistemas. Práctica de sistemas*. Primera edición. Grupo Noriega Editores. México.
(3) Middleton, P. (1999). *Managing information system development in bureaucracies. Information and software technology*. Nº 41. Elsevier Science, p 473-482.
(4) Ponsot, E. et al. (2001). *Organización de los servicios de información administrativa en la Universidad de Los Andes*. 1ª Edición. Consejo de Publicaciones de la ULA, Mérida, Venezuela.
(5) Ponsot, E. (1998). *Sistema automatizado de Registro, Control y Análisis de Inventarios (SARCAI) para PROULA, Planta de medicamentos. V.2.0*. 1ª Edición. Universidad de Los Andes (sin editar). Mérida, Venezuela.

capítulo 2

Diseño de bases de datos relacionales

2.1 Arquitectura de bases de datos

Una base de datos es simplemente un contenedor especializado de datos que atiende a ciertas reglas preestablecidas para cumplir su propósito de la mejor manera posible. Conceptualmente se pueden identificar capas o niveles enmarcados en lo que se conoce como su "arquitectura". La figura 9 muestra estos niveles.

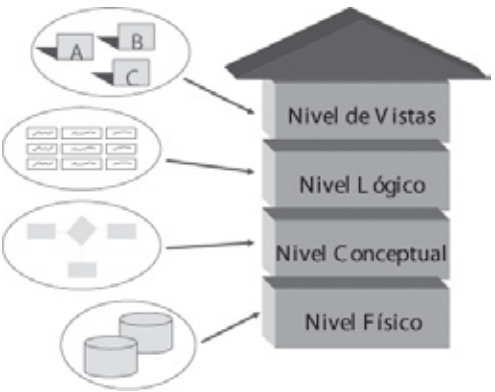


Figura 9. Arquitectura de las bases de datos

El nivel físico de la arquitectura, el primero de abajo hacia arriba en la figura 9, es el responsable de entenderse con el medio de almacenamiento donde residen los datos. Consta de los algoritmos de bajo nivel y directivas que hacen posible comunicarse con los medios y darles órdenes convenientemente, bien sea pasando a través del sistema operativo del computador o directamente, para incorporar nuevos datos, recuperarlos o eliminarlos definitivamente.

Por lo general, la organización física de los datos no necesariamente tiene que corresponder a la organización que el usuario percibe de ellos. A menudo, por el contrario, es muy diferente, pues en este nivel lo importante es la eficiencia en el almacenamiento y la recuperación. Por eso precisamente, el nivel físico de una base de datos es materia de expertos programadores, capaces de codificar algoritmos muy eficientes para conducir discos duros, disquetes, unidades lecto-escritoras de CD's y cintas magnéticas, entre otros medios de almacenamiento.

Los niveles conceptual y lógico de la arquitectura, siguientes bloques en el edificio mostrado en la figura 9, son responsables de la definición de la base de datos tal como la concibe el diseñador. Para ello se sirven de modelos de datos en los cuales puede definirse con todo detalle cuáles de ellos deberán almacenarse en la base de datos y de qué forma se relacionan unos con otros. Los modelos de datos "*Entidad–Asociación*" y "*Relacional*", que se verán con detalle más adelante, pertenecen a estos niveles de la arquitectura.

El *pent-house* del edificio lo constituye el nivel de vistas. Toda base de datos puede ser "vista" con diversas ópticas. El contador de una organización la verá como una fuente para sus registros contables, el ingeniero responsable de la producción la verá como un registro de la operación de la organización, el analista de la nómina la verá como un contenedor de datos sobre el personal y su labor cotidiana, y así sucesivamente. Dar a cada quien la visión que necesita de la base de datos es la tarea del nivel de vistas. Construir los informes y las consultas a partir de los datos, relacionándolos de maneras diversas para satisfacer los requerimientos de información de cada uno de los integrantes de la organización, es su papel fundamental.

El surgimiento de sistemas manejadores de bases de datos hizo posible el diseño de bases de datos dejando a éstos los problemas inherentes al nivel físico, de manera que en nuestros tiempos, el diseñador

se concentra fundamentalmente en los últimos tres niveles, razón por la cuál sólo trataremos con ellos a lo largo de este texto.

2.2 El Sistema Manejador de Bases de Datos (SMBD)

Un *Sistema Manejador de Bases de Datos* (SMBD) es un conjunto de programas diseñados para crear y manipular eficientemente bases de datos (que responden a algún modelo de datos particular). La figura 10 muestra el lugar que ocupa el SMBD en un ambiente informatizado con enfoque de bases de datos.

Como muestra la figura 10, los usuarios del sistema de información se entienden bien sea con los aplicativos programados solamente, bien con ellos y con el SMBD, o bien con el SMBD y directamente con la base de datos. La primera es la situación más común, la segunda corresponde a una situación en la que se cuenta con usuarios avanzados, y la tercera situación comprende usuarios muy avanzados, que generalmente son los propios constructores del sistema de información.

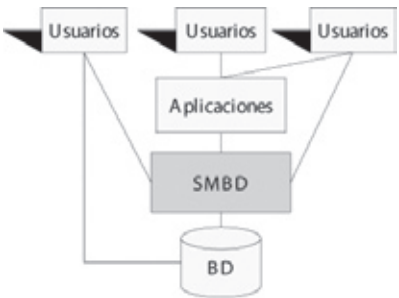


Figura 10. Lugar del sistema manejador de bases de datos

Por lo general, las aplicaciones se construyen para entenderse exclusivamente con el SMBD, el cual, a su vez, toma el control de la base de datos físicamente hablando.

Con el enfoque de bases de datos, considerando la existencia de un SMBD, se pretenden cuando menos las tres características siguientes:

1. *Independencia entre niveles de la arquitectura.* Los cuatro niveles de la arquitectura descrita en la figura 9 se procuran lo más independientes entre sí que sea posible. Esto significa que un cambio en uno de los niveles debe afectar lo mínimo e inevitable a los restantes. Por ejemplo, una mejora en el disco duro de la computadora (nivel físico) no debería ocasionar la modificación del esquema de la base de datos (nivel conceptual o lógico). Si así fuera, el trabajo de mantenimiento de los sistemas de información se haría muy cuesta arriba. Una nueva vista de la base de datos no tendría por qué implicar la modificación de las vistas existentes previamente o forzar un cambio en el disco duro de la computadora. Por otra parte, no siempre se logra este propósito, aunque es siempre una meta de alto valor agregado, por ejemplo, quisiéramos que la eliminación de un dato en la base de datos no afectara el nivel de vistas, sin embargo, inevitablemente afectará aquellas vistas que hagan uso del dato.
2. *Datos definidos aparte de los programas*. Uno de los factores que más afecta la independencia entre niveles de una base de datos, se produce cuando las definiciones de las estructuras de datos se encuentran inmersas en los programas que acceden a ellos. Versiones antiguas de lenguajes de programación como Pascal, Basic, C o Fortran, por citar algunos, operaban de esta forma, es decir, lo hacían de manera que los archivos contenedores de datos debían definirse y caracterizarse, junto con los algoritmos que hacían uso de ellos, dentro de los propios programas. La estandarización de las estructuras de datos (por ejemplo, las relacionales que tratamos en este texto), aunada a su creación independiente de los programas que accederán a ellas, es un gran éxito logrado gracias al surgimiento de los SMBD. En los antiguos compiladores, una modificación en la estructura de datos afectaba por lo general a todos los programas y hacía necesario alterar cientos y hasta miles de líneas de código fuente, con la consiguiente pérdida de tiempo y recursos que ello implicaba. En la actualidad, una sencilla modificación en la base de datos, posiblemente afectará a los programas que hagan uso del dato que sufrió alteraciones, pero no tiene por qué afectar a los demás. Brillante avance de la tecnología informática que, no obstante, ha sido puesto en discusión por la nue-

va orientación a objetos, a juicio del autor, equivocadamente, la cual pretende volver a encapsular datos y programas juntos.[2]

3. *Eliminación de duplicidad de datos innecesaria.* Aun cuando la idea de centralizar en un solo receptáculo todos los datos es el eje central del enfoque de bases de datos, no toda redundancia (sinónimo de duplicidad) es mala *per se*. La redundancia que resulta perniciosa en un sistema informático es la resultante de la proliferación incontrolada de datos. No resulta dañina aquella que duplica los datos para mejorar la seguridad o confiabilidad de los sistemas. Conociendo este hecho, el advenimiento de los SMBD ha logrado poner bajo control la redundancia de datos facilitando las tareas de mejora en la confiabilidad y disponibilidad de los sistemas, sin afectar la idea central "un dato, en un sólo lugar". Los nuevos SMBD gestionan automáticamente tareas como la distribución y replicación de datos hacia lugares geográficamente distantes, logrando así que los sistemas ganen confiabilidad y velocidad, manteniendo "memoria" de la fuente en que tales datos deben reposar por definición.

Dado que el SMBD implementa el nivel físico de la arquitectura, el diseñador tiene libertad de establecer los niveles conceptual, lógico y de vistas, para lo cual puede utilizar las siguientes herramientas que debe proporcionarle el SMBD:

- Un lenguaje de *definición* de datos (para implementar el nivel conceptual y lógico de la arquitectura). Se trata de un lenguaje de programación que permite crear, modificar o eliminar estructuras de datos, así como las relaciones que deban existir entre ellas.
- Un lenguaje de *manipulación* de datos (para resolver la recuperación de los datos y el nivel de vistas). Se trata de un lenguaje de programación para la elaboración de órdenes, cuyas respuestas son datos extraídos de las estructuras de datos definidas.

---

[2] Para ahondar en este interesante tema y ver una brillante defensa de la independencia entre datos y programas, puede consultarse el artículo de Darwen y Date titulado "Introducing ... The Third Manifesto" (1).

- Utilidades para el control de la edición y la concurrencia. La concurrencia es el fenómeno que surge cuando dos usuarios distintos tratan de acceder simultáneamente a un mismo dato, bien sea en la edición, consulta o eliminación. Asegurar que el acceso al dato no derive en una inconsistencia, es tarea del SMBD.
- Utilidades para supervisar la seguridad e integridad de los datos.
- Funciones para crear el diccionario de datos. El diccionario de datos es un repositorio, generalmente incluido en la propia base de datos, contentivo de la lista de todos los datos utilizados con su descripción semántica y sus características de tipos y restricciones o validaciones. Cuando las bases de datos crecen, resulta en una fuente de información de primera importancia.
- Herramientas para evaluar y mejorar el desempeño.
- Facilidades para crear secciones frontales de la base de datos (vistas o aplicaciones), o bien librerías que permitan utilizar algún lenguaje de programación para este propósito.

Además, concretamente, un SMBD debe facilitar:

1. Procesadores de lenguajes de consulta (como SQL[3] o QBE[4]) en los que se implementen los lenguajes de definición y manipulación de datos. Más adelante ahondaremos en el estándar de la actualidad para este tipo de lenguajes: SQL.
2. Generadores de informes o reportes impresos.
3. Herramientas para realizar respaldo de los datos.
4. Generadores de gráficos a partir de los datos y funciones básicas de estadística.
5. Generadores de aplicaciones (como 4GL[5], CASE[6] u otras).

---

[3] SQL: *Structured Query Language* o, en español, Lenguaje de Consulta Estructurado.

[4] QBE: *Query By Example* o, en español, Consulta vía Ejemplo.

[5] 4GL: por *Fourth Generation Language* o, en español, Lenguaje de Cuarta Generación.

[6] CASE: por *Computing Assisted Software Engineering* o, en español, Ingeniería del *Software* Asistida por Computadora.

2.3 Modelos de datos

Un *modelo de datos* es una abstracción sistémica de la realidad, considerando sólo el aspecto de datos y sus interacciones. Una organización necesita para funcionar distintos y variados datos. Éstos se listan y describen mnemotécnicamente en un modelo que plasma las relaciones entre ellos. Los modelos de datos pueden estar basados en:

- Objetos: Aquellos que utilizan objetos gráficos con un significado conocido para describir la realidad, tal como lo hacen los mapas de carretera con los símbolos. Generalmente se usan conectores para asociar dichos objetos. El más utilizado en la actualidad es el modelo Entidad-Asociación (o Entidad-Relación), que utiliza para modelar rectángulos, rombos y óvalos, con un significado sobreentendido.
- Registros: Aquellos que utilizan el concepto de registro[7] para establecer los datos presentes en el modelo. Como ejemplos se pueden citar los modelos *Jerárquico, de Redes y Relacional,* este último más ampliamente difundido en la actualidad y foco central en este texto.

2.4 Modelo Entidad–Asociación (E–A)

El modelo Entidad–Asociación (o Entidad–Relación) es un modelo de datos basado en objetos. Vio la luz por primera vez en el año 1976 de la mano de su autor, Peter Chen, en su memorable artículo intitulado "*The Entity – Relationship Model – Toward a Unified View of Data*" (2), y desde entonces hasta nuestros días ha sido ampliamente utilizado para diseñar bases de datos. Las ideas que se exponen en esta sección son en esencia una síntesis adaptada de los conceptos de Chen.

El modelo E–A es una herramienta del nivel conceptual cuya premisa es la visualización de los datos en términos de conjuntos de "entidades" y "asociaciones" (o "relaciones") entre ellos. Los conjuntos de en-

---

[7] Un registro es una estructura abstracta de datos que se representa como conjunto de campos, en general de tipo distinto, referidos siempre a un mismo individuo.

tidades son descritos listando la entidad abstracta que les representa, con sus "atributos" principales, y representados explícitamente como objetos que se conectan mediante líneas. De manera similar, las asociaciones son expuestas como elementos del diagrama que conectan entidades.

Una *entidad* es un objeto abstraído de la realidad, que existe, que es distinguible de otros objetos de la propia especie y de otras especies mediante la precisión de sus atributos elementales. Por ejemplo, una persona, un computador, una ciudad, un departamento de la empresa, un artículo del inventario, son entidades.

Se habla de "conjunto de entidades" en lugar de entidades a secas, ya que los elementos que se pretende representar en el diagrama son efectivamente colecciones o conjuntos de entidades y no las entidades individuales.

Una persona es claramente una entidad a la luz de la definición, pues podemos distinguirla de otra por sus características intrínsecas y podemos además identificar un objeto específico, digamos una silla, definitivamente fuera del conjunto de las personas. En otras palabras, podemos distinguir una persona de otra y distinguir una persona de entre los demás objetos que no son personas.

Un tanto más difícil es distinguir aquello que no es una entidad. La manera de hacerlo pasa por dilucidar si aquel objeto de nuestro interés tiene existencia propia, individual, por sí mismo, o no la tiene. Por ejemplo, el nombramiento de un empleado de la empresa como gerente de un departamento, existe, es real, tiene implicaciones verdaderas, puesto que dicho empleado asume responsabilidades concretas que afectan a la organización pero, desde el punto de vista de los datos, tal nombramiento sólo se materializa cuando se combina la persona nombrada con el departamento que le recibirá como gerente; de otro modo, este objeto denominado "gerente de" carece de sentido. Decimos entonces que tal cosa como un nombramiento no es una entidad. Puede ser una entidad débil (de la que hablaremos más adelante) o, con mayor seguridad, una asociación entre el conjunto de entidades "Empleados" y el de "Departamentos".

Una *asociación* es una conexión que se establece entre uno, dos o más conjuntos de entidades y se materializa disponiendo atributos distintivos de las entidades involucradas, como una unidad aparte. Por ejemplo, los empleados de la empresa se relacionan con sus departamentos, a través del concepto "Adscrito a", y esta rela-

ción se materializa en un nuevo elemento del modelo de datos que se forma a partir del código del empleado si éste le representa inequívocamente, y el código del departamento si éste hace lo propio con los departamentos.

Un *atributo* es una característica elemental o atómica de entidades y asociaciones. La cédula de identidad de una persona, el nombre del departamento de la empresa, el código del artículo del inventario, etc., son atributos.

Los atributos son los elementos que distinguen las entidades y las asociaciones en cuanto representan sus características interesantes para el modelo de datos. Por ejemplo, de una persona pueden interesar muchos atributos, tales como el color de sus ojos, su estatura, la nacionalidad de sus abuelos y los nombres de los profesores con quienes asistió a la universidad, entre otros tantos, sin embargo, para los efectos de la situación que hace a las personas interesantes en un sistema de nómina, tal vez con su número de identificación personal, apellidos y nombres, fecha de nacimiento y algunos otros datos sea suficiente. Luego parte importante del diseño de una base de datos es decidir qué atributos son interesantes o, mejor aún, podrían llegar a ser interesantes en el futuro en un conjunto de entidades o asociaciones. La figura 11 muestra un típico diagrama E–A.

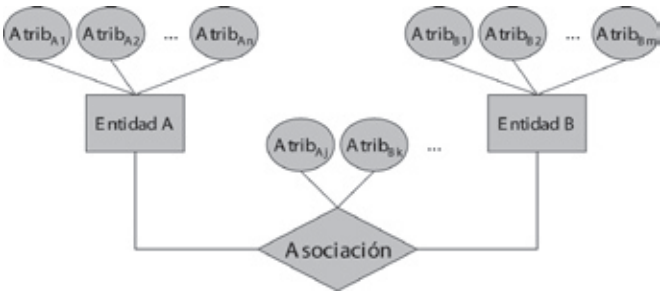


Figura 11. Diagrama E–A

Los conjuntos de entidades se representan mediante rectángulos cuyos rótulos son sus nombres, las asociaciones mediante rombos igualmente rotulados, y los atributos mediante círculos u óvalos también rotulados con el nombre que corresponde a cada uno.

Las *entidades débiles* son entidades en el sentido de que su existencia se debe a sus propios atributos, sin embargo están supeditadas o subordinadas a la existencia de otra entidad, llámesela fuerte. Se representan mediante rectángulos rotulados de doble línea, como muestra la figura 12. Por ejemplo, la transacción (compra o venta) de un artículo del inventario está supeditada a la existencia del mencionado artículo, sin embargo existe por sí misma cuando agrega atributos propios, como la fecha en que se transa, el monto global de la transacción o el número de la factura correspondiente.

Figura 12. Representación de entidades débiles

Por otra parte, en el modelo E–A es útil descubrir anticipadamente la *cardinalidad de mapeo* de las asociaciones entre entidades. Esta cardinalidad de mapeo es el tipo de asociación que se establece entre los conjuntos de entidades, considerando el número de entidades individuales que están involucradas (figura 13).

Figura 13. Ejemplos de la cardinalidad de mapeo de asociaciones

La cardinalidad de mapeo entre dos conjuntos de entidades (o de una asociación) puede ser "una a una", "una a varias", "varias a una" y "varias a varias". Cada tipo puede hacerse explícito poniendo símbolos (por ejemplo "1" e "∞") al lado de las líneas que unen a las entidades con la relación que las asocia, indicando el sentido de la cardinalidad.

En la figura 13 se muestran ejemplos de la cardinalidad de mapeo en situaciones comunes. El primer esquema de la figura representa una asociación del tipo "varias a una". La cardinalidad de mapeo de la relación entre empleado y departamento ("Adscrito A"), en este sentido es varias a una, ya que una entidad de empleado (un empleado) puede pertenecer a un solo departamento, sin embargo, un departamento cuenta con varios empleados. El segundo esquema de la figura representa una asociación del tipo "varias a varias". La asociación "Proveída Por", entre los conjuntos de entidades "Proveedor" y "Pieza" de una venta de repuestos, es varias a varias, ya que una pieza puede ser provista por varias entidades de "Proveedor" (proveedores) y varias piezas pueden ser provistas por un mismo proveedor. El tercer esquema de la figura representa una asociación del tipo "una a una". La asociación "Gerente Actual", establecida entre los empleados y las gerencias de una empresa, tiene cardinalidad de mapeo una a una, bajo el supuesto de que un empleado no puede ejercer más de una gerencia y una gerencia no puede ser ejercida por más de un empleado.

Un diagrama E–A, entonces, es un mapa de la base de datos diseñada que, independientemente del *hardware* o *software* que se empleará luego para materializar tal base de datos, muestra todos los elementos fundamentales que ésta contendrá y cómo se relacionan unos con otros.

2.5 Modelo relacional (R)

Cuando ha llegado la hora de materializar una base de datos en un computador, se requiere una herramienta del nivel lógico de la arquitectura, es decir, una herramienta que esté más cerca de las estructuras de datos reales que habrán de materializarse en la máquina. Específicamente, por su gran aplicabilidad y popularidad en la actualidad, escogimos para este texto el modelo relacional de bases de datos, como tal herramienta.

Recomendamos, no obstante, la realización del diagrama E–A como paso previo a su "traducción" en términos del modelo relacional, y aunque es perfectamente posible llevar a efecto el diseño de una base de datos directamente bajo los postulados del modelo R, no lo recomendamos.

Una base de datos *relacional* es aquella que utiliza el concepto **relación**, en términos de la teoría de conjuntos, como la única estructura lógica capaz de contener datos. Así, se le percibe como una colección de **relaciones** exclusivamente. En este contexto el término "relación", derivado de las matemáticas, es el producto cartesiano de conjuntos que forma un nuevo conjunto (la relación). El término preciso "relación" ha sido asimilado en la mayoría de los SMBD al vocablo "tabla" por su analogía con esta estructura formada por filas y columnas.

El modelo relacional de bases de datos nace en el año 1970 de la mano de su creador, E. F. Codd, cuando publica su artículo seminal intitulado "*A Relational Model of Data for Large Shared Data Banks*" (3). Desde entonces ha recibido aportaciones diversas y, sobre todo, un apoyo mayoritario de las grandes empresas de software en el mundo como IBM, Microsoft, Oracle o Sybase, por citar sólo algunas connotadas, que construyeron, aún construyen, soportan y comercializan SMBD íntegramente basados en el modelo R. En particular, un otrora miembro del equipo de IBM, C. J. Date, es uno de los autores que más aportes ha realizado sobre el tema. Su libro *Introducción a los sistemas de bases de datos* (4) es un verdadero tratado sobre las ideas originales de Codd, que recomendamos al lector ampliamente. Tal como la sección anterior fue una síntesis de las ideas de Chen, ésta lo es de las ideas de Codd y Date.

Para el modelo relacional, un *atributo* es una característica de interés del objeto en estudio, identificado con un nombre que expresa su significado. Por ejemplo, si el objeto en estudio es una persona, atributos son su nombre, apellido, fecha de nacimiento, etc.

Se entiende por *valor escalar* la menor unidad semántica de información (es decir, el dato más elemental que tiene significado). Es un valor atómico, ya que no se puede descomponer desde el punto de vista del modelo o, lo que es igual, una descomposición del valor escalar produciría que se perdiese su significado. Veamos un ejemplo: si se registra la ciudad en la que habita una persona, el dato "Caracas" es un valor escalar, ya que una nueva descomposición de éste ocasiona la pérdida de su significado. En este caso, si se descompone "Caracas" en los caracte-

res que forman la palabra, "c", "a", "r", etc. se pierde su connotación de ciudad (recordar las propiedades emergentes de los sistemas).

Un *dominio* es un conjunto de valores escalares, todos del mismo tipo, de entre los cuales toman sus valores los atributos. También se entiende un dominio como el conjunto de todos los valores escalares posibles que puede tomar un atributo. El concepto de dominio, en este caso, se asemeja al concepto de dominio de una variable desde el punto de vista matemático, ya que efectivamente, el concepto de atributo es equivalente al concepto de variable matemática, no necesariamente valorada en el campo de los números.

Se habla entonces de los atributos y sus dominios subyacentes, o bien de los dominios sobre los cuales se definen los atributos. Así, en el contexto del modelo R, es posible que dos o más atributos distintos se definan sobre el mismo dominio, sin embargo, no es posible que un atributo sea definido sobre dos o más dominios. Considérese por ejemplo el atributo "ciudad donde habita una persona" (o simplemente "ciudad"). Este atributo tiene un dominio subyacente que podríamos, como todo conjunto, explicitar por comprensión, esto es, D = {todas las cadenas de caracteres de longitud fija máxima k, que representan nombres de ciudades} o por extensión, es decir, D = {Caracas, Londres, Madrid, París, Miami, Mérida, Sevilla, Liverpool, ...}.

La definición de un dominio debe contener, además del tipo de dato que corresponde a sus valores escalares, su longitud y las restricciones o validaciones conceptuales que le son propias, para comprender el tipo de valor escalar al que se refiere.

El concepto de dominio permite establecer cuándo las comparaciones entre atributos tienen o no sentido. Si se quiere comparar la edad de una persona con la ciudad en la que habita en términos de "mayor que", por ejemplo, claramente se ve que tal comparación carece de sentido, pero la razón formal es que ambos atributos se definen sobre distintos dominios y, por consiguiente, no son comparables.

C. J. Date conceptúa la relación en los siguientes términos: "Una ***relación*** definida sobre un conjunto de dominios (no necesariamente todos distintos), es una estructura abstracta de datos compuesta por dos partes: una cabecera y un cuerpo.

La *cabecera* está formada por un conjunto (en general fijo) de pares atributo - dominio, de la forma $\{(A_1 : D_1),(A_2 : D_2),\cdots,(A_n : D_n)\}$.

Así, cada atributo *Aj* está asociado con un y sólo un dominio *Dj* (*j* = 1,2,...,*n*).

El *cuerpo* está formado por un conjunto (en general dinámico) de *tuplas*. Cada tupla está formada por un conjunto de pares atributo - valor, de la forma:

$$t_i = \{(A_1 : v_{i1}), (A_2 : v_{i2}), \cdots, (A_n : v_{in})\} \quad \text{, para } i = 1,2,\ldots,m$$

Una tupla cualquiera t_k, contiene valores escalares v_{kj} de los atributos A_j respectivamente; cada valor v_{kj} es un elemento tomado del dominio D_j sobre el cual se ha definido el atributo A_j ($j = 1,2,\ldots,n$ y $k = 1,2,\ldots,m$)".

El *grado* de la relación es *n* y, de acuerdo con él, las relaciones se clasifican en unarias, binarias, terciarias y en general *n*-arias. La *cardinalidad* de una relación es *m*.

Como se ve, la relación es un conjunto. Un conjunto formado con el producto cartesiano de los dominios sobre los que subyacen sus atributos. Toda relación puede representarse como una "tabla" de filas y columnas. Esta representación implica buscar una *instancia* de la relación (una fotografía de la relación, tomada en un instante fijo del tiempo), donde se listan tuplas ejemplo de ésta. Esta representación es explicativa, pero un tanto aparatosa, como se muestra en la figura 14.

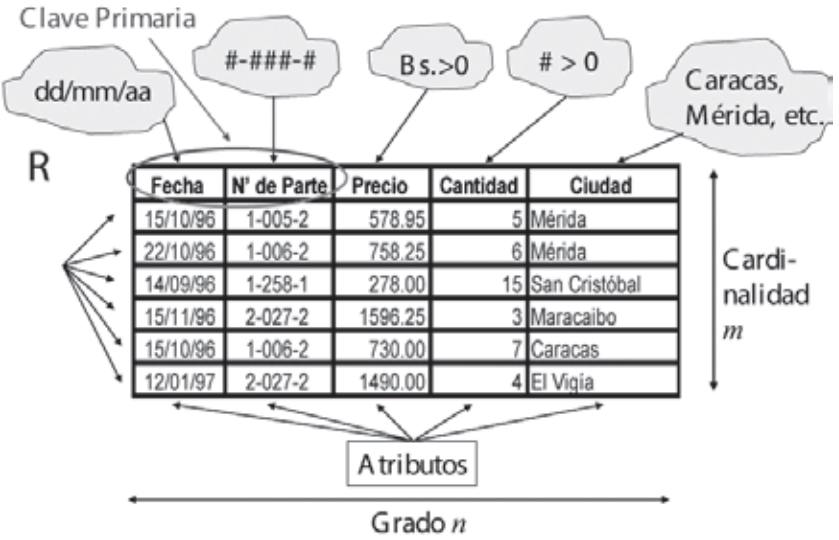


Figura 14. Ejemplo de una relación y sus elementos

La figura 14 ilustra cómo los pares atributo–dominio de la cabecera de la relación se consideran los encabezados de las columnas de la tabla, y las tuplas del cuerpo de la relación se consideran las filas de la tabla, en las que cada campo se corresponde con un valor específico del atributo señalado en la cabecera. El grado y la cardinalidad de una relación se señalan también en la figura.

Por último, la figura 14 muestra un ejemplo de lo que podría ser la clave primaria de una relación, a lo cual nos referiremos más adelante.

Otra representación de una relación que no muestra ejemplos de las tuplas se consigue al usar la notación de conjuntos y exponer únicamente los atributos de la cabecera, subrayando aquellos que forman la clave primaria. La relación *R* mostrada puede explicitarse de esta manera como:

R = {Fecha, Nº de parte, precio, cantidad, ciudad}

2.5.1 Claves de una relación

Intuitivamente, la palabra *clave* implica un identificador único para cada ítem de datos presente en la relación, y lo de *primaria* da la idea de principal o preponderante. Formalmente hay varias definiciones asociadas con este concepto:

Todo subconjunto *K* de los atributos de la cabecera de una relación R que satisface las propiedades de **unicidad** y **minimalidad,** se denomina una **clave candidata** de *R.*

Dado un subconjunto *K* de los atributos de la cabecera de una relación *R,* se dice que *K* satisface la propiedad de **unicidad**, si, y sólo si, en cualquier instante del tiempo no existen entre las tuplas del cuerpo de *R* valores iguales de *K.* En este caso, la igualdad se utiliza en el contexto de conjuntos (dos conjuntos son iguales, si, y sólo si, están formados por los mismos elementos).

Dado un subconjunto *K* de los atributos de la cabecera de una relación *R* que satisface la propiedad de unicidad, se dice que K cumple también la propiedad de **minimalidad**, si, y sólo si, no es posible desechar alguno(s) de los elementos de *K* sin que el nuevo subconjunto deje de satisfacer la propiedad de unicidad.

Claves candidatas son entonces los subconjuntos con el menor número de elementos de la cabecera de la relación, cuyos correspondientes valores en el cuerpo de la relación no se repiten. Esto es, el menor número de atributos que identifican de forma única a cualquier tupla de la relación. La *clave primaria*, en una relación, es sencillamente aquella clave candidata seleccionada por el diseñador de la base de datos como la que identificará a las tuplas de la relación. Las claves candidatas no seleccionadas se denominan claves alternativas y, en realidad, pierden interés frente a la clave primaria.

En vista de que una relación está compuesta por un "conjunto" de tuplas, se puede asegurar que toda relación tiene al menos una clave candidata y por lo tanto puede definirse una clave primaria (los conjuntos no contienen elementos repetidos, el cuerpo de la relación es un conjunto de tuplas, luego, las tuplas -elementos del conjunto- no pueden repetirse).

La escogencia de la clave primaria se realiza con "sentido común" y en general debe escogerse aquel subconjunto de los atributos que tenga el menor número de elementos o que intuitivamente represente mejor a las tuplas. La búsqueda de claves candidatas se fundamenta en los supuestos que realice el diseñador y no necesita ser exhaustiva. Puede encontrarse una (la primera) y decidirse que ésta será la clave primaria, con lo que se detiene el proceso.

Supóngase como ejemplo una relación que almacenará datos de los empleados de una empresa, de la forma siguiente:

Empleados = {C.I., Apellidos, Nombres, Dirección, Fecha de Nacimiento, Estado Civil}

Examinemos algunas claves posibles de *Empleados:* K_1 = {C.I.}, K_2 = {C.I., Apellidos}, K_3 = {Apellidos, Nombres}, K_4 = {Apellidos, Nombres, Dirección, Fecha de Nacimiento}. Reflexionemos ahora lo siguiente:

- K_1 es una clave candidata porque se supone que la cédula de identidad (C.I.) no se repite para ningún empleado, y si se elimina este atributo, el conjunto queda vacío (el cual, por razones obvias, no puede ser una clave candidata).
- K_2 cumple con el principio de *unicidad* (derivado de que C.I. es única), ya que ninguna combinación de C.I. y Apellidos se repetirá en la relación, pero no cumple con el principio de *minimalidad*, ya que existe un subconjunto de K_2 más pequeño (de hecho, K_1), que no viola el principio de *unicidad*.

- K_3 no puede ser una clave candidata, ya que no cumple con el principio de *unicidad*. Pueden encontrarse en algún momento dos empleados con iguales apellidos y nombres.
- K_4 sí puede ser una clave candidata, ya que puede suponerse que en ningún momento habrá dos empleados que tengan los mismos apellidos, nombres, dirección y fecha de nacimiento, y además, si se retira cualquiera de estos atributos, ya no se puede garantizar la *unicidad*.

Ahora bien, entre K_1 y K_4, dos claves candidatas de *Empleados*, nos quedamos con K_1 como la clave primaria. De esta forma, la relación *Empleados* queda determinada así:

Empleados = {C.I., Apellidos, Nombres, Dirección, Fecha de Nacimiento, Estado Civil}

Quedando por expresar los dominios asociados con cada atributo.

Existen otros tipos de claves sumamente importantes para el modelo relacional: *las claves ajenas*. Supóngase la siguiente sección de una base de datos de empleados:

Empleados = {C.I., Apellidos, Nombres, Código del Departamento}
Departamentos = {Código, Nombre, C.I. del Gerente}

El atributo "Código del Departamento" en *Empleados*, obviamente está definido sobre el mismo dominio que el atributo "Código" en *Departamentos*. Más aún, no se esperaría que existiese en *Empleados* un valor de "Código del Departamento" que no estuviese previamente en *Departamentos* (ya que esto implicaría la adscripción de un empleado a un departamento inexistente).

De la misma forma, el atributo "C.I. del Gerente" en *Departamentos* está definido sobre el mismo dominio que el atributo "C.I." en *Empleados* y no se esperaría que un departamento tuviese un gerente sin que éste estuviera registrado primero como empleado.

Los atributos "Código del Departamento" en *Empleados* y "C.I. del Gerente" en *Departamentos* representan respectivamente a las claves primarias de sus relaciones originales y se les conoce entonces como *claves ajenas*.

Por consiguiente, una **clave ajena** "en" una relación R_1, es cualquier subconjunto de atributos de R_1 que representen a una clave primaria de otra relación R_2.

En el ejemplo, "Código del Departamento" es una clave ajena en *Empleados*, ya que representa la clave primaria de *Departamentos*, y

"C.I. del Gerente" es una clave ajena en *Departamentos*, ya que representa la clave primaria de *Empleados*.

La clave ajena es en realidad la forma que tiene el modelo relacional de explicitar las asociaciones entre las distintas relaciones que lo componen. Como la clave primaria de una relación identifica de manera única a cada tupla de la relación, al conocer el valor de la clave primaria de una tupla se conocen inmediatamente los valores del resto de los atributos de la tupla (se conoce toda la tupla).

Conocidas las claves ajenas de todas las relaciones puede construirse entonces un diagrama que muestre las distintas asociaciones existentes en la base de datos. Este diagrama se conoce con el nombre de **diagrama referencial**.

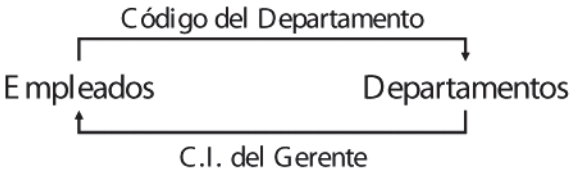


Figura 15. Diagrama referencial del esquema Empleados–Departamentos

La figura 15 muestra un sencillo diagrama referencial para las dos relaciones *Empleados* y *Departamentos*. Partiendo de *Empleados* hay una flecha que llega a *Departamentos*, que se rotula como "Código del Departamento". Esto quiere decir que en *Empleados* hay una clave ajena denominada "Código del Departamento" que *representa* en ella a *Departamentos*. Asimismo, partiendo de *Departamentos* hay una flecha que llega a Empleados y se rotula como "C.I. del Gerente", lo cual muestra que en *Departamentos* hay una clave ajena denominada "C.I. del Gerente" que *representa* en ella a *Empleados*.

Los diagramas referenciales son en extremo útiles para el diseño de bases de datos relacionales y, en general, por supuesto, alcanzan a ser mucho más extensos que el mostrado en la figura 15, aunque en esencia contienen la misma información.

2.5.2 Normalización

En la jerga de bases de datos relacionales se entiende por *normalización*, el proceso mediante el cual se transforma un conjunto de relaciones dadas en otro conjunto equivalente de relaciones "más deseable". En este sentido, la normalización es un proceso que se basa en el estudio crítico de todas y cada una de las relaciones incluidas en el modelo diseñado. Previamente a la discusión sobre normalización, son necesarias algunas definiciones importantes; veamos:

Dos atributos *A* y *B* de una relación *R* (simples o compuestos) son tales que *B* es **dependiente funcional** de *A* si la ocurrencia de cualquier valor de *A* en *R* implica necesariamente la ocurrencia del mismo valor de *B* en *R*. Se denota $A \Rightarrow B$.

Dados dos atributos *A* (en general compuesto) y *B* de una relación *R,* tales que $A \Rightarrow B$, se dice que esta es una **dependencia funcional completa**, si *B* no es dependiente funcional de ningún subconjunto de *A* de cardinalidad menor.

Luego si *B* es dependiente funcional de *A* y *A* es un atributo simple, entonces necesariamente *B* es dependiente funcional completo de *A*. Si *A* es la clave primaria de la relación, entonces, por definición, todos los atributos de la relación que no forman parte de *A* son dependientes funcionales de *A*.

Nótese que la dependencia funcional se establece en una sola dirección; sin embargo, entre dos claves candidatas de una relación cuya intersección es vacía, hay una dependencia funcional en ambas direcciones.

Dependencias funcionales transitivas. Dados tres atributos *A*, *B* y *C* de la cabecera de una relación *R*, si $A \Rightarrow B$ y $B \Rightarrow C$, entonces $A \Rightarrow C$. En palabras, si *B* es dependiente funcional de *A* y *C* es dependiente funcional de *B*, entonces resulta que *C* es dependiente funcional de *A*. A este hecho le denominamos dependencia funcional transitiva.

Definidos los conceptos de dependencias funcionales, veamos ahora las formas normales más importantes. Hay más de tres formas normales, sin embargo, las formas normales superiores a la tercera están fuera del alcance de este texto.

- *1ª Forma Normal (1FN):* Una relación está en 1FN si todos los dominios sobre los que se definen sus atributos están formados por valores atómicos (escalares que no admiten descomposición). Es decir, si cumple estrictamente con la definición de "relación".

- *2ª Forma Normal (2FN):* Una relación está en 2FN si está en 1FN, y los atributos que no forman parte de la clave primaria son dependientes funcionales *completos* de ella.
- *3ª Forma Normal (3FN):* Una relación está en 3FN si está en 2FN, y *no* existen dependencias funcionales transitivas.

Como ejemplo, supóngase la relación *Libros* definida a continuación:

Libros = {Código del Libro, Nombre del Libro, Nombres de los Autores, Tema Clave, Descripción del Tema, Ubicación={Estante, Repisa}, Proveedor del Libro, Precio del Libro}

Esta relación no está en 1ª Forma Normal, ya que contiene atributos definidos sobre dominios cuyos elementos no son escalares. Tal es el caso de los atributos "Nombres de los Autores" y "Ubicación", que en realidad se definen sobre una lista de valores.

Para resolver este problema, la primera tentación es sustituir el plural "Nombres de los Autores" por el singular "Nombre del Autor". Sin embargo, cuando se incluyó el plural "Nombres de los Autores" en la relación original, lo que seguramente se pensó fue que un libro podría tener más de un autor. Luego, la transformación propuesta ocasionaría, bien la pérdida de información (porque se tendrían que restringir los autores a uno sólo), o bien la redundancia excesiva (pues de no restringirse el número de autores, habría que repetir toda la información del libro para cada autor -componiendo una tupla por autor-).

Otra tentación es eliminar el atributo "Ubicación", poniendo en su lugar dos atributos, "Estante" y "Repisa". No obstante, cuando se incluyó el atributo "Ubicación", la idea fue identificar el par Estante-Repisa como una sola característica. Hay redundancia excesiva en la solución tentativa, puesto que muchos libros pueden compartir el mismo estante y repisa, y deberá repetirse esta información (dos atributos) demasiadas veces.

Así, una mejor solución a este problema pasa por una expansión del número de relaciones del esquema de la base de datos, que ahora quedaría como:

Autores = {Código del Autor, Nombre del Autor}
Ubicaciones = {Código de la Ubicación, Estante, Repisa}
Autores de Libros = {Código del Libro, Nombre del Libro, Código del Autor}

Libros = {Código del Libro, Nombre del Libro, Tema Clave, Descripción del Tema, Proveedor del Libro, Precio del Libro, Ubicación}

Las nuevas relaciones "más deseables" están en 1FN; sin embargo, la relación *Libros* no está en 2ª Forma Normal, pues los atributos que no forman parte de la clave primaria no son dependientes funcionales completos de ella. En efecto, "Tema Clave", "Descripción del Tema", "Proveedor del Libro", "Precio del Libro" y "Ubicación" son dependientes funcionales de {Código del Libro, Nombre del Libro}, pero también lo son de {Código del Libro}. La solución es ahora más sencilla. Basta que el atributo "Nombre del Libro" no forme parte de la clave primaria de la relación (se cometió un error en la escogencia de la clave). Para subsanar este dilema, el nuevo conjunto de relaciones es entonces:

Autores = {Código del Autor, Nombre del Autor}

Ubicaciones = {Código de la Ubicación, Estante, Repisa}

Autores de Libros = {Código del Libro, Código del Autor}

Libros = {Código del Libro, Nombre del Libro, Tema Clave, Descripción del Tema, Proveedor del Libro, Precio del Libro, Ubicación}

Ahora, las relaciones están en 2FN, sin embargo, *Libros* no está aún en 3ª Forma Normal. Existen atributos en *Libros* que son dependientes de la clave primaria, de forma transitiva. El atributo "Descripción del Tema" es directamente dependiente funcional de "Tema Clave", y como éste lo es de la clave primaria, "Descripción del Tema" termina siendo dependiente funcional, de manera transitiva, de la clave primaria (esto es, sin ser realmente dependiente funcional de ella). Algo similar ocurre con "Precio del Libro" y "Proveedor". La solución definitiva hasta la 3ª Forma Normal, es entonces:

Autores = {Código del Autor, Nombre del Autor}

Ubicaciones = {Código de la Ubicación, Estante, Repisa}

Autores de Libros = {Código del Libro, Código del Autor}

Libros = {Código del Libro, Nombre del Libro, Tema Clave, Código de la Ubicación}

Temas = {Tema Clave, Descripción del Tema}

Proveedores de Libros = {Código del Libro, Proveedor, Precio del Libro}

Todos estos elementos, integrantes del modelo relacional, deben ser decididos antes de llevar la base de datos al computador. El orden sugerido para tales efectos, es el siguiente:

1. Definir el sistema, hacer el análisis, documentar la organización y sus procesos, estudiar la factibilidad del proyecto, acordar un plan de informatización y los demás pasos preliminares de importancia capital mencionados al inicio del texto.
2. Construir el modelo E–A.
3. Traducir los elementos del modelo E – A, léase entidades, entidades débiles y asociaciones, con sus atributos y dominios subyacentes, al modelo R, en donde todo debe ser visto exclusivamente como relaciones.
4. En el modelo R, estudiar y decidir las claves primarias y ajenas de las relaciones.
5. Construir el diagrama referencial y revisar cuidadosamente el diseño hasta garantizar que se encuentra en 3FN cuando menos.
6. Llevar los elementos al computador y crear la base de datos utilizando para ello un SMBD relacional (en nuestro caso haremos esto con *Microsoft Access*).
7. Programar y probar todas las interfaces, informes, menús y demás elementos computacionales que funcionarán contra la base de datos creada.
8. Documentar lo producido con los manuales correspondientes e implantar la solución en la organización.

Bibliografía

(1) Darwen, H.; Date, C. (1995). *Introducing ... The third manifesto.* Database Programming & Design 1(8). Enero. EEUU, pp 25-35.

(2) Chen, Peter (1976). *The Entity-Relationship Model–Toward a Unified View of Data.* ACM Transactions on Database Systems, Vol. 1, Nº 1. Marzo. EEUU, pp 9-36.

(3) Codd, E. F. (1970). *A Relational Model of Data for Large Shared Data Banks.* Communications of the ACM. Vol. 13, Nº 6. Junio. EEUU, pp 377-387.

(4) Date, C. J. (1993). *Introducción a los Sistemas de Bases de Datos.* Vol. 1 de 2. 5ª Edición. Addison-Wesley Iberoamericana, S. A. Wilmington, Estados Unidos.

capítulo 3

Bases de datos y sistemas de información con Microsoft Access

3.1 Access como SMBD

Microsoft Access, en la versión que utilizamos en esta obra (2003), es un sistema manejador de bases de datos diseñado por la corporación *Microsoft* dentro del conjunto de programas aplicativos *Microsoft Office*. La suite *Office* comprende además un procesador de palabras (*Word*), una hoja de cálculo (*Excel*) y un diagramador de presentaciones (*Power Point*). La versión profesional del producto se vende con estos cuatro componentes y tanto su aceptación en el mercado de usuarios de computadoras personales como su utilidad práctica, son innegables. El número de sus seguidores ha crecido notablemente a lo largo de más de los 12 años de vida con que cuenta. Su precio moderado, sus excelentes características de "usabilidad" e integración y sus mínimas exigencias de hardware han sido la clave de la popularidad de estos aplicativos, que corren sobre el sistema operativo *Windows*, también de *Microsoft*.

Por su parte, *Access* es en toda la extensión de la frase un sistema manejador de bases de datos. Está dirigido a soportar sólo aplicaciones de pequeña y mediana escala (entendiéndose éstas como volúmenes de datos inferiores al Giga Byte y no más de 10 usuarios concurrentes) pero aun así contiene todas las características importantes para definirlo como tal.

Con *Access* se puede realizar un sistema de información completo que contenga una base de datos relacional, las interfaces deseadas, las consultas y reportes requeridos, los menús y demás características operacionales, los esquemas de usuarios, de seguridad y la configuración de red escogida. Estos son los ingredientes básicos de todo SI. Las condiciones son ejecutar el sistema operativo *Windows* sobre uno o varios computadores personales de arquitectura Intel Pentium (y prestaciones razonables), similares a las que muchos seguramente tendrán en su escritorio para labores de oficina.

Para invocar *Access* desde *Windows XP*, se localiza la opción homónima en el menú Inicio → Todos los programas, señalada en la figura 16.

Figura 16. Icono identificador del Access

Por supuesto, valen los atajos típicos de *Windows*, como pulsar un acceso directo creado en el escritorio o en una barra de herramientas y hasta ejecutar el archivo MSACCESS.EXE desde el directorio de instalación del producto (C:\Archivos de programa\Microsoft Office\Office11).

En lo sucesivo, y principalmente por razones de espacio, indicaremos una sola forma de hacer las cosas, no la más usual ni la más rápida, sino la que "siempre funciona", dejando al lector la sobreentendida tarea de curiosear las muchas otras formas abreviadas de hacer lo mismo, características de un entorno gráfico amigable como *Windows*.

Al invocar el SMBD se obtiene una ventana *Windows* convencional, que es el frontal del *Access* 2003, como muestra la figura 17.

La ventana principal del *Access* en la figura 17 representa el ambiente de trabajo en el que se desarrollan todas las tareas de construcción del SI. Cuando se abre la aplicación, se presenta una ventana interior denominada "Nuevo Archivo" [1]. Si esto no ocurriera o bien si dicha ventana se cerrara por alguna causa, se puede abrirla pulsando el primero de los íconos en la barra de menús principal, opción "Nuevo" [2].

La ventana "Nuevo archivo" es en realidad una ventana para la "navegación", de forma muy similar a como se hace en un explorador de Internet. Las opciones son efectivamente hipervínculos. Son varias las interesantes, quizá más de las que habitualmente son necesarias, como ocurre con la mayoría de los elementos de *Windows*. Nos concentraremos entonces en aquellas acciones inmediatamente necesarias para completar las tareas que nos interesan.

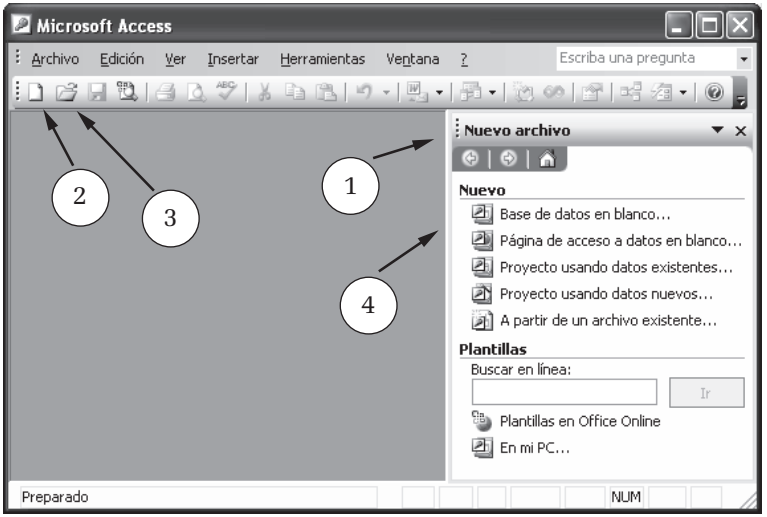


Figura 17. Ambiente inicial del Microsoft Access 2003

En el ambiente inicial (figura 17) se pueden hacer cosas como abrir un archivo de base de datos (BD) existente, que haya sido manipulado recientemente, dando clic sobre el icono de la carpeta [3], o bien puede crearse un nuevo archivo de BD, dando clic sobre la opción "Base de datos en blanco" [4].

3.2 Esquema de la base de datos

Como concluimos en secciones anteriores del texto, el resultado del análisis de sistemas y diseño de la base de datos es una idea clara y documentada de ambos elementos (SI y BD). En lo atinente a la BD, ésta, una vez diseñada, se resume en lo que se conoce como el esquema

de la base de datos. Un esquema es el conjunto que forman el diagrama E – A y el diagrama referencial, junto con los demás detalles sobre las relaciones, atributos, sus dominios, claves primarias y ajenas, propias del modelo R. Sólo después de que se tiene esta documentación se recurre al computador para plasmarlos en el medio físico, real.

En particular con *Access*, la materialización de una base de datos ocurre con la creación de las tablas (o relaciones) al interior de lo que se denomina un "archivo de base de datos". Todos los elementos del esquema de la BD quedan almacenados en un único archivo de computadora, especial para ser manipulado por el SMBD *Access* y que debe recibir un nombre apropiado, con la extensión ".mdb".

3.2.1 El archivo de la base de datos

Este es un archivo binario del sistema de archivos de la computadora, que se almacena en el disco duro. Se crea desde dentro del *Access* la primera vez y, una vez creado, puede ser objeto de un acceso directo sobre el cual, haciendo un doble clic se ordena al computador su apertura con *Access*. Existe una versión "*Run Time*"[8] de *Access*, sin costo adicional para el usuario, que se distribuye con la versión de desarrollo del *Office* y es utilizada por los programadores de sistemas para la distribución de sus productos terminados, si el usuario final no desea adquirir *Office*. Esta versión sólo cuenta con un subconjunto reducido de las características del *Access* que permite operar con el sistema programado, pero no modificarlo en forma alguna. Para poder alterar el sistema se necesita el SMBD entre los programas instalados en la computadora.

Un archivo de bases de datos contiene en realidad muchos más elementos, además de la propia base de datos. En él se incluyen también los fragmentos de código fuente que constituyen las consultas o vistas persistentes, los formularios o ventanas que conforman la interfaz que se proporciona al usuario del sistema, los informes o reportes creados para presentar la información, las macros y módulos escritos en lenguaje de programación (para el caso, Visual Basic para Aplica-

[8] En español, "tiempo de ejecución".

ciones) que aportan la lógica general del sistema, los menús y barras de herramientas que complementan la interfaz y demás detalles propios de un SI profesional.

Esta característica no niega la posibilidad de que el aplicativo sea utilizado con la filosofía Cliente-Servidor. De hecho, *Access* incluso proporciona utilidades para tal esquema de diseño, que permiten dividir un archivo de base de datos en dos, uno con la base de datos en si y otro con el resto de los elementos mencionados, ambos vinculados. También es posible (e incluso frecuente) que *Access* se utilice como el panel frontal de una aplicación, en la que la base de datos se implemente con otro SMBD, nativamente si se trata de *Microsoft SQL Server,* vía ODBC[9] en otro caso. Esta práctica es popular entre los analistas por la siguiente razón: *Access* proporciona herramientas poderosas y sencillas para el prototipado de sistemas y su posterior implantación. Una buena estrategia entonces es comenzar a pequeña escala programando todo en *Access*, planificar una primera etapa de crecimiento en la que la base de datos se separe del resto de las funciones del sistema, aun en *Access*, para finalmente pasar a un esquema más potente en el que se conserve la interfaz con *Access* pero se materialice la base de datos en un SMBD de mayores prestaciones.

Como sucede a menudo, ambas estrategias tienen puntos a favor y en contra que el analista debe sopesar cuidadosamente. Hacerlo todo en *Access* es sin duda lo más simple y por tanto lo más rápido, pero implica que el SMBD debe arrastrar no sólo la gestión de datos de la aplicación, sino además todo el procesamiento requerido por la interfaz, con la consiguiente degradación del desempeño global del SI. Por otro lado, la interfaz *Access* funcionando contra una base de datos implantada en otro SMBD más potente, seguramente gana mucho en desempeño al SI, pero suele requerir más esfuerzo, tiempo e inversiones tanto en software como en hardware.

El analista debe decidir en cada caso particular, y dejando fuera los gustos personales y dogmas, la mejor solución para el problema que enfrenta sopesando todas las aristas y especialmente las características de la organización que utilizará su producto.

---

[9] ODBC: *Object Data Base Connectivity* o, en español, conectividad de objetos de base de datos.

En este texto no abordaremos aspectos de la estrategia Cliente-Servidor. En tanto que todos los conceptos asociados con el uso del *Access* son válidos en ambos contextos, supondremos en general una estrategia en la que base de datos y demás elementos componentes del sistema se almacenan dentro de un mismo archivo.

Para crear un nuevo archivo de base de datos debe invocarse *Access* y desde el ambiente inicial (figura 17) pulsar sobre "Base de datos en blanco" [4].

Lo que se obtiene es el despliegue del cuadro de diálogo que muestra la figura 18.

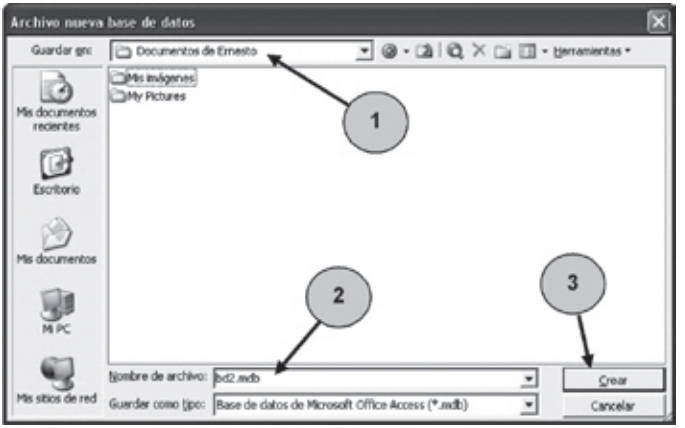


Figura 18. Ambiente inicial del Microsoft Access 2003

La figura 18 despliega una ventana estándar del *Windows* denominada cuadro de diálogo (por cuanto el usuario debe interactuar con ella, parafraseando una conversación). En la sección "Guardar en:" [1] se selecciona la carpeta o subdirectorio que recibirá el archivo, en la sección "Nombre de archivo:" [2] se escribe el nombre que desea dársele a la base de datos y, por último, al pulsar el botón "Crear" [3], se materializa la creación del archivo en el lugar establecido.

Una vez creado el archivo de BD, el cuadro de diálogo de la figura 18 se cierra y automáticamente se carga la base de datos en el ambiente *Access*. Las versiones modernas de sistemas operativos como *Windows* permiten manipular nombres largos y descriptivos de los archivos, sin

embargo, es conveniente procurar que los nombres de archivos sean mejor mnemotécnicos, no muy largos y sin caracteres especiales.

Recuérdese que la extensión de un archivo no es en realidad la que determina su tipo, se trata sólo de una indicación que forma parte de su nombre para efectos informativos. No obstante, cambiar la extensión de un archivo es un asunto delicado, pues gracias a ella, el sistema operativo reconoce su tipo (en principio) e invoca el aplicativo apropiado para manipularlo. En el caso del *Access*, sólo archivos con la extensión ".mdb" podrán ser manipulados, por lo cual es importante conservarla.

En la figura 18 se muestra la creación de un archivo de BD nombrado como "bd2.mdb" que quedará guardado en la carpeta "Documentos de Ernesto" del disco duro. Por supuesto, este nombre puede y debe ser más explicativo sobre el propósito de la base de datos que se crea.

Una vez concluido este paso, el sistema retorna el control al *Access* que ha cerrado la ventana "Nuevo archivo" y en su lugar está preparado para que el programador comience su trabajo desplegando la "ventana base de datos", a la cual nos referiremos en seguida.

3.2.2 La ventana base de datos

La figura 19 muestra la "ventana de bases de datos" tal como aparece al crear un archivo de base de datos en blanco, en el ambiente de trabajo *Access*.

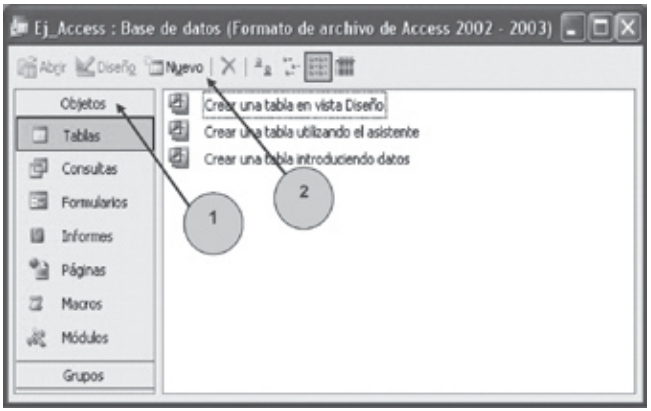


Figura 19. La ventana base de datos del Access

En la figura 19 se puede apreciar la sección "Objetos" [1], desde la cual se accede a las distintas pestañas de la ventana, cada una contenedora de los elementos que forman la base de datos. Los objetos con los que se trabaja en primera instancia son las tablas. Luego, en el proceso de construcción del sistema de información, se opera con Consultas, Formularios, Informes, Macros, Módulos y otros, no necesariamente en ese orden.

En la barra de herramientas señalada en la figura 19 [2] se encuentran las tres opciones comúnmente utilizadas para trabajar con las tablas de la base de datos, léase: "Abrir", que en este caso se refiere a abrir una tabla que ya exista en el archivo de BD (lo cual en nuestro ejemplo aún no ha ocurrido, por lo que esta opción se encuentra deshabilitada), "Diseño", que se refiere a la alteración del diseño de una tabla que ya existía (deshabilitada por la misma razón que antes), y "Nuevo", referido a la creación de una nueva tabla. A medida que se crean tablas dentro del archivo de BD, sus nombres aparecen en el área color blanco de la figura 19, desde la cual el usuario podrá seleccionarlas para realizar las acciones que desee. En la pestaña "Tablas" y en las restantes pestañas de la ventana de base de datos, los objetos pueden ser eliminados, copiados y demás, seleccionándolos y presionando las teclas adecuadas (p. ej.: "Suprimir" o Ctrl.-Ins).

Antes de continuar avanzando propondremos un ejemplo de base de datos fruto de un análisis previo -supuesto- y del diseño de un sistema de información que, para propósitos explicativos, será lo más sencillo posible.

Supongamos que nos hemos propuesto crear una biblioteca casera, en el sentido discutido en la primera parte del texto. Luego de todos los pasos descritos antes, llegamos al diagrama E – A, al Modelo R y al diagrama referencial. De allí tomaremos sólo algunas tablas y atributos para ilustrar la forma de proceder en *Access*. Nuestro interés se centra en la sección del sistema dedicada a los *Libros*, cuyo diagrama E–A muestra la figura 20.

La figura 20 es tan sólo una simplificación de todo lo que sería menester estudiar en una biblioteca casera. Más específicamente es una simplificación que comprende un subconjunto reducido de datos que tienen relación con los *libros*. Se suponen algunas cosas importantes, a saber:

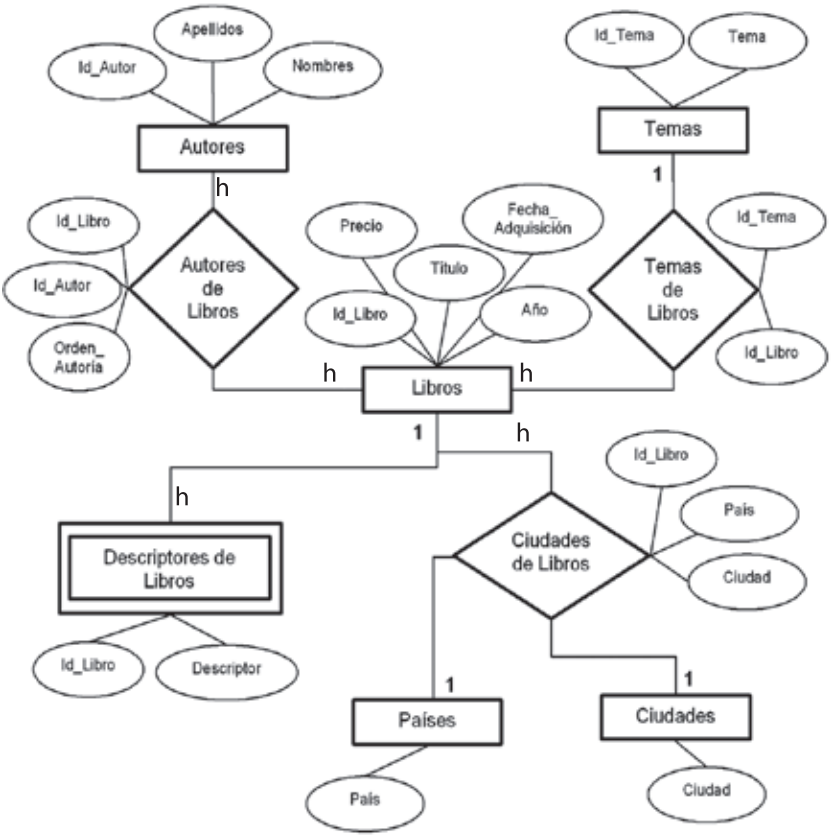


Figura 20. Diagrama E–A de la BD de libros

1. Entre los autores y los libros se establece una asociación de forma tal que un libro puede tener muchos autores y un autor puede haber escrito muchos libros.
2. Entre los temas y los libros se establece también una asociación, pero esta vez de manera que cada libro se corresponda a un solo tema, pero un mismo tema puede corresponder a muchos libros.
3. Los descriptores de los libros existen por sí mismos, pero supeditados a la existencia de los libros, esto es, el conjunto de entidades *Descriptores de Libros* es en realidad un conjunto de entidades débiles de los libros.

4. Entre los países, las ciudades y los libros se establece una asociación de forma que un libro se relaciona con un país y una ciudad, pero un país y ciudad puede ser la cuna de muchos libros.

Al construir el diagrama E–A, algunos autores omiten los atributos de las entidades que forman la asociación. Como se comprueba en el diagrama de la figura 20, en este caso preferimos no omitirlos en pro de la claridad de lectura del diagrama.

Nótese también que una asociación puede perfectamente contener atributos propios, aparte de aquellos representantes de las entidades que la forman. Es el caso de la asociación "Autores de Libros", en la cual además del "Id_Libro" y el "Id_Autor" aparece el atributo "Orden_Autoría". Podría pensarse en este atributo como una propiedad emergente de la autoría de libros, aparecida para indicar el primer autor, el segundo autor y así sucesivamente, en vista de que un libro puede tener varios autores.

En la tabla 1 se muestra el diseño de la BD, siguiendo ahora el modelo relacional.

TABLA 1. ESQUEMA RELACIONAL DE LA BD DE LIBROS

Libros = {Id_Libro, Título, Año, Precio, Fecha_Adquisición, Id_Tema, País, Ciudad}
Autores = {Id_Autor, Apellidos, Nombres}
Autores de Libros = {Id_Libro, Id_Autor, Orden_Autoría}
Temas = {Id_Tema, Tema}
Países = {País}
Ciudades = {País, Ciudad}
Descriptores de Libros = {Id_Libro, Descriptor}

En la tabla 1, el diseño relacional aludido no se desprende exactamente del diseño E–A de la figura 20 en el sentido de que cada objeto del diagrama E–A no resulta convertido en una relación. Obsérvese lo siguiente:

1. La asociación *Temas de Libros* no aparece en el esquema relacional; en su lugar, el atributo "Id_Tema" ha sido incorporado a la cabecera de la relación *Libros*. Tratándose de una relación en la

que se ha supuesto que a cada libro puede corresponderle sólo un tema, no es necesario crear una nueva relación. Basta entonces con agregar en la relación *Libros* un atributo (clave ajena) que representa al tema de cada uno.

2. No puede hacerse lo mismo con la asociación *Autores de Libros*, puesto que un libro suele tener varios autores. Luego, sería inconveniente disponer la lista de autores como atributos en *Libros*, entre otras razones porque de antemano no se conoce el número de autores de cada nuevo libro que se incorpore al sistema (de hecho, tal acto violaría principios de la normalización). Así pues, la forma correcta de pasar al modelo R preserva la asociación, ahora relación, *Autores de Libros*.
3. Los países que se definieron como un conjunto de entidades en el diagrama E–A se preservan ahora como una relación. Las ciudades, sin embargo, también representadas como conjuntos de entidades en el diagrama E–A, ahora son consideradas entidades débiles dependientes de los países. La razón es que no tiene sentido almacenar ciudades independientemente de los países que las contienen. Por otra parte, dado que un libro se ha supuesto proveniente de un solo país y ciudad, por las mismas razones que en (1) se ha obviado la asociación correspondiente y en su lugar se han incorporado los atributos "Id_País" e "Id_Ciudad" en la relación *Libros*.
4. La mayoría de las claves primarias escogidas no ameritan mayor discusión, excepto tal vez las de las relaciones *Autores de Libros*, *Descriptores de Libros* y *Ciudades*. En *Autores de Libros*, la clave primaria resulta compuesta de dos atributos, a saber: "Id_Libro" e "Id_Autor". La sola escogencia de "Id_Libro", aunque garantiza la unicidad en la relación *Libros*, no la garantiza en la relación *Autores de Libros*, pues cada libro puede tener múltiples autores. Igualmente sucede con el atributo "Id_Autor", el cual satisface las propiedades de unicidad y minimalidad en *Autores,* más no así en *Autores de Libros*. En consecuencia, la combinación de estos dos atributos formando la clave primaria de la relación *Autores de Libros* es la única garantía de unicidad y minimalidad. Razonamientos completamente análogos pueden hacerse para las relaciones *Descriptores de Libros* y *Ciudades*.

En la figura 21 se muestra el diagrama referencial de la base de datos alcanzada. Allí puede notarse cómo ha quedado finalmente el diseño.

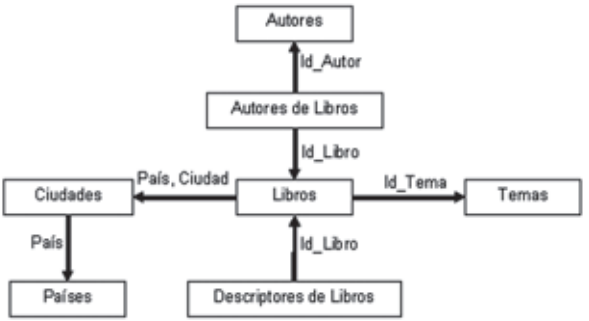


Figura 21. Diagrama referencial del esquema Libros

Como ilustra el diagrama referencial de la figura 21, las flechas parten de la relación referencial y llegan a la relación referenciada con base en las claves ajenas que pueden deducirse del esquema relacional.

Por ejemplo, *Autores de Libros* parte a buscar los libros utilizando el atributo "Id_Libro", que por ser la clave primaria de *Libros*, resulta una clave ajena en *Autores de Libros*. Dos anotaciones importantes: (1) En la práctica, esto quiere decir que no puede completarse la lista de los atributos de la relación referencial si primero no se encuentra la relación referenciada, y (2) Las claves ajenas pueden o no formar parte de la clave primaria de la relación referencial (sobre el particular no hay regla definida), por ejemplo, "Id_Tema" no forma parte de la clave primaria en *Libros*, no obstante, es una clave ajena que referencia a la relación *Temas*; por otra parte, "Id_Libro", en *Descriptores de Libros* sí forma parte de su clave primaria y es también una clave ajena que representa a los *Libros*.

Tabla 2. Dominios subyacentes del esquema Libros

Relación	Atributo	Semántica	Tipo	Restricciones
Libros	Id_Libro	Identificador del libro. Único y diferente para cada uno	Número entero largo	>0
	Título	Título o nombre del libro	Texto(200)	
	Año	Año en que se publica el libro	Número entero	>0
	Precio	Precio en Bs. al que se obtuvo el libro	Número monetario	>0
	Fecha_ Adquisición	Fecha de adquisición del libro	Fecha/Hora	
	Id_Tema	Identificador del tema del que trata el libro. Clave ajena que referencia la relación Temas	Texto(5)	Tiene que existir en Temas
	País	País en el que se publicó la edición del libro. Junto con Ciudad, clave ajena referenciando a Ciudades	Texto(30)	Tiene que existir la combinación con Ciudad en Ciudades
	Ciudad	Ciudad en la que se publicó la edición del libro, dentro del país. Junto con País, clave ajena referenciando a Ciudades	Texto(40)	Tiene que existir la combinación con País en Ciudades
Autores	Id_Autor	Identificador del autor. Único para cada autor de libros	Número entero	>0
	Apellidos	Apellidos del autor	Texto(20)	
	Nombres	Nombres del autor	Texto(20)	
Autores de Libros	Id_Libro	Identificador del libro. Clave ajena hacia Libros	Número entero largo	Tiene que existir en Libros
	Id_Autor	Identificador del autor. Clave ajena hacia Autores	Número entero	Tiene que existir en Autores
	Orden_ Autoría	Número en el orden de autoría que le corresponde al autor del libro	Número Byte	0-255
Temas	Id_Tema	Identificador del tema. Único para cada uno	Texto(5)	Exactamente 5 caracteres alfabéticos
	Tema	Descripción del tema	Texto(60)	
Países	País	Nombre del país. Identificador único para cada uno	Texto(30)	

Relación	Atributo	Semántica	Tipo	Restricciones
Ciudades	País	Identificador del país. Clave ajena referenciando a Países	Texto(30)	Tiene que existir en Países
	Ciudad	Nombre de la ciudad. Único en cada país	Texto(40)	
Descriptores de Libros	Id_Libro	Identificador del libro. Clave ajena hacia Libros	Número entero largo	Tiene que existir en Libros
	Descriptor	Frase que describe el contenido de un libro, única para cada libro. Un libro puede tener varios descriptores distintos	Texto(40)	

Por último y antes de comenzar la discusión sobre cómo se crean todos estos elementos en *Access*, echemos un vistazo a los dominios subyacentes de los atributos descritos en la tabla 2.

Veamos a continuación cómo se llevan a la realidad todos estos elementos, diseñados para nuestro sistema de información de *Libros*.

3.3 Tablas

Una *relación* es equivalente a una *tabla*, según como se concibe en el modelo relacional de base de datos. Se ha popularizado este vocablo en lugar de aquél, ya que representa un concepto menos abstracto, de uso común fuera del contexto teórico de la disciplina. La palabra "relación" se utiliza con su connotación matemática, mientras que el término "tabla" pretende familiarizar al usuario de bases de datos con una estructura de datos concreta, computacional, cuya idea le resulte más cercana.

Haciendo entonces la analogía, podría decirse que una tabla es una estructura computacional capaz de albergar datos agrupando un número no predeterminado de registros. A su vez, cada registro de una tabla es otra estructura computacional capaz de albergar datos de diferentes tipos referidos a un mismo objeto y que se corresponden con una colección predeterminada de campos.

Cada campo de un registro representa una característica diferente, un atributo de interés común para todos los objetos a los que se refiere la tabla.

De esta forma, una tabla resulta en un arreglo bidimensional de datos, tal que sus filas son registros referidos a cada objeto o individuo y sus columnas son los campos o características interesantes de dichos individuos.

Así, los conceptos vistos en el modelo relacional son directamente trasladables al contexto práctico de la manera siguiente: una tabla es una relación, un registro es una tupla del cuerpo de la relación y un campo es un atributo de su cabecera. Por extensión, también es cierto que todas las propiedades matemáticas de una relación son también las propiedades de una tabla, por ejemplo, en la tabla, así como en la relación, no hay orden de precedencia de los registros, tampoco de los campos; los campos se definen sobre dominios subyacentes como los atributos y se puede hablar de cardinalidad, claves primarias, etc.

La estrategia de creación de tablas partiendo del diagrama referencial (en este caso el diagrama que se muestra en la figura 21), es la siguiente:

- Las primeras tablas a materializar serán aquellas del diagrama que siendo o no referenciadas no hagan ellas referencia a otras tablas. En nuestro ejemplo, ellas son tres: *Autores*, *Temas* y *Países*. Éstas son las únicas que no hacen referencia a otras, por lo cual podemos crearlas en primer lugar.
- Las siguientes tablas a materializar serán aquellas que requieran de otras a las que hacen referencia, construyéndose a medida que se vayan completando sus prerrequisitos. En el ejemplo, el orden subsiguiente de creación de las tablas podría ser: *Ciudades*, *Libros*, *Autores de Libros* y *Descriptores de Libros*.

Esta estrategia permite aprovechar las características de búsqueda y enlace automáticas que proporciona el *Access*, así como definirlos con un formato del tipo "cuadro combinado" que, como veremos más adelante, fácilmente establezca listas de valores posibles tomados de sus tablas referenciadas.

Otra consideración importante es establecer cuidadosamente tanto las validaciones de cada campo, como las reglas de integridad referencial, antes de dar por completada la base de datos y proseguir con los demás elementos del sistema (como consultas, formularios e informes). Mucho tiempo se ahorra si formatos, máscaras de entrada y otras características se establecen desde el principio, pues *Access* las

reconoce e implanta automáticamente en formularios e informes. En caso contrario, se perderá bastante repitiendo estos elementos en ambos niveles (base de datos e interfaz).

Además, permitir la incorporación de datos sin haber previamente establecido las reglas de integridad referencial conducirá inevitablemente a inconsistencias que luego, cuando se apliquen las necesarias reglas, implicarán gran retrabajo sobre los datos.

Por otra parte, el diseñador debe indicar el tipo de datos que se espera en cada campo o atributo de manera que el computador reserve el espacio de almacenamiento correspondiente e incorpore las características específicas para su manipulación.

El lector debe estar familiarizado con los tipos básicos de datos en general, como enteros, reales, caracteres, cadenas de caracteres y otros. En nuestro ejemplo, la tabla 2 muestra los tipos requeridos para cada atributo del diseño. En el Apéndice "A" (incluido en el CD que acompaña el texto) pueden consultarse los tipos admitidos en *Access*.

Ahora bien, cuando desde la pestaña "Tablas" en la ventana de base de datos se presiona el botón "Nuevo" (ver figura 19), se obtiene el asistente para la creación de una nueva tabla. La figura 22 muestra la estructura de nuestra primera tabla, *Autores*.

Como puede observarse en la figura 22, la estructura de una tabla es a su vez una tabla. Contiene tres campos: "Nombre del Campo", "Tipo de Datos" y "Descripción". En este caso, la tabla *Autores* se ha creado con sus tres atributos, uno en cada fila del asistente mostrado en la figura. Nótese la llave a la izquierda del campo "Id_Autor". Esta llave se coloca seleccionando uno o varios campos (en esta caso, filas) y presionando el icono en la barra de herramientas –que no se muestra en la figura pero aparece cuando se invoca el asistente para la creación de una nueva tabla–, cuyo símbolo es precisamente una llave y que *Access* denomina "Clave principal". Los campos que muestren esta llave serán considerados la clave primaria de la tabla.

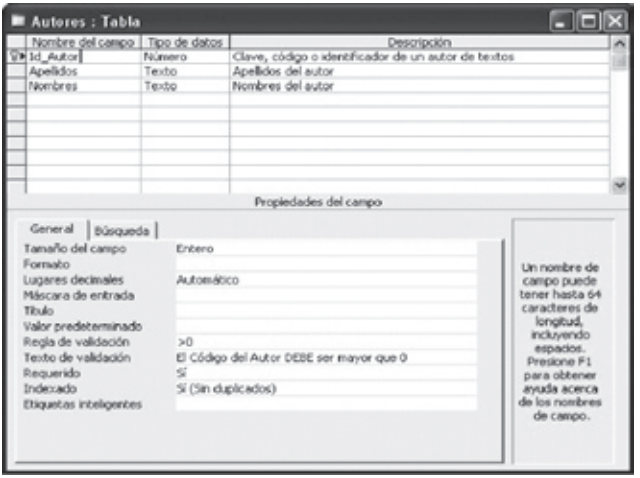


Figura 22. Estructura de la tabla Autores

Por otra parte, cuando se selecciona un campo de la tabla que está siendo creada o modificada, se obtienen en la parte inferior del asistente sus propiedades básicas, dispuestas en dos pestañas: "General" y "Búsqueda".

La pestaña "General" contiene las siguientes propiedades básicas de un campo:

1. Tamaño del campo. En el caso que muestra la figura 22, el campo es entero y su tamaño está predeterminado (ver en CD, Apéndice "A"). Si se tratara de un campo de texto, por ejemplo, en este lugar debería ponerse la extensión máxima permitida de la cadena de caracteres. De hecho, aunque en la figura no puede verse, para los campos Apellidos y Nombres este valor fue establecido en 20 caracteres.
2. Formato. En este caso, el campo "Id_Autor" no contiene formato alguno. Esto significa que aceptamos el formato que da *Access* a los números enteros; sin embargo, esta propiedad permite establecer la forma en que quiere el diseñador que sea visualizado el campo. Pueden establecerse aquí distintos tipos numéricos, por ejemplo, fijos con un número dado de decimales, o en notación científica, o si se trata de una fecha, en forma corta o larga, por mencionar sólo algunas posibilidades.

3. Lugares decimales. En este caso, "Automático". Claro está, tratándose de un campo de tipo entero, no interesa el número de lugares decimales que deseamos ver, puesto que no tendrá ninguno. En otros casos sí será de interés establecer el número de lugares decimales a mostrar en el campo.
4. Máscara de entrada. Todo campo puede ser visto por el usuario según una máscara predefinida. Por ejemplo, podemos querer que una cadena de caracteres se presente siempre en letras mayúsculas, o bien que un número telefónico incluya el código de área encerrado entre paréntesis y otros. Las máscaras a través de las cuales los usuarios verán los datos, son establecidas en esta propiedad. Para el caso que nos ocupa no ha sido necesario decidir máscara alguna.
5. Título. El nombre que recibe un campo no siempre es suficientemente claro con respecto a lo que significa dicho campo. De hecho, el nombre del campo señalado en la figura 22, no es un vocablo castellano (aunque se comprende bien). Si se quisiera mostrar al usuario un nombre más explicativo que el que se ha asignado al campo, eso debe hacerse en esta propiedad. Por ejemplo, podríamos haber escrito "Identificador del Autor" y en todas partes donde se utilizara este campo aparecería esta frase en lugar de "Id_Autor". No hemos considerado necesario hacerlo para manejarnos con el mismo nombre y título de campo, abreviando un poco. En general, sin embargo, es una buena práctica clarificarlo, especialmente si el nombre del campo no es mnemotécnico.
6. Valor predeterminado. También puede establecerse un valor por defecto para cada campo. Esta opción resulta muy útil cuando se sabe con antelación que un cierto valor será repetido muchas veces para un campo. Por ejemplo, si almacenáramos un campo para la nacionalidad de una persona según la cédula de identidad venezolana (que admite los valores V o E), sería apreciado por el transcriptor de datos que el valor por defecto fuera "V", ya que la mayoría de las personas en Venezuela tendrán este valor en su cédula, y entonces ahorraría a la larga mucho esfuerzo de escritura.
7. Regla de validación. Se trata de un espacio para fijar una condición lógica que aplicada sobre el campo determine los valores que serán aceptables. La condición lógica puede ser extensa,

puede incluir operadores lógicos y funciones del sistema, pero no puede tener relación con otro campo de la tabla. Para este tipo de validaciones se requiere recurrir a la interfaz o al control de integridad referencial, de lo cual hablaremos más adelante. Para el caso que nos ocupa, la condición lógica prevista es ">0". Ello implica que cada vez que un usuario introduzca un valor en el campo "Id_Autor", el SMBD comprobará automáticamente que dicho valor sea superior a cero, y de no ser así, será presentado un mensaje de error y no le será permitido almacenar el campo en la base de datos. Son permitidas validaciones de números, fechas, caracteres y en general cualquier tipo de dato.

8. Texto de validación. En esta espacio debe escribirse el texto del mensaje que será mostrado al usuario en caso que la regla de validación no sea satisfecha.
9. Requerido. Puede tomar los valores "Si" o "No", dependiendo de si se desea que el campo al cual se refiere la propiedad deba escribirse obligatoriamente o no. En otros términos establece si el campo en cuestión puede o no quedar sin valor al momento de la inserción de datos en el registro. Si el campo puede quedar en blanco, entonces no es requerido y viceversa. Para el caso de la figura 22, tratándose de la clave primaria de la tabla es redundante establecer la propiedad a "Si", pues todo atributo integrante de una clave primaria siempre es requerido, nunca puede quedar en blanco y esto lo controla automáticamente el SMBD.
10. Indexado. Puede decidirse crear un índice para un campo y establecer su tipo. En este caso se ordena al SMBD que cree un índice para el atributo "Id_Autor", sin duplicados. Esto siempre ocurrirá sobre la clave primaria de la tabla (toda clave primaria recibirá un índice sin duplicados), pero no ocurre así para los demás campos. Los índices en las bases de datos relacionales son construcciones internas del manejador que, al mantener un orden específico de los datos (como si se tratara del índice alfabético de un libro, por ejemplo) y ser en extremo eficientes en la búsqueda y recuperación, aligeran en gran medida el proceso de manipulación de datos a partir de los valores del campo indizado o indexado. El SMBD primero consulta el índice donde se indica la dirección de disco en la que se encuentra la información del registro y se dirige directamente a dicha dirección. Se trata de

una característica clave de los manejadores para mejorar su desempeño, pero lamentablemente consume considerables recursos del computador, así que es menester establecerlos sólo para aquellos campos por los cuales se espera recibir frecuentes pedidos de búsqueda.

11. Etiquetas inteligentes. Es una nueva característica de la versión 2003 que opera de manera similar en los demás aplicativos de la suite *Office*. Se trata de una marca especial que permite al sistema reconocer el campo "inteligentemente" en todo lugar en el que se le haga mención. Esta "inteligencia" significa que el *Office* sugerirá acciones particulares, como enviar correos electrónicos o agregar direcciones, cada vez que aparezca la etiqueta. No la utilizaremos nuevamente a lo largo del texto.

Finalmente, para cada propiedad en el asistente, el recuadro que se encuentra en la parte inferior derecha de la figura 22 mostrará un breve comentario de ayuda.

Abordaremos la pestaña "Búsqueda" a partir de otra nueva tabla que incluya claves ajenas. Antes, y siguiendo el proceso descrito, deben crearse (lo que se deja como tarea al lector) las tablas *Temas* y *Países*.

Un comentario sobre el particular: En *Temas* será necesario establecer una máscara de entrada para el campo "Id_Tema". Una máscara de entrada se compone de tres secciones separadas por puntos y comas (u otro separador de listas definido en el computador). La primera sección es la máscara en sí, la segunda indica con un número si deben o no almacenarse los caracteres de relleno que posiblemente contenga la máscara (paréntesis o guiones generalmente), y la tercera sección indica el caracter que se desea aparezca en lugar de los espacios vacíos en el campo. La máscara en cuestión es entonces ">LLLLL;;_". Este pseudocódigo que entiende *Access* como una máscara de entrada, se interpreta de la siguiente forma: la primera sección comienza con ">" indicando que todo caracter será convertido a letras mayúsculas, luego, "LLLLL", es decir, cinco veces "L", indica que los cinco caracteres son de entrada obligada y alfabéticos, no numéricos. La segunda sección está vacía, indicando que aceptamos el valor por defecto (que es no guardar) sobre los caracteres de relleno –de hecho, aquí es irrelevante porque nuestra máscara no contiene caracteres de relleno–, y la tercera sección contie-

ne el caracter "_", ordenando que sea un subrayado lo que se presenta en lugar de los espacios vacíos.

Continuando con la implementación, la figura 23 muestra la estructura de la tabla *Libros*, en particular la conformación de las propiedades del campo "Id_Tema". Se trata de un atributo que participa como clave ajena de la tabla *Temas*, de manera que se espera que sus valores puedan seleccionarse de una lista. Para lograr este comportamiento, el objeto contenedor del campo debe ser un "cuadro combinado".

Un cuadro combinado es un objeto de la interfaz *Windows* capaz de recibir valores desde una lista que se despliega a voluntad. En este caso pretendemos que la lista de temas que hayan sido definidos en el sistema aparezca en el cuadro combinado que representa al "Id_Tema", de forma que el usuario pueda seleccionar el identificador del tema al que corresponde el libro cuyos datos se editan para el momento.

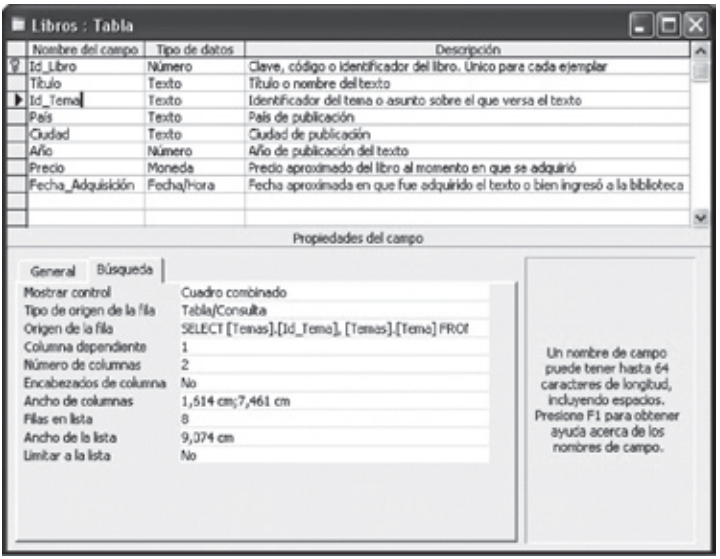


Figura 23. Estructura de la tabla Libros

La primera propiedad de búsqueda en este campo, "Mostrar control", lo define precisamente como un cuadro combinado, lo que habilita el resto de las propiedades que son inherentes a este tipo de objetos. Le

sigue la propiedad "Tipo de origen de la fila", en la cual se establece el tipo de la fuente de los valores que serán mostrados en la lista del cuadro combinado. Aunque hay otros tipos posibles (como, por ejemplo, una lista estática de valores que no provienen de tablas), en un modelo relacional, la más frecuente es la que se ve en la figura 23, es decir, Tabla/Consulta. Esto indica al SMBD que debe encargarse automáticamente de poblar la lista de valores del cuadro combinado a partir del resultado de consultar una tabla. La propiedad "Origen de la fila" establece la fuente de donde se extraerán los datos que poblarán el cuadro combinado. Como dijimos, esta fuente es el resultado de ejecutar una consulta a la base de datos. Más adelante examinaremos con detalle las consultas, pero por lo pronto véase la figura 24, en la que se construye la consulta para la lista del cuadro combinado que representa al campo "Id_Tema".

La figura 24 es tal vez la representación más sencilla de una consulta a una tabla. Muestra un asistente del *Access* especialmente diseñado para tomar provecho del modelo relacional y facilitar la construcción de consultas de forma visual (no programática). Se le denomina "Generador de Consultas" del tipo QBE.

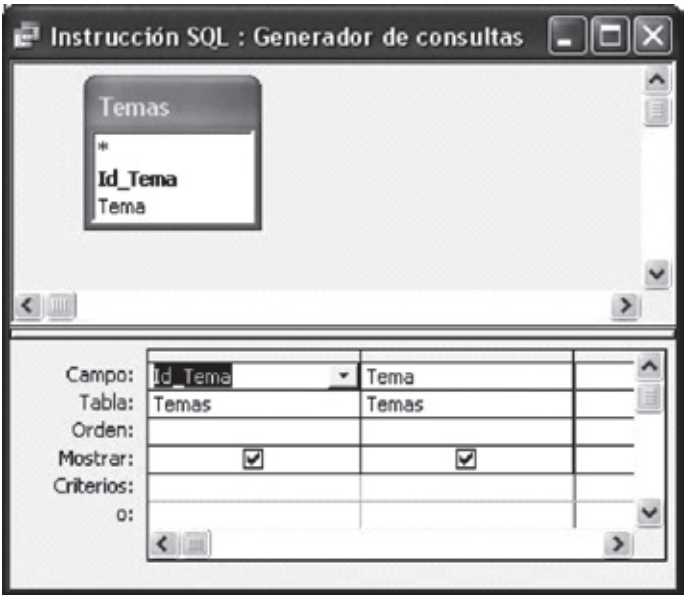


Figura 24. Consulta sobre la tabla Temas

En el generador de consultas se selecciona la tabla que será objeto de interrogación – en este caso la tabla *Temas* –. De ella, en la parte inferior, se seleccionan los campos que intervendrán en la respuesta (aquí "Id_Tema" y "Tema"). Este generador permite muchas otras opciones que en cualquier caso siempre resultan en una instrucción en sintaxis SQL que el SMBD puede resolver para devolver, en forma de tabla, los valores indicados. La instrucción SQL que contiene la consulta esbozada en la figura 24, es la siguiente:

```
SELECT [Temas].[Id_Tema], [Temas].[Tema]
FROM Temas;
```

En otras palabras, "seleccione de la tabla *Temas*, los campos 'Id_Tema' y 'Tema'". Esta instrucción SQL es la que se asoma en la propiedad "Origen de la fila" de la figura 23. El resultado es que el cuadro combinado que contiene el campo "Id_Tema" incluirá una lista a dos columnas de todos los identificadores de temas y temas que se encuentren presentes en la tabla *Temas*. De entre ellos deseamos que el usuario pueda seleccionar uno y, hecha la selección, el valor escogido del "Id_Tema" se almacene en el campo homónimo de la tabla *Libros*.

La propiedad "Columna dependiente" de la figura 23 permite al diseñador establecer qué columna de las de la tabla o consulta origen de la fila será la que deberá copiarse al cuadro combinado, a partir de la selección que haga el usuario de la lista que éste le presenta. En este caso, dicho valor se tomará de la primera columna de la lista, o en otras palabras, el campo "Id_Tema" de la tabla *Temas* será copiado al cuadro combinado "Id_Tema" de la tabla *Libros*.

La propiedad "Número de columnas", como su nombre indica, determina el número de ellas que contiene la lista, en este caso dos ("Id_Tema" y "Tema"). La propiedad "Encabezados de columna" permite establecer si se deberán o no mostrarse los rótulos de los campos que forman la lista en el cuadro combinado. Por lo general, esta información es redundante, pues el usuario sabrá que se encuentra editando el campo y no necesitará ver los rótulos. No siempre es evidente y por ello la existencia de esta propiedad.

Las propiedades "Ancho de columnas", "Filas en la lista" y "Ancho de la lista" completan el formato que desea darse a la lista del cuadro combinado. En este caso, cada columna tiene un ancho preestablecido de 1,614 cm. y 7,461 cm. respectivamente. Se mostrarán directamente hasta 8 filas en la lista, quedando las demás, en caso de existir, sujetas

al movimiento de las barras de desplazamiento del cuadro combinado, y el ancho total de la lista a mostrar será 9,074 cm.

La última de las propiedades mostradas en la figura 23, "Limitar a la lista", permite al diseñador decidir si el usuario sólo puede incorporar valores que se encuentren en la lista desplegada o le dejará en libertad de incorporar otros valores ausentes de la lista. En nuestro ejemplo se ha dejado el valor por defecto que es negativo, sin embargo, esta propiedad está subordinada al mantenimiento de la integridad referencial que indicará al SMBD, como veremos en la siguiente sección, no permitir valores del campo "Id_Tema" que no se encuentren primeramente en la tabla *Temas* (lo cual automáticamente anula la posibilidad de admitir valores fuera de la lista).

Sería entonces momento de completar el diseño siguiendo el diagrama referencial de la figura 21, así como la tabla 2, lo cual se deja como tarea al lector.

3.4 Integridad referencial y la ventana relaciones

Estando en la pestaña "Tablas" de la ventana de base de datos, la barra de herramientas habilita un botón denominado "Relaciones" (ver figura 25).

Figura 25. Barra de herramientas de edición de tablas

"Relaciones", en *Access*, es el nombre que reciben los vínculos existentes entre las distintas tablas que conforman la base de datos. Representan la materialización del esquema relacional de la base de datos. Operativamente se trata de una ventana en la que el diseñador define "visualmente" las condiciones de integridad referencial. La gráfica resultante es una variación del diagrama referencial que incluye los campos de cada tabla y la cardinalidad de mapeo entre ellas. Aquí

se encuentra el corazón de toda base de datos relacional. La figura 26 muestra el esquema de la base de datos de libros.

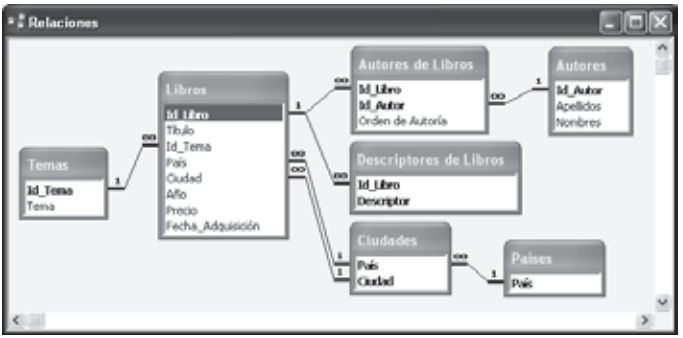


Figura 26. Esquema de la base de datos de Libros

Nótese de la figura 26 que se han creado las siete tablas del ejemplo que venimos siguiendo, exactamente como estaba previsto en el diseño relacional (ver tabla 1). Los campos de las tablas que se encuentran en negritas en el diagrama son sus claves primarias, y las líneas que las unen son las relaciones que se establecen entre ellas.

Cuando se establece una relación entre dos tablas, o más específicamente entre dos (o más) atributos de dos tablas, *Access*, por defecto, no verifica el cumplimiento de la integridad referencial. La integridad referencial es el término utilizado para denotar el que en una relación, el SMBD controle que tal vínculo sea consistente en el tiempo.

Una relación entre dos tablas en la que se controla la integridad referencial, deja de manos del SMBD controlar que los datos de una se correspondan con los de la otra. Por ejemplo, la relación entre las tablas *Libros* y *Temas,* que incluye controlar su integridad referencial, se comporta de manera tal que no puede incluirse en la tabla *Libros* ningún "Id_Tema" que no esté previamente incluido en la tabla *Temas*.

Adicionalmente, al controlar la integridad referencial, el SMBD puede ejecutar también actualizaciones y eliminaciones en cascada a voluntad del diseñador. Una actualización en cascada produce que cuando se cambie el valor de un campo, clave primaria de una relación referenciada, este cambio se propague automáticamente a todas las claves ajenas de ese campo en las relaciones referenciales. En nuestro ejemplo, si hay libros de la tabla *Libros* con un Id_Tema="MAT_A" y el usuario

cambia en la tabla *Temas* Id_Tema="MAT_A" por Id_Tema="MAT_Z", el SMBD automáticamente cambiará el valor del campo Id_Tema en *Libros* actualizándolo a "MAT_Z" en lugar de "MAT_A".

De forma análoga, si se ha establecido la indicación de eliminación en cascada para una relación entre atributos de dos tablas, cuando el usuario elimine un registro de la tabla referenciada, el manejador automáticamente eliminará todos aquellos registros de la tabla referencial que contengan el valor en cuestión. En nuestro ejercicio, si eliminásemos ahora en *Temas* el registro Id_Tema="MAT_Z", se borrarían automáticamente todos los libros en la tabla *Libros* cuyo tema fuese "MAT_Z". Obviamente, ese no es el comportamiento que quisiéramos para nuestras tablas *Libros* y *Temas*.

Para establecer una relación entre campos de dos tablas, desde la ventana "Relaciones" mostrada en la figura 26 se "toma y arrastra" uno de los campos de la tabla referencial y se "suelta" sobre el campo correspondiente en la tabla referenciada (esta operación, muy común en *Windows* y con la que el lector seguramente está familiarizado, se denomina *drag and drop*, en inglés, o "pulsar y arrastrar", en español). Ello abre un asistente para la creación o modificación de relaciones, como el que se muestra en la figura 27 para aquella entre *Ciudades* y *Libros*.

En la figura 27 se puede apreciar en primer lugar que la relación entre dos tablas puede establecerse mediante varios campos y no ne-

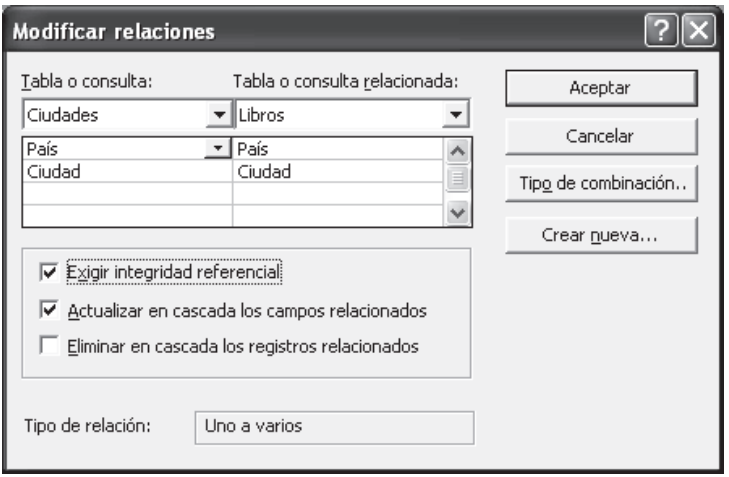


Figura 27. Características de la relación entre Ciudades y Libros

cesariamente por medio de uno sólo. En general, los campos que se utilizarán son los correspondientes a clave primaria y ajena, según se trate de la relación referenciada o referencial. En el ejemplo mostrado la relación referencial es *Libros*, y la referenciada es *Ciudades*.

Otro aspecto a señalar es que una relación no necesariamente debe establecer el control de la integridad referencial. Ello queda a voluntad del diseñador, quien puede activar o desactivar las casillas de verificación correspondientes.

Por último, en caso que se controle la integridad referencial, resulta de primera importancia comprender el funcionamiento de las opciones de actualización y eliminación en cascada. En el ejemplo de la figura 27 sólo se ha activado la actualización en cascada. Así, cuando se actualiza una ciudad-país de la tabla *Ciudades,* indicamos que automáticamente se actualicen los nuevos valores en *Libros*. Por otro lado, aunque podría pensarse que parte de la consistencia de los datos tiene que ver con la desaparición o no de una ciudad-país de la tabla *Ciudades*, debiera hacerse lo propio con las tuplas de *Libros* que le hayan incluido. En realidad, no es precisamente eso lo que desearía el diseñador de la base de datos. Dado que las ciudades y países incluidos en *Libros* son sólo algunos atributos de cada ejemplar, no necesariamente los más importantes, lo que el diseñador desearía en realidad es que si el usuario intentase eliminar un registro de la tabla *Ciudades*, y los valores de este registro estuvieran siendo utilizados en *Libros*, el SMBD automáticamente le impidiese su eliminación. El comportamiento, en este caso, es preservar la consistencia de la base de datos por la vía de impedir una eliminación, en lugar de desencadenarla en cascada.

No ocurre así por ejemplo entre las tablas *Libros* y *Descriptores de Libros*. Tratándose la segunda de una entidad débil de la primera (y no de una asociación), el comportamiento previsible debe ser tal que si desaparece un libro de la base de datos, desaparezcan también sus descriptores. En este caso, debería entonces activarse la casilla de verificación para la eliminación en cascada. En síntesis, la actualización en cascada generalmente es apropiada y deseable, pero la eliminación en cascada no siempre será apropiada. Debe pensarse cuidadosamente en cuáles oportunidades se desean estas características, que ponen al SMBD en el control de modificaciones críticas en la BD.

3.5 Lenguaje de consulta estructurado (SQL)

Una consulta es en la práctica un conjunto de instrucciones escritas en un lenguaje estructurado que, al ser obedecidas por el SMBD, actúan sobre las tablas y los datos combinándolos de la forma indicada en la lógica de la instrucción, bien sea para presentar un resultado al usuario que agrupe o no datos, o modificar el contenido de la base de datos.

Hoy en día, la mayoría de las versiones de SQL contienen tanto el lenguaje de definición como el de manipulación de datos. Este también el caso en *Access*, sin embargo nos concentraremos en este texto en la sección del SQL que se dedica a la manipulación de datos, en vista de que las facilidades del SMBD para la definición de datos (como creación, modificación o eliminación de tablas) utilizando asistentes, ya han sido cubiertas.

Access clasifica las consultas SQL en tres grupos, a saber:

- Consultas de selección simple: aquellas que aun cuando pueden involucrar múltiples tablas e incluso otras consultas previamente almacenadas, se destinan a presentar los datos en toda su extensión, esto es, sin resumir, tomando subconjuntos de campos, atendiendo a criterios lógicos e incluyendo operaciones sobre ellos a voluntad.
- Consultas de agrupamiento: aquellas consultas de selección que además de poder incluir múltiples tablas u otras consultas, criterios y operaciones sobre los campos, se destinan a combinar los datos presentando resultados de algún tipo de cálculo que les sintetice o agrupe de alguna forma; en otras palabras, su resultado ya no contiene los mismos datos presentes en la base de datos, sino otros nuevos como resultado de aplicar operaciones de agrupamiento sobre la base de datos.
- Consultas de acciones: aquellas consultas cuyo propósito es la modificación de datos. Las modificaciones que se permiten son la inserción y eliminación de registros, así como la actualización de sus campos. También entran en esta categoría las instrucciones SQL que se destinen a la creación y eliminación de tablas.

Consultas de selección simple

La figura 28 muestra una consulta sencilla que busca obtener todas las ciudades que pertenezcan a "Estados Unidos" en nuestra BD de *Libros*.

Para construir esta consulta desde el asistente que proporciona *Access*, es necesario encontrarse en la pestaña "Consultas" de la ventana de base de datos y allí presionar el botón "Nuevo". Se ejecutará entonces un subprograma que proporciona múltiples maneras de crear una consulta, no obstante, la más general y que utilizamos aquí, es la opción "Vista Diseño".

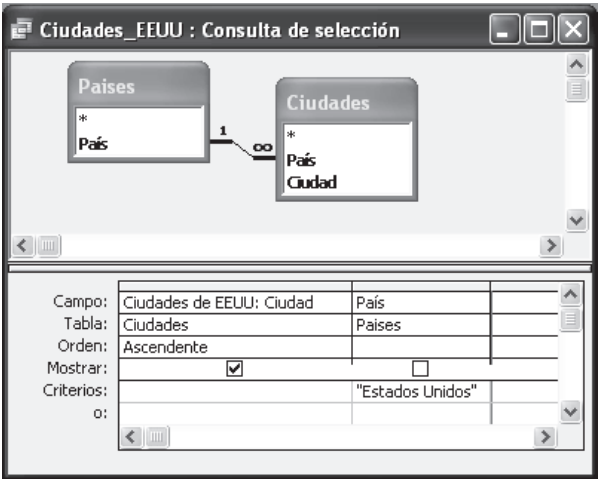


Figura 28. Consulta de las ciudades de los EEUU

Como puede notarse en la figura 28, el asistente para consultas proporciona dos áreas de trabajo. El área superior se destina a las tablas que participarán en la consulta, mientras que la inferior permite la selección, ordenamiento y establecimiento de criterios lógicos sobre los campos de dichas tablas. Para indicar al diagramador de consultas las tablas que participarán, se debe acceder a la lista de tablas presentes en la base de datos que se muestra al presionar el icono "Mostrar tabla", rotulado con el número 5 en la figura 29 (de la barra de herramientas para consultas).

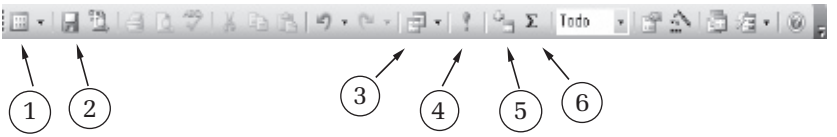


Figura 29. Barra de herramientas para las consultas

En la figura 30 se aprecian todas las tablas que contiene nuestra base de datos de libros, según se muestran al presionar el botón "Mostrar tabla" [5] de la figura 29. En este caso se han seleccionado simultáneamente[10] dos de ellas: *Ciudades* y *Países*. Cuando se presiona luego el botón "Agregar" mostrado en la figura 30, ambas tablas pasan al área dispuesta para ello en el diagramador de consultas.

Figura 30. Tablas de la BD de Libros mostradas para elaborar consultas

[10] Recuérdese que en algunas ventanas de Windows se permite la selección de múltiples elementos mediante la combinación de la tecla "Ctrl" y un clic de ratón para elementos separados, o de la tecla "Shift" y el ratón para elemento consecutivos.

Aunque puede seleccionarse cualquier cantidad de tablas y consultas de la base de datos y combinarse de muchas maneras arrastrando campos de unas a otras para establecer eventualmente cualquier relación que se desee, por defecto, el diagramador de consultas supone las relaciones entre tablas que se han definido en la ventana "Relaciones" (ver figura 26), esto es, en el esquema de la base de datos. En este caso, al seleccionar las tablas *Países* y *Ciudades,* y presentarlas en el área de tablas, automáticamente se supone que la forma en que ambas se asocian parte del campo "País".

Volviendo a la figura 28 puede notarse que de la tabla *Países* se selecciona el campo "País" y de *Ciudades* el campo "Ciudad". La casilla de verificación de la sección "Mostrar" está activada en el campo "Ciudad" y desactivada en el campo "País", indicando al SMBD que la respuesta deberá contener los valores obtenidos para las ciudades pero no los que se desprenden de sus países asociados. Obviamente, si estamos preguntando por las ciudades de los Estados Unidos, todos los valores del campo "País" serán los mismos y evidentes, por lo cual no es necesario mostrarlos. Aún así, para evitar confusión al usuario se ha previsto que el campo "Ciudad" reciba otro nombre más explicativo, específicamente, "Ciudades de EEUU". La forma de renombrar un campo en la respuesta de una consulta es anteceder el nombre de dicho campo por el nombre que se desea mostrar, seguido del carácter de dos puntos (":"). La opción "Ascendente" en la sección "Orden" para el campo "Ciudad", indica al SMBD que ordene las ciudades alfabéticamente de la "A" a la "Z".

Asimismo obsérvese que el hecho de que un campo no se muestre en la respuesta de una consulta no quiere decir que no participe en ella. De hecho, el campo "País" de la figura 28 no se mostrará en el resultado aun cuando participa en la consulta, pues sobre él se establece un criterio que filtra o tamiza los registros que serán mostrados.

Una vez construida y almacenada la consulta (presionando el botón "Guardar" [2] de la barra de herramientas mostrada en la figura 29), se puede presentarla desde el diagramador presionando el botón "Ejecutar" [4]. En *Access*, el símbolo de cierre de exclamación ("!") siempre se identifica con la ejecución de algún proceso.

Los botones "Tipo de consulta" [3] y "Totales" [6] de la barra de herramientas mostrada en la figura 29 se destinan a establecer respectivamente la caracterización de la consulta, en este caso como consulta

de selección simple y si se trata o no de una consulta de agrupamiento (en este caso, no).

La respuesta que se obtiene del SMBD al ejecutar esta consulta se muestra en la figura 31. Hay 19 ciudades en la base de datos consideradas como pertenecientes a los Estados Unidos. Nótese que el SMBD es bien capaz, pero no lo suficiente, como para darse cuenta de que no todas las ciudades mostradas son tales; por ejemplo, aparece "Massachussets" como ciudad y sabemos que esta no es una ciudad, sino un estado de la Unión. Se trata de un error conceptual cometido por el transcriptor de datos que incorporó esta cadena de caracteres como si se tratara de una ciudad sin serlo, pero el SMBD no está en capacidad de detectarlo.

Nótese además que no todas las ciudades presentes en la BD son visibles en la ventana, pero esto se debe a su tamaño y no representa problema alguno, pues para inspeccionar todos los elementos de la respuesta el usuario puede apoyarse en la barra de desplazamiento vertical.

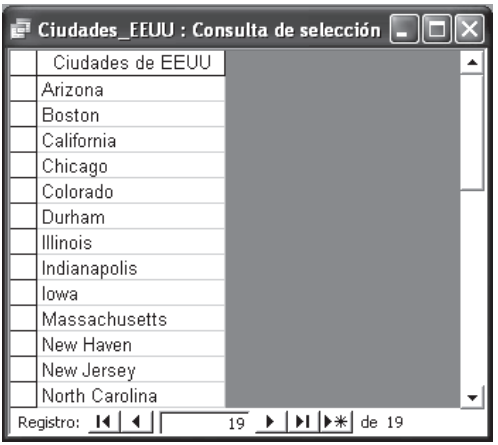


Figura 31. Resultado de la consulta Ciudades_EEUU

Nótese por último que la respuesta a este tipo de consultas es también una tabla en el sentido de que contiene filas, columnas, registros y sus atributos, sin embargo, puede comprobarse que en realidad los datos mostrados en la figura 31 no son nuevos y no existe una estructura de datos permanente que los contenga tal como se presen-

tan aquí. En realidad, los datos de las ciudades de los Estados Unidos se encuentran donde deben, esto es, en la tabla *Ciudades*. La consulta sólo los ha capturado de allí cumpliendo los criterios de filtrado, renombrando los campos a voluntad y ordenándolos para presentarlos al usuario sin almacenarlos permanentemente. Cuando se cierre la consulta en cuestión, estos datos serán desechados de la memoria (no de la base de datos), y si vuelve a invocarse la consulta en el futuro, ésta será objeto de una nueva revisión de la base de datos. Así ocurrirá cada vez que se ejecute una consulta, lo cual garantiza que su resultado siempre contendrá los datos actualizados.

La figura 32 contiene la misma consulta de la 28, esta vez en su forma SQL. Para acceder a las distintas formas de presentación de una consulta se utiliza el botón "Vistas" [1] de la barra de herramientas desplegada en la figura 29.

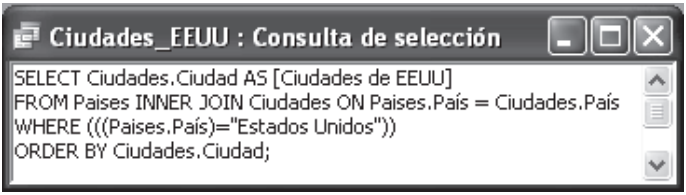


Figura 32. Consulta de las ciudades de los EEUU como instrucción SQL

La instrucción escrita en lenguaje de consulta estructurado que se presenta en la figura 32 fue redactada automáticamente por el diagramador de consultas a partir de las instrucciones que se dieron de forma "visual". Se encuentran las cuatro cláusulas principales del SQL, a saber: *SELECT*, *FROM*, *WHERE* y *ORDER BY*.

La cláusula *SELECT* ("seleccionar", en español) se utiliza para indicar qué campos de las tablas o consultas escogidas serán mostrados en la respuesta a la interrogante planteada. En este caso será el campo Ciudad de la tabla *Ciudades*, renombrado (lo que se logra con la palabra clave "*AS*") como "Ciudades de EEUU". Como puede verse en la figura 32, se utilizan nombres completos de los campos. Estos nombres completos se componen del nombre de la tabla, a continuación un punto ("."), seguido del nombre que recibe el campo en la tabla. Este nombre

completo cobra especial relevancia cuando hay campos de tablas intervinientes en la consulta con idénticos nombres y el gestor de consultas necesita entonces identificarlos sin ambigüedad.

Cuando un campo al que se desea hacer referencia contiene espacios u otros caracteres especiales, debe encerrarse entre corchetes ("[ ]"). *Access* entenderá en todos los contextos que una cadena de caracteres encerrada entre corchetes es el nombre de un campo de tabla o consulta, o un parámetro.

La cláusula *SELECT* admite comodines, esto es, símbolos con un significado especial que se emplean para abreviar. Uno muy frecuente es el asterisco ("*") que indicará al gestor de consultas que muestre "todo". Por ejemplo, si en lugar de sólo mostrar el campo "Ciudad" de la tabla *Ciudades* se deseara mostrar todos sus campos, la porción de la instrucción SQL en la cláusula *SELECT* sería: "SELECT Ciudades.*", claro está, esta opción no admite cambiar el nombre a los campos.

La cláusula *FROM* ("de", en español) en la instrucción SQL se utiliza para indicar las tablas o consultas de donde debe extraerse la información que se desea presentar. En el ejemplo que seguimos a partir de la figura 32 puede verse que la información saldrá de las tablas *Países* y *Ciudades,* combinadas de manera que los registros que se mostrarán serán aquellos cuyo valor del campo "Países.País" sea igual al de "Ciudades.País". Esto se consigue con la frase reservada *INNER JOIN – ON*, que podría interpretarse como "obtenga los datos del producto cartesiano de las tablas *Países* y *Ciudades*, restringido a aquellos registros que comparten el valor del campo 'País'". Por supuesto, existen varias formas adicionales de relacionar dos (o más) tablas en una consulta utilizando el *JOIN*, pero están fuera del alcance introductorio que nos proponemos en este texto.

La cláusula *WHERE* ("donde", en español) de la figura 32 se utiliza para indicar los criterios de filtrado o tamizado de los registros que deberán aparecer en la respuesta. El filtrado de un conjunto relacional de datos se refiere a la selección de sólo aquéllos que cumplen estrictamente con la condición. En nuestro ejemplo, de todas las ciudades de países que comparten el mismo valor para el campo "País" sólo se mostrarán aquellas cuyo valor para este campo sea "Estados Unidos". En términos de condiciones lógicas computacionales esto es "Países. País = 'Estados Unidos'".

Las condiciones lógicas permiten cualquier combinación de campos y admiten cualquier tipo de operadores lógicos y de comparación. Se pueden utilizar los operadores *AND* (Y), *OR* (O) y otros, así como los operadores de comparación IGUAL QUE ("="), MAYOR O IGUAL QUE (">="), MENOR QUE ("<"), etc. Un operador de comparación muy útil y especial para el SQL es el operador *LIKE* (en español se entendería como "parecido a"). Por ejemplo, si en lugar de exigir la identidad de los valores del campo "País" con la cadena de caracteres "Estados Unidos" se deseara encontrar los registros cuyos valores en el campo "País" se parecen a "Estados", la frase de la cláusula *WHERE* correspondiente sería: "WHERE Países.País LIKE 'Estados*'". De hecho, la respuesta a ambas interrogantes con los datos actuales sería idéntica, la ventaja estriba en que el usuario no tendría que saber que el país "Estados Unidos" está escrito exactamente así. Sólo debería suponer que comienza por la cadena 'Estados' seguida de cualquier otra cosa ("*") y que lo que desea se parece a esto.

La secuencia de paréntesis que contiene la cláusula *WHERE* de la figura 32 es redundante y ha sido introducida por el traductor de consultas para acentuar la precedencia de los operadores. La instrucción "WHERE Países.País = 'Estados Unidos'" funcionaría de igual forma en este caso particular, ya que no hay ambigüedades ni precedencias especiales en los criterios de nuestro ejemplo.

Por último, la cláusula *ORDER BY* ("ordenado por", en español) de la figura 32 indica al procesador de consultas los campos por los cuales se desea ordenar el conjunto de registros resultante. En este caso hay un solo campo, la ciudad. No obstante, si se diera el caso de que se deseara ordenar por varios campos, éstos deberán disponerse a continuación de la cláusula separados por comas (","), precisamente en el orden que se desea. Por ejemplo, si en una consulta de personas se deseara la lista ordenada alfabéticamente por apellidos y nombres, la cláusula sería: "ORDER BY Personas.Apellidos, Personas.Nombres". Ello indicaría al gestor de consultas que los registros que obtenga deberán ordenarse primero por apellidos y, dentro de los apellidos, por nombres (es importante comprender qué significa "dentro de"). La cláusula *ORDER BY* por defecto ordena de manera ascendente, esto es, caracteres de la "A" a la "Z", números enteros positivos del "1" en adelante, fechas de las más antiguas a las más recientes, etc. A continuación de cada campo de la lista de ordenamiento puede colocarse una palabra clave que refuerce

el orden ascendente o le invierta a su forma descendente. Esta palabra clave es "ASC" para indicar ordenamiento ascendente, o "DESC" para indicar lo contrario. En nuestro ejemplo, si en lugar de ordenar las ciudades alfabéticamente deseáramos hacerlo en forma descendente, la instrucción sería: "ORDER BY Ciudades.Ciudad DESC;"

Consultas de agrupamiento

La figura 33 muestra una consulta similar a una de selección simple, pero al ser detallada con más precisión debe lucir ligeramente diferente. Sólo utiliza la tabla *Libros* y su intención es obtener el promedio de los precios de los libros que se corresponden con cada tema (o más específicamente con cada valor del campo Id_Tema).

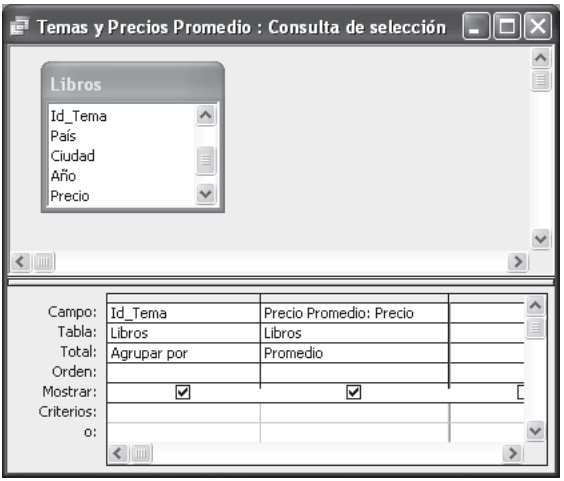


Figura 33. Consulta para el precio promedio de libros, por temainstrucción SQL

Lo primero a resaltar de la figura 33 es que ahora aparece una sección denominada "Total". En ella se han seleccionado las opciones "Agrupar por" para el campo "Id_Tema" y "Promedio" para el campo "Precio", cuyo nombre será cambiado por "Precio Promedio".

Esta sección del área destinada a configurar las condiciones de la consulta se obtiene cuando se presiona el botón "Totales" [6], iconizado con un símbolo de sumatoria en la barra de herramientas de la figura 29.

Todas las demás secciones continúan estando disponibles, así que aquí también se puede ordenar o establecer criterios de tamizado.

La instrucción SQL resultante es la siguiente:

```
SELECT Libros.Id_Tema, Avg(Libros.Precio)
AS [Precio Promedio] FROM Libros
GROUP BY Libros.Id_Tema;
```

Dicho en palabras comunes, estamos indicando al SMBD que agrupe los registros de la tabla *Libros* de acuerdo con el contenido del campo "Id_Tema" y, dentro de cada grupo, obtenga el promedio de sus precios utilizando la función *AVG( )*. A este promedio, bautícelo y preséntelo como "Precio Promedio".

Las funciones de "dominio agregado" que admite el SQL, son las siguientes:

- Suma (*SUM*). Esta función retorna la suma aritmética simple de los valores del campo pasado como parámetro, considerada para todos los elementos dentro de cada grupo resultante en el agrupamiento definido por la cláusula *GROUP BY*.
- Promedio (*AVG*). Retorna el promedio aritmético simple del campo pasado como parámetro, es decir, el cociente entre la suma de los valores del campo y el número de ellos que participan en la mencionada suma, considerado para todos los elementos dentro de cada grupo resultante.
- Mínimo (*MIN*). Devuelve el mínimo valor del campo pasado como parámetro, de entre aquellos elementos que participan en cada grupo.
- Máximo (*MAX*). Retorna el máximo valor del campo pasado como parámetro, de entre aquellos elementos que participan en cada grupo.
- Cuenta (*COUNT*). Cuenta el número de elementos resultantes en cada grupo.
- Desviación estándar (*STDEV*). Retorna el valor del estimador de la desviación estándar muestral, según como se le define estadísticamente, es decir, utilizando como denominador el número de datos menos 1. La desviación estándar es una medida de la dispersión de los datos con respecto a su media y representa un importante estadístico para el estudio de la variabilidad de un conjunto de datos. También es posible, desde la forma de sin-

taxis SQL (no desde el asistente), sustituir este estadístico por su equivalente poblacional (*STDEVP*). En tal caso, el denominador de la expresión es el número total de datos sin restar 1.

- Varianza (*VAR*). Devuelve el valor del estimador muestral de la varianza, definida estadísticamente como el cuadrado de la desviación estándar muestral. La varianza también es una medida de la dispersión de los datos con respecto a su media, pero arroja un resultado siempre positivo y sus unidades de medida ya no corresponden a los datos que le dan origen. También en este caso es posible, desde la forma de sintaxis SQL, sustituir el estadístico por su equivalente poblacional (*VARP*). En tal caso, el cuadrado se toma a partir de la desviación estándar poblacional.
- Primero (*FIRST*). Retorna el primer elemento del campo pasado como parámetro, de entre aquellos que participan en cada grupo. No debe confundirse esta función con la función *MIN*. Aquí, el elemento retornado es el que aparece en primer lugar de la lista de valores del campo, que no necesariamente se encuentran ordenados en forma ascendente, mientras que con *MIN* se retorna el elemento cuyo valor sea mínimo no importa el lugar que ocupe en el conjunto.
- Último (*LAST*). Devuelve el último elemento del campo pasado como parámetro, de entre aquellos que participan en cada grupo. Una vez más, no debe confundirse esta función con la función *MAX*. Aquí, el elemento retornado es el que aparece en último lugar de la lista de valores del campo, que no necesariamente se encuentran ordenados en forma ascendente, mientras que con *MAX* se retorna el elemento cuyo valor sea máximo sin importar el lugar que ocupe en el conjunto.

Las funciones de dominio agregado descritas pueden utilizarse simultáneamente en una instrucción, por ejemplo, si deseáramos conocer no sólo el promedio de los precios para cada tema, sino también el número de libros que pertenecen a cada grupo (o tema), la instrucción podría ser entonces:

```
SELECT Libros.Id_Tema, Avg(Libros.Precio)
AS [Precio Promedio], Count(Libros.Id_Libro)
AS [N° de Libros] FROM Libros
GROUP BY Libros.Id_Tema;
```

Nótese que la función *COUNT* se ejecuta pasando como parámetro el campo "Id_Libro" y no el campo "Precio". Para que la enumeración de los registros de un grupo sea correcta, siempre es menester contar sobre un campo que les identifique inequívocamente, pues en caso contrario se corre el riesgo de que la cuenta excluya aquellos valores iguales en el campo. Con esto en mente, es claro que no resulta correcto contar sobre los precios, ya que ellos pueden coincidir para varios registros de un grupo.

También pueden utilizarse expresiones de cálculo definidas por el usuario optando en la sección "Total" por el valor "Expresión". Además, cuando un campo no será objeto de agrupamiento sino que se utilizará como criterio de filtrado de registros, debe optarse por el valor "Dónde" en la sección "Total".

La figura 34 despliega el resultado de la consulta que hemos discutido incluyendo la enumeración de los libros que pertenecen a cada grupo.

Temas y Precios Promedio : Consulta de selección

Id_Tema	Precio Promedio	Nº de Libros
BDATO	Bs 30.064,57	21
BIOLO	Bs 24.203,50	2
COMIN	Bs 24.570,00	1
COMPU	Bs 29.375,50	14
CONCA	Bs 29.543,14	7
DICCI	Bs 30.745,36	14
ECONM	Bs 28.961,13	8
ECONO	Bs 25.516,50	4
ECOPO	Bs 28.725,50	4
EDUCA	Bs 31.722,29	7
ESTAD	Bs 30.128,23	110
FILOS	Bs 28.489,10	10
FISIC	Bs 32.632,50	4
GEOME	Bs 15.071,00	1

Registro: 28 de 46

Figura 34. Resultado del cálculo del promedio de precios y enumeración de libros, agrupando por temas en la tabla Libros

Los precios que se muestran en la figura 34 corresponden al promedio del precio de cada libro dentro del grupo al que pertenece, y el valor "Nº de Libros" a su enumeración. Por ejemplo, del tema cuyo "Id_Tema" es "BDATO" (que corresponde al tema "Base de Datos") podemos decir que los libros presentes en nuestro sistema son 21 y el precio promedio de todos ellos es Bs. 30.064,57. También apreciamos en la figura que nuestro sistema tiene muchos libros dedicados al tema

cuyo "Id_Tema" es "ESTAD" (de hecho, se trata del tema "Estadística"); en total, 110 libros tocan este tema. Podríamos notar también que los libros más económicos de la ventana, en promedio, son los correspondientes a los temas con identificadores "BIOLO" (Bs. 24.203,50) y "COMIN" (Bs. 24.570,00).

Examinemos a continuación una consulta con elementos un tanto más complejos, como la que se muestra en la figura 35.

Figura 35. Libros adquiridos entre fechas, agrupados por año y mes

La instrucción SQL resultante es:

```
SELECT Month([Fecha_Adquisición]) AS Mes,
Count(Libros.Id_Libro) AS [N° de Libros
Adquiridos] FROM Libros WHERE (((Libros.
Fecha_Adquisición) Between #1/1/2002# And
#12/31/2002#)) GROUP BY Year([Fecha_Adquisición]),
Month([Fecha_Adquisición]);
```

La consulta de la figura 35 contiene varias novedades:

1. Se utilizan dos funciones muy comunes para el tratamiento de fechas, a saber, *Month* (Mes) y *Year* (Año). Estas funciones obtienen el mes y el año en términos numéricos de una fecha pasada como parámetro, es decir, números del 1 al 12 correspondientes a los meses del año y números enteros cualesquiera correspondientes a los años. Se les utiliza entonces como si fueran campos para establecer el agrupamiento, ahora a dos niveles, primero por año y, dentro de cada año, por mes. *Month* también se muestra como resultado de la consulta. Lo que ocurre es que de cada fecha se

extrae el año, el mes y, una vez conocidos estos valores, se agrupan los registros (y no antes).

2. Esta vez, el criterio de tamizado o filtrado tiene que ver con un campo de fecha (Fecha_Adquisición). La forma de instruir al gestor de consultas para que filtre los registros entre dos fechas dadas utiliza la frase clave *Between* <Fecha1> *And* <Fecha2> (en español, Entre <Fecha1> Y <Fecha2>). Se sobreentiende que Fecha1 debe ser anterior que Fecha2, de lo contrario no se mostrará ningún registro. Otro aspecto notorio es que en la figura 35 se han escrito las fechas entre símbolos de numeral ("#") en formato castellano, esto es "día/mes/año". Esta es la manera en que se le indica al gestor de consultas que el dato en cuestión corresponde a una fecha. No obstante, cuando examinamos la instrucción SQL correspondiente nos encontramos con que las fechas aparecen en formato inglés, esto es, "mes/día/año". La razón del cambio es que el control de fechas es una utilidad que depende del sistema operativo, sólo para efectos de presentación, pero cada vez que el computador procesa una fecha, lo hace en formato anglosajón.

El resultado de la consulta desplegada en la figura 35 lo encontramos a continuación en la figura 36.

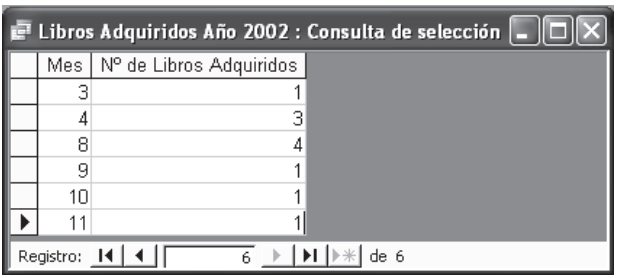


Figura 36. Nº de libros adquiridos por mes durante el año 2002

De la figura 36 comentaremos que no todos los meses del año aparecen en ella. Efectivamente, el gestor de consultas no presenta información de los meses en los que no se adquirieron libros. Adicionalmente, conociendo que los datos corresponden al año 2002 (lo que puede deducirse del título), no fue necesario incluir este dato en los resultados.

Por último comentaremos que existe otra forma de hacer esta consulta de manera más simple. Efectivamente, el tamizado de los registros se hizo en función de la fecha de adquisición de los libros buscando que participaran aquellos adquiridos entre el primer y el último día del año 2002. Más simple hubiese resultado tamizar a partir de la evaluación de la función *Year* estableciendo el criterio como "2002". La respuesta sería exactamente la misma que obtuvimos antes.

Consultas de acciones

Como mencionamos, las consultas de acciones tienen el propósito de alterar el contenido de la base de datos. Se permiten eliminaciones de registros (y tablas completas), modificaciones o actualizaciones de los datos de uno o varios registros a la vez e inserción de nuevos registros. Para ejemplificar, supongamos que nos hemos dado cuenta de que los descriptores de nuestros libros deberían contener una referencia automática al tema que tratan. La figura 37 muestra los descriptores de tres libros correspondientes al tema "Bases de Datos" (Id_Tema="BDATO").

En la figura 37 se aprecian los descriptores de los libros cuyos identificadores son 1, 2 y 39, en ese orden, que pertenecen al grupo de libros que tocan el tema cuyo identificador es "BDATO".

Descriptores_Libros_BD : Consulta de selección

Id_Libro	Id_Tema	Descriptor
1	BDATO	DB2
1	BDATO	Modelo Relacional
1	BDATO	SQL
2	BDATO	Modelo Conceptual
2	BDATO	Modelo Relacional
2	BDATO	SQL
39	BDATO	Microsoft Access

Registro: 11 de 42

Figura 37. Descriptores de algunos libros de tema Base de Datos

La figura 38 muestra la consulta que produce el resultado de la figura 37. La instrucción SQL para los efectos, es la siguiente:

```
SELECT Libros.Id_Libro, Libros.Id_Tema,
[Descriptores de Libros].Descriptor
FROM Libros INNER JOIN [Descriptores de Libros]
ON Libros.Id_Libro = [Descriptores de Libros].
Id_Libro WHERE (((Libros.Id_Tema)="BDATO"));
```

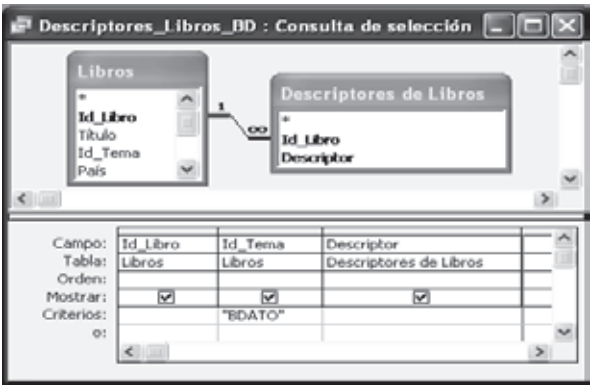


Figura 38. Consulta para obtener los descriptores de libros con Id_Tema="BDATO"

Se trata de una consulta simple que a partir de las tablas *Libros* y *Descriptores de Libros*, vinculadas por el identificador del libro, muestra los descriptores de cada libro que pertenezca al tema "Base de Datos".

Ahora bien, en la figura 37 puede verse que los tres libros mostrados, aun cuando pertenecen al tema "Bases de Datos", no tienen un descriptor que incluya esta frase. Pensamos entonces que para que los descriptores cumplan su propósito, describir de lo que un libro trata, haría falta agregar como descriptor, además de otros muchos posibles, el tema establecido para el libro (lo que está haciendo falta).

La figura 39 muestra ahora el número de descriptores de los tres libros del ejemplo.

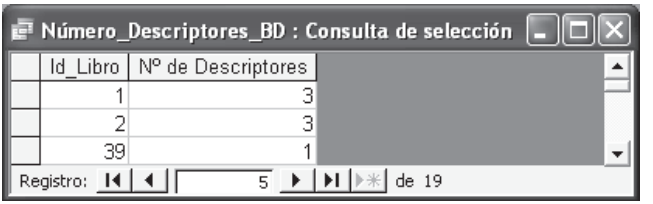


Figura 39. Nº de descriptores de los libros con Id_Tema = "BDATO"

La figura 40 representa la consulta que resulta en la 39.

La instrucción SQL correspondiente a la consulta de la figura 40, es la siguiente:

```
SELECT Descriptores_Libros_BD.Id_Libro,
Count(Descriptores_Libros_BD.Id_Libro) AS
[N° de Descriptores] FROM Descriptores_Libros_BD
GROUP BY Descriptores_Libros_BD.Id_Libro;
```

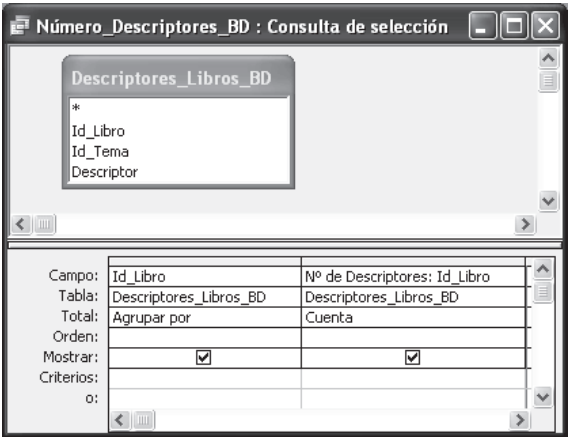


Figura 40. Consulta del Nº de descriptores de los libros con Id_Tema="BDATO"

Una consulta de agrupamiento que cuenta registros a partir de identificadores que luciría simple si no fuera porque en nuestra base de datos no existe una tabla denominada "Descriptores_Libros_BD". Lo que si está presente en nuestra base de datos es una consulta que denominamos de esa forma, así que esta nueva consulta de datos agrupados llamada "Número_Descriptores_BD", no está basada en una tabla, sino que lo está en otra consulta persistente (es decir, una consulta que se deja permanentemente en la base de datos) lo cual, dicho sea de paso, significa que el tratamiento requerido para hacer una consulta a partir de otra, es exactamente el mismo que se daría para hacer una consulta a partir de una tabla.

Retomando nuestro ejemplo que tiene que ver con descriptores y consultas de acción, ahora que ya sabemos que los libros del tema "Base de Datos" no contienen esta frase como descriptora (ver figura 37) y que en este momento los libros 1, 2 y 39 tienen 3, 3 y 1 descriptores respecti-

vamente (ver figura 39), nos proponemos el plan de agregar a la tabla de descriptores de libros, una frase que sea idéntica al tema de cada libro para aquellos ejemplares del tema "Bases de Datos".

Hacerlo manualmente requeriría un buen esfuerzo, así que nuestro plan es hacerlo automáticamente. Al invocar el diseño de una nueva consulta se despliega el diagramador, como hemos explicado con anterioridad. Como también explicamos antes, en la barra de herramientas para el diseño de consultas (figura 29) se dispone de un botón para seleccionar el tipo de consulta que deseamos construir, denominado "Tipo de consulta" [3]. Para construir una consulta que nos permita agregar o anexar nuevos registros a la base de datos, debe seleccionarse este botón. Al hacerlo se despliega una lista contentiva de todos los tipos de consulta que pueden construirse desde el asistente para consultas del *Access*. Esta lista se muestra en la parte (a) de la figura 41. La opción resaltada, "Consulta de datos anexados", es la que permite la operación que nos ocupa.

Al seleccionar el tipo de consulta como "Consulta de datos anexados", tal como se esboza en la figura 41(a), se despliega inmediatamente el cuadro de diálogo mostrado en la 41(b). Este cuadro de diálogo está destinado a que el diseñador escoja la tabla a la que desea anexar nuevos registros, en nuestro caso la tabla *Descriptores de Libros* que se encuentra en la base de datos que estamos utilizando (como puede verse en la figura 41(b), también es posible anexar registros a tablas que estén en otra base de datos diferente a la que se encuentre activa para el momento).

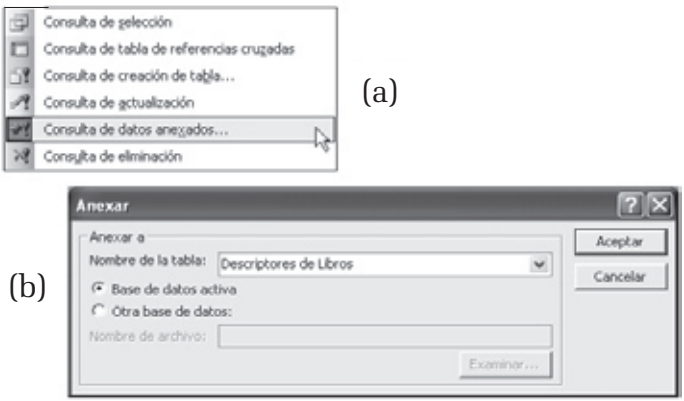


Figura 41. (a) Tipos de consulta desde el asistente y
(b) Selección de la tabla a la que se desea anexar nuevos datos

La figura 42 muestra, completados los pasos anteriores, la consulta para anexar registros a la tabla *Descriptores de Libros*, tal como traza el plan que nos hemos propuesto.

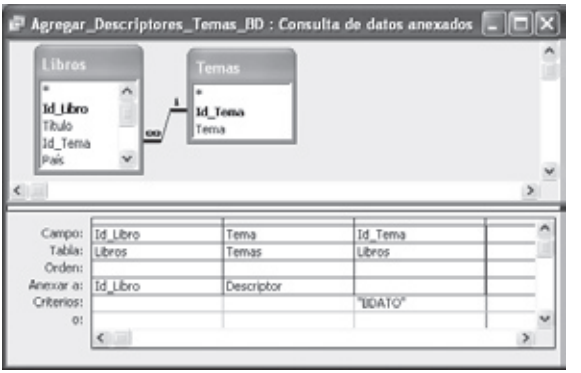


Figura 42. Consulta para anexar datos a la tabla Descriptores de Libros

Al seleccionar la tabla *Descriptores de Libros* y especificar que deseamos agregarle registros se activa una sección del asistente para consultas denominada "Anexar a". En esta sección están disponibles todos los campos de la tabla a la que deseamos agregar registros. Para nuestro caso, como se ve en la figura 42, deseamos anexar valores a los campos "Id_Libro" y "Descriptor", por supuesto, en la tabla *Descriptores de Libros*.

Lo que anexaremos será el contenido del campo "Id_Libro", de la tabla *Libros*, en el campo "Id_Libro" de la tabla *Descriptores de Libros*, así como el contenido del campo "Tema", de la tabla *Temas*, en el campo "Descriptor" de la tabla *Descriptores de Libros*. Esto ocurrirá para todos aquellos libros vinculados con sus temas, de manera que el identificador del tema de la tabla *Libros* contenga el valor "BDATO", como puede constatarse en la figura 42.

La correspondiente instrucción SQL producida por el diagramador de consultas esbozado en la figura 42, es la siguiente:

```
INSERT INTO [Descriptores de Libros] ( Id_Libro,
[Descriptor] ) SELECT Libros.Id_Libro, Temas.Tema
FROM Temas INNER JOIN Libros ON Temas.Id_Tema =
Libros.Id_Tema WHERE (((Libros.Id_Tema)="BDATO"));
```

Para completar esta consulta se dan los pasos siguientes:

1. Se juntan los registros de las tablas *Libros* y *Temas* de acuerdo con la igualdad del identificador del tema que ambas tablas comparten.
2. Se desechan todos aquellos registros que no cumplen la condición de tener como identificador del tema la cadena de caracteres "BDATO".
3. De los registros resultantes se desechan todos los campos excepto el campo "Id_Libro" (de la tabla *Libros*) y el campo "Tema" (de la tabla *Temas*).
4. La lista de datos resultante, contentiva de los dos atributos que quedaron, se agrega a la tabla *Descriptores de Libros* haciendo corresponder su campo "Id_Libro" con el primer elemento de la lista resultante (Libros.Id_Libro), y su campo "Descriptor" con el segundo (Temas.Id_Tema).

Cuando se "corre" una consulta de acciones no se producen resultados visibles inmediatamente, de hecho, los únicos resultados que podríamos ver en este caso son mensajes de advertencia sobre que se insertarán nuevos registros. Para comprobar que la acción se realizó exitosamente debemos examinar la tabla que ha debido ser modificada en su contenido. La figura 43(a) muestra el resultado de la consulta "Descriptores_Libros_BD" (utilizada antes en las figuras 37 y 38) aplicada sobre lo que creemos es un nuevo conjunto de datos. Asimismo, la figura 43(b) muestra el resultado de la consulta Número_Descriptores_BD" (utilizada antes en las figuras 39 y 40), que cuenta nuevamente los descriptores de cada libro.

Figura 43. (a) Descriptores de los libros con Id_Tema = "BDATO" y (b) Número de descriptores de los libros con Id_Tema = "BDATO"

Tuvimos éxito. La figura 43, partes (a) y (b), demuestra que los libros cuyos identificadores son 1, 2 y 39 tienen un nuevo descriptor, "Bases de Datos", y de hecho, ahora estos libros cuentan con 4, 4 y 2 descriptores en lugar de los 3, 3 y 1 que tenían antes.

De esta forma se procede en todos los casos de inserción de nuevos registros. Nuestro plan era ambicioso, pues no sólo agregó registros cualesquiera o valores constantes (lo cual, por supuesto, es posible) sino que lo hizo extrayendo el contenido de los temas asociados con los libros. La cláusula SQL *INSERT INTO* <Tabla>(<Lista de campos>), seguida de una consulta de selección simple que contendría los datos a insertar, en el orden de la lista de campos, lo consiguió.

Supongamos ahora que nos hemos arrepentido de tal hazaña y estamos en el dilema de tener que eliminar todos los registros de la tabla *Descriptores de Libros* cuyo contenido sea la cadena de caracteres "Bases de Datos", para aquellos libros cuyo campo "Id_Tema" sea "BDATO". Un arrepentimiento como este es muy común en la profesión. Puede ser por ejemplo que el usuario de la base de datos indique que prefiere los temas por un lado y los descriptores por otro, o puede ser que opine que la descripción del tema no es buen descriptor del libro. Para ejecutar esta acción se necesita otro tipo de consulta. En la figura 41(a) debe escogerse entonces la opción "Consulta de eliminación". La figura 44 muestra el asistente con la consulta de eliminación deseada.

La instrucción SQL que se configura a partir de la figura 44, es la siguiente:

```
DELETE [Descriptores de Libros].* FROM Temas INNER
JOIN (Libros INNER JOIN [Descriptores de Libros]
ON Libros.Id_Libro = [Descriptores de Libros].
Id_Libro) ON Temas.Id_Tema = Libros.Id_Tema
WHERE ((([Descriptores de Libros].Descriptor)=
[Temas].[Tema]) AND ((Libros.Id_Tema)="BDATO"));
```

En este caso, como puede verse en la figura 44, se habilita una nueva sección en el área de trabajo rotulada como "Eliminar". Allí pueden escogerse sólo dos alternativas, "Desde" y "Donde". La palabra clave "Desde" se refiere a los registros que serán eliminados, en nuestro caso todos los campos de los registros de la tabla *Descriptores de Libros* (el "*" es el responsable del "todos"). La palabra clave "Dónde" impone condiciones o restricciones a la eliminación, en nuestro caso

fundamentales, pues si no eliminaríamos todos los registros y de ningún modo queremos hacer eso.

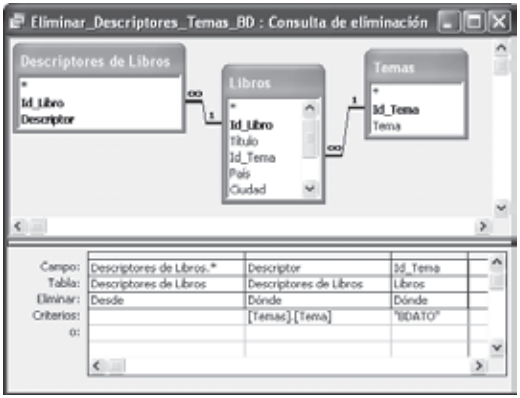


Figura 44. Consulta de eliminación de registros en Descriptores de Libros con condiciones para Descriptor e Identificador del Tema

Las tres tablas que participan en esta consulta se encuentran vinculadas de acuerdo con el esquema de la base de datos. Los descriptores con los libros a través del identificador del libro y los libros con los temas a través del identificador del tema. Ello se desprende de la cláusula *FROM* y también del diagrama de tablas desplegado en la figura 44.

Lo único novedoso en esta consulta es que una parte de la condición lógica que imponemos debe cumplir el conjunto de registros a ser eliminado, contiene a su vez no un valor, como hasta ahora habíamos hecho, sino un campo de otra tabla. Es el caso de la sección "[Descriptores de Libros].Descriptor = [Temas].[Tema]" en la cláusula WHERE.

En el esquema de nuestra base de datos no hay asociación alguna entre las tablas *Descriptores de Libros* y *Temas*. ¿Cómo sabe entonces el SMBD a qué temas nos referimos? Lo sabe gracias a que la cláusula *FROM* ha establecido una relación entre ellas a través de los libros. Así pues, la consulta desplegada en la figura 44 se procesa de la siguiente manera:

1. Se juntan los registros de las tablas *Libros* y *Descriptores de Libros* según compartan o no los valores del campo "Id_Libro". De esta reunión se toman todos los campos en ambas tablas. Llamemos a este resultado parcial C_1.

2. Se juntan los registros de la tabla *Temas* y el resultado parcial C_1 según compartan o no los valores del campo "Id_Tema". Nótese que este campo efectivamente se encuentra en el resultado parcial C_1 que aporta la tabla *Libros*. De esta reunión se toman todos los campos en ambas tablas. Llamemos a este resultado parcial C_2. Ahora C_2 es una gran tabla (temporal) que contiene en secuencia todos los campos de los registros de las tablas *Descriptores de Libros*, *Libros* y *Temas*, que cumplen con las condiciones dadas.
3. Se desechan aquellos registros del resultado parcial C_2 que no satisfacen la condición lógica de que el descriptor del libro debe ser idéntico a la descripción del tema y (*AND*) que el identificador del tema del libro debe ser igual a la constante "BDATO". Llamemos a este resultado parcial C_3.
4. Algunos campos del resultado parcial obtenido en C_3 se utilizan ahora como una lista de los valores que se tratarán de hacer coincidir con la tabla *Descriptores de Libros*. Aquellos registros que coincidan en ambos conjuntos (de acuerdo con sus campos "Id_Libro" y "Descriptor"), serán eliminados de la tabla *Descriptores de Libros*.

Puede sonar algo complejo sin serlo realmente. Lo que está ocurriendo es que se conjugan una serie de productos cartesianos entre los conjuntos de registros que componen las tablas, a los que luego se les aplican operaciones restrictivas de proyección.

El último tipo de consulta de acción que estudiaremos es aquella que nos permite modificar el contenido de una tabla. Se logra construir este tipo de consultas al seleccionar la opción "Consulta de actualización" dentro del asistente para consultas y su barra de herramientas, como se muestra en la figura 41(a).

Supongamos que finalmente estamos de acuerdo en dejar como descriptor de los libros de tema "Bases de Datos" dicha frase, pero decidimos que las frases utilizadas como descriptores de libros deben estar escritas con todas sus letras en minúsculas.

La figura 45 muestra la solución a este nuevo dilema, creando una consulta de actualización llamada "Cambia_Descriptor_Min_BD".

Ahora, como se desprende de la figura 45, una nueva sección se encuentra presente: "Actualizar a". En ella se debe incluir, coincidiendo con el campo cuyo contenido será alterado, el nuevo valor que habrá

de contener. Este nuevo valor será dispuesto atendiendo al criterio deseado. En este caso buscamos que todos los registros de la tabla *Descriptores de Libros* que contengan como valor del campo "Descriptor" la frase "Bases de Datos" (escrita con algunas mayúsculas), alteren su contenido sustituyéndolo por la frase "bases de datos" (en minúsculas).

La instrucción SQL correspondiente a la figura 45, es la siguiente:

```
UPDATE [Descriptores de Libros] SET [Descriptores
de Libros].[Descriptor] = "bases de datos"
WHERE ((([Descriptores de Libros].Descriptor)=
"Bases de Datos"));
```

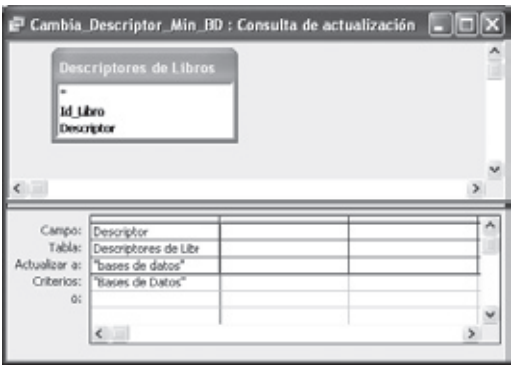


Figura 45. Actualización del descriptor "Bases de Datos" por "bases de datos"

La cláusula SQL importante es *UPDATE - SET*. A continuación de la primera palabra clave se indica la tabla que será objeto de actualización y, luego de la segunda palabra clave, la lista de campos que recibirán nuevos valores, por supuesto, con dichos nuevos valores. A todos los registros que serán afectados por el cambio se les tamiza de acuerdo con la condición lógica incluida en la cláusula *WHERE*.

Figura 46. Descriptores de Libros con la frase "bases de datos" escrita en letras minúsculas

El resultado que se ilustra con la figura 46, al compararse con la figura 43(a) muestra que la instrucción de actualización ejecutada logró su cometido, pues todas las frases "bases de datos" del campo "Descriptor" están ahora escritas en letras minúsculas.

Más sobre consultas

Todos los tipos de consultas estudiados antes son susceptibles de recibir parámetros como parte de sus expresiones SQL. Los parámetros son muy útiles cuando se desea que una misma consulta, estructurada sobre la base de una constante, pase a estarlo sobre la base de una variable que nuestro usuario pueda cambiar a voluntad cada vez que la ejecute. Por ejemplo, lo establecido antes para los descriptores de libros, en el caso de aquellos libros cuyo tema era "Base de Datos", tanto en la inserción como en la eliminación o actualización, perfectamente podría generalizarse ahora parametrizando el identificador del tema en lugar de preestablecerlo como "BDATO".

Cuando se agrega un parámetro a una consulta, el gestor de consultas detiene su ejecución hasta tanto el usuario no haya incorporado un valor para dicho parámetro. Una vez recibido un valor para el parámetro, el gestor de consultas sustituye apropiadamente dicho valor en aquellos lugares de la instrucción SQL donde se encuentra el parámetro.

Los valores que se pasan como parámetros a una consulta pueden colectarse de dos maneras, a saber: directamente en la ejecución de la consulta, mediante un cuadro de diálogo dispuesto a tal fin que controla el SMBD o, a partir de una ventana específicamente diseñada para ello por el analista, que contenga un objeto con el nombre del parámetro. Para este último caso se requiere conocer cómo se construyen "Formularios", lo que haremos más adelante. Nos concentraremos por ahora en la primera forma, la más simple, que deja todo en manos del *Access*.

Recordemos a manera de ejemplo la figura 35. En ella se construye una consulta de datos agregados que enumera los libros adquiridos entre dos fechas, agrupados por año y, dentro del año, por mes. Todo lo dicho alrededor de esta consulta se basó en que deseábamos encontrar el conjunto de libros resultante entre las fechas 01/01/2002 y

31/12/2002. Establecimos entonces la condición "Entre #01/01/2002# y #31/12/2002#".

Pero, ¿qué tal si quisiéramos encontrar la misma información para cualquier par de fechas que el usuario designe? Pues bien, necesitaríamos definir parámetros. Así, la metodología recomendada es resolver la consulta sin parámetros y, una vez demostrado que la lógica funciona, incorporar los parámetros para generalizarla.

La figura 47 muestra una composición hecha de retazos del área donde se establecen los criterios, en una consulta que contiene sólo la tabla *Libros* (similar a la que se ve en la figura 35). Hemos preferido mostrar una composición en las secciones (a), (b) y (c) por razones de espacio, pero (a), (b) y (c) van en realidad una después de la otra, como en todas las consultas.

De la figura 47 podemos inferir cómo se ha construido esta consulta y para qué se espera utilizarla. En la parte (a) de la figura se aprecia que los campos que se muestran son el año de adquisición, el mes de dicho año y el número de libros adquiridos en ese mes, convenientemente renombrados luego de utilizar las funciones Año (*Year*), Mes (*Month*) y la cuenta (*Count*) sobre el agrupamiento. Hasta aquí no existe mucha diferencia con la consulta que utilizáramos en la figura 35.

En la parte (b) de la figura 47 se aprecia la forma en que incorporamos parámetros en lugar de las fechas constantes, entre las que se desea mostrar la información. Se ven dos parámetros encerrados entre corchetes. Sabemos que son tales porque se encuentran entre corchetes y no corresponden a nombres de campos de la tabla. Estos parámetros son "[Escriba la Fecha Inicial (dd/mm/aa):]" y "[Escriba la Fecha Final (dd/mm/aa):]".

Podemos poner cualquier nombre a un parámetro, así que decidimos asignar uno que sea explicativo para cuando llegue la hora de pedir su valor al usuario. El primero de ellos contendrá la fecha inicial para la consulta, y el segundo, la fecha final.

Agregamos dos condiciones más a la consulta, como se muestra en la figura 47(c). Estas últimas condiciones contemplan el caso de que el usuario desee mostrar todos los registros posibles y no sólo los que se encuentren entre las fechas dadas. Si es así, debe dejar sin valor los parámetros y el SMBD entenderá que desea mostrarlo todo.

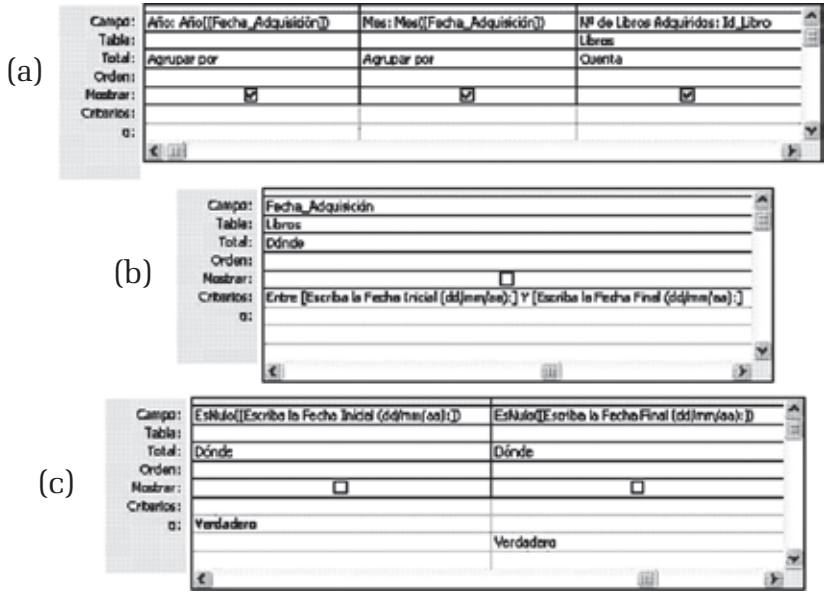


Figura 47. Definiciones de una consulta entre dos fechas, con parámetros

En la figura 48 se ilustra cómo deben completarse las características de los parámetros que se incorporan a una consulta. La figura 48(a) muestra un menú emergente que se presentará en la pantalla cuando demos clic al botón derecho del ratón sobre el área de tablas del asistente para consultas. Muchas de las cosas que hemos hecho hasta ahora pueden hacerse también desde este menú, sin embargo, especialmente nos interesa invocar aquí la opción "Parámetros…". Al hacerlo se activa la ventana que se muestra en la figura 48(b). Nótese que definir un parámetro requiere escribir nuevamente su nombre (exactamente igual que como se le está usando en la consulta), seguido del tipo de dato que le corresponde.

La instrucción SQL de la consulta de la figura 47, es la siguiente:

```
PARAMETERS [Escriba la Fecha Inicial (dd/mm/aa):]
DateTime, [Escriba la Fecha Final (dd/mm/aa):]
DateTime; SELECT Year([Fecha_Adquisición]) AS Año,
Month([Fecha_Adquisición]) AS Mes, Count(Libros.
Id_Libro) AS [N° de Libros Adquiridos] FROM Libros
WHERE (((Libros.Fecha_Adquisición) Between
[Escriba la Fecha Inicial (dd/mm/aa):] And
[Escriba la Fecha Final (dd/mm/aa):]))
```

```
OR (((IsNull([Escriba la Fecha Inicial
(dd/mm/aa):]))=True)) OR (((IsNull([Escriba
la Fecha Final (dd/mm/aa):]))=True))
GROUP BY Year([Fecha_Adquisición]),
Month([Fecha_Adquisición]);
```

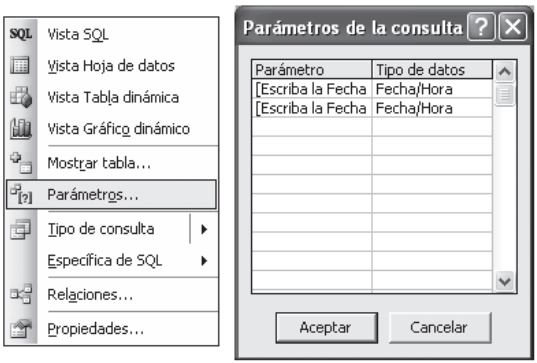


Figura 48. Definición de parámetros en una consulta

Esta instrucción no es muy diferente a la que resultaba a partir de la figura 35. Difiere de aquella en que ahora se añade la cláusula *PARAMETERS,* dedicada a establecer la definición de los parámetros que mencionamos antes. Otra diferencia es que en lugar de emplear los valores constantes de fechas dadas allí, ahora van los parámetros. Los dos últimos miembros de la condición lógica establecida en la cláusula *WHERE* otorgan la posibilidad de que, dejando en blanco los valores de los parámetros, no ocurra filtrado alguno de los registros. Por lo demás, el procesamiento de esta consulta es idéntico al que explicamos antes. Cuando se le ejecute, mostrará dos cuadros de diálogo consecutivos que interrogan sobre la fecha inicial y final. Tales diálogos se muestran ya rellenos en la figura 49, partes (a) y (b).

El primer diálogo que se muestra al usuario se esboza en la figura 49(a). El nombre del primer parámetro aparece como la descripción del campo que debe llenarse, explicando al usuario el valor que de él se espera. El cuadro en cuestión aparece vacío y debe completarse con una

fecha cualquiera, que en este ejemplo es "01/01/04". El segundo diálogo se muestra al usuario luego de que éste ha dado clic sobre el botón "Aceptar" del primero. La figura 49(b) esboza tal diálogo muy similar al primero, pero que ahora espera una segunda fecha o fecha final. Hemos introducido en él la fecha "31/12/04". Al "Aceptar", luego de introducidos los valores señalados, el SMBD continúa con la ejecución de la consulta, sustituyendo los valores donde corresponda.

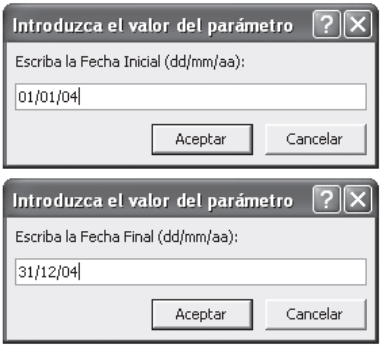


Figura 49. Diálogos para introducir parámetros

Un último aspecto que mencionaremos sobre las consultas es que no todas ellas pueden lograrse a partir del asistente o diagramador de consultas. Una consulta que no puede hacerse desde el asistente es la consulta de unión.

La unión de dos tablas (o consultas) es una operación que toma los registros de la primera y añade los de la segunda a continuación. Evidentemente debe tratarse de dos tablas con los mismos campos, de lo contrario, la consulta de unión fallará. Como toda consulta, los datos no se guardan permanentemente. Cada vez que se le invoca se revisa la base de datos y se conforma el resultado a partir de las fuentes establecidas.

Como ejemplo suponga que a partir de dos listas contentivas de los primeros y segundos autores de libros quisiéramos construir una sola uniendo ambos conjuntos. Para ello podríamos utilizar, entre otras estrategias, una consulta de unión. La primera lista, que guardaremos con el nombre "Primeros Autores", se obtiene con la siguiente instrucción:

```
SELECT [Autores de Libros].[Orden de Autoría],
[Apellidos] & ", " & [Nombres] AS Autor FROM
Autores  INNER JOIN [Autores de Libros] ON
Autores. Id_Autor = [Autores de Libros].Id_Autor
WHERE ((([Autores de Libros].[Orden de
Autoría])=1));
```

La segunda lista, que guardaremos con el nombre "Segundos Autores", se obtiene con la siguiente instrucción SQL, casi idéntica a la primera:

```
SELECT [Autores de Libros].[Orden de Autoría],
[Apellidos] & ", " & [Nombres] AS Autor FROM
Autores INNER JOIN [Autores de Libros] ON
Autores.Id_Autor = [Autores de Libros].Id_Autor
WHERE ((([Autores de Libros].[Orden de
Autoría])=2));
```

Ambas consultas tienen un número (1 o 2 respectivamente) en el campo "Orden de Autoría", y un campo rotulado como "Autor", que contiene una composición o concatenación de los apellidos y los nombres de los autores, con una coma (",") de por medio. Nótese que la forma de concatenar cadenas de caracteres en *Access* utiliza el operador "&". Ahora bien, si quisiéramos juntar la información de los primeros y segundos autores en una sola lista ordenada por autor, deberíamos dar la instrucción SQL que sigue:

```
SELECT * FROM [Primeros Autores] UNION ALL
SELECT * FROM [Segundos Autores] ORDER BY Autor;
```

De la lista de los primeros autores se toman todos sus campos y se unen, sin evitar repeticiones (lo que se consigue con la palabra clave *ALL* de la cláusula *UNION*), a los de la segunda lista. Finalmente, todo se ordena por el campo "Autor". Se obtiene entonces una sola estructura que contiene la información de los primeros y segundos autores.

La clave de todo sistema de información con enfoque de bases de datos relacionales está en un buen diseño del esquema de la base de datos primero, y luego, en un dominio importante de las consultas, con su lenguaje de consulta estructurado.

3.6 Interfaces de captura y presentación de datos. Formularios

Un "formulario" en *Access* es una ventana que el constructor del sistema de información puede diseñar según su criterio, principalmente para capturar los datos que formarán parte de la base de datos o presentarlos en la pantalla del computador.

Hay dos tipos de formularios: vinculados y no vinculados. Los primeros se enlazan con, o se basan en, una fuente de datos (tabla o consulta), mientras que los segundos no dependen de ninguna fuente. Los formularios vinculados se utilizan para proporcionar una interfaz al usuario, desde la que pueda introducir (o recuperar) datos que van hacia la fuente (o provienen de ella). Los no vinculados se utilizan para manejar datos que no dependen de una tabla o consulta de la base de datos y, en general, se emplean con múltiples propósitos.

Diseñar un formulario significa entonces configurar una ventana al estilo *Windows.* Para tales efectos, *Access* proporciona formularios que contienen secciones, a las cuales se pueden incorporar objetos "visuales", disponerlos de múltiples maneras y alterar su aspecto, tamaño y demás características. Prácticamente todos los elementos de un formulario, el formulario mismo, los registros de la fuente de datos -si está vinculado-, las secciones y sus objetos, tienen propiedades que pueden establecerse a voluntad de manera que la interfaz se comporte como el diseñador lo prefiera. Adicionalmente, un formulario puede contener código de programación *Visual Basic para Aplicaciones*, inmerso en un módulo cuyo ámbito es el mismo formulario, y gracias a lo cual se transforma en una herramienta de interfaz sumamente versátil y adaptable.

Los formularios vinculados tienen una manera estándar de comportarse frente a los datos que se vinculan. Con estos formularios se puede acceder a un registro de tabla o consulta, escribir sobre él alterando el valor de sus campos o incluso eliminarlo de la base de datos. El movimiento entre registros se hace gracias a una sección del formulario prediseñada para ello. Si el constructor acepta las facilidades por defecto que le proporciona *Access*, su tarea se simplifica en grado superlativo pero se atiene a las características estándar del producto. Por otra parte, si desea dar características propias al comportamiento de la interfaz, su tarea será de mayor envergadura, aun cuando puede lograr un comportamiento completamente personalizado. En este texto, dado su carácter

introductorio, nos concentraremos en el uso estándar de los formularios vinculados, variando en muy pocos aspectos su comportamiento establecido por defecto.

Los formularios son importantes, pues representan, junto con los menús, la fachada a través de la cual los usuarios del sistema de información se entenderán con él. Ya hemos visto a lo largo del texto que se pueden definir comprobaciones y validaciones de datos tanto en el diseño de los campos de cada tabla como al establecer reglas de integridad referencial, no obstante, hay muchas consideraciones adicionales y validaciones importantes, sobre todo aquellas que se establecen a partir de relaciones entre los mismos datos, que no es posible establecer en aquellas instancias. Tratándose los formularios de las ventanas a través de las cuales los usuarios verán el sistema (y la base de datos), en ellos pueden (y deben) completarse las reglas para la entrada de datos que no han podido proporcionarse en otros objetos del sistema. Así, por ejemplo, una fecha que depende de otra que se ubica en una fuente distinta, dos campos que deben ser uno superior al otro, aclaratorias de qué significan los códigos dados a un dato, en fin, todos los demás detalles de implantación que no pueden darse al sistema dentro del propio esquema de la base de datos, se dan en los formularios como parte de las características de la interfaz.

Pero tal vez el propósito más relevante de los formularios es implementar el nivel de vistas de la arquitectura de la base de datos (ver figura 9). El nivel de vistas se refiere a la óptica que de la base de datos debe darse a cada usuario del sistema, considerando que no todos los usuarios tienen que, o deben, ver los datos de la misma manera o con las mismas prerrogativas. El diseño de la interfaz y los formularios que de él se desprenden encapsulan y ocultan la globalidad de la base de datos, logrando que el usuario que no necesita o debe verla en toda su extensión, sólo la vea a través de los mecanismos específicos que se le proporcionan con ellos.

Luego el primer paso siempre será el diseño de la interfaz, que incluye las decisiones sobre qué será proporcionado por el sistema en términos de datos y de qué forma se accederá a ello en términos de opciones. Esta sección se dedica a la primera parte del diseño de la interfaz. La segunda parte se tratará más adelante, cuando se estudien los menús de opciones.

Para diseñar los formularios que contendrá la aplicación es menester ubicarse del lado del usuario que los utilizará. Transcriptores de datos, productores de informes y analistas, directivos de la organización y demás personas vinculadas al sistema de información verán la base de datos de forma distinta. Por lo tanto, el diseñador debe proporcionar a cada uno las herramientas que necesita, ajustadas a su particular forma de entenderse con el sistema.

Otra cuestión a dilucidar en el diseño de la interfaz tiene que ver con la manera más apropiada de introducir los datos. En cada aplicación particular deberán analizarse y tomarse las decisiones que favorezcan la facilidad de operación, el trabajo de transcripción, contemplen el mayor grado de sentido común y aseguren, hasta donde sea posible, la integridad de la base de datos. Por ejemplo, en nuestro sistema abocado a los *Libros* podría seleccionarse introducir cada libro a partir de su autor, sin embargo, esta decisión no parece apropiada ni muestra sentido común. Resulta mucho más apropiado introducir cada autor a partir del libro, esto es, introducir autores como atributos del libro que al revés. Esta conclusión se desprende de la observación de cómo ocurren las cosas en general y, en general, lo que un transcriptor haría sería tomar el libro, anotar sus datos y, entre ellos, sus autores. Difícilmente le veríamos buscar a partir de una lista de autores los libros de cada uno, encontrándolos en lugares que pueden estar muy dispersos para, dado el autor, introducir la información de todos sus libros al sistema.

Aunque no existen reglas absolutas sobre cómo deben diseñarse los formularios, algunos lineamientos pueden resultar útiles. Mencionemos los siguientes:

- Debe proporcionarse el número mínimo de formularios tal que todos los datos de la base de datos puedan ser introducidos sólo a partir de ellos. En general, no es buena práctica dejar que algunos datos deban introducirse sin pasar por formulario alguno, directamente en las tablas. Tampoco es buena idea incluir formularios redundantes, es decir, ventanas que hacen lo mismo o que tienen el mismo propósito. Esto último no significa que sobre una tabla o consulta no puedan hacerse distintos formularios, sólo significa que tales formularios deben responder a distintos propósitos.
- Debe mantenerse una uniformidad en la disposición y características gráficas de los objetos que componen todos los formularios. No es apropiado emplear colores muy vivos o en una gran

diversidad, tampoco lo es disponer los objetos en las ventanas de forma desordenada o cambiarlos de lugar de una ventana a la otra. Todos los formularios deberán contener el mismo conjunto de colores y los objetos que en ellos se dispongan, deberán estar ordenados y ser consecuentes en todas las ventanas. Debe recordarse que el sistema de información, aunque puede tener mucho de artístico, en realidad es una herramienta de trabajo y, por lo tanto, habrá personas que invertirán muchas horas de su tiempo operando con él. Así, claves de color como presentar los mensajes de error siempre en rojo o las advertencias en azul, deben ser mantenidas consecuentemente en todos los formularios. Fondos no muy vistosos y que no sobrecarguen la vista del usuario, son preferibles. Si se decide que los identificadores de los registros sean los primeros en aparecer en un formulario, debe mantenerse esta decisión en todos los casos, y así con todos los demás elementos.

- Hasta el máximo posible, los formularios deben proporcionar información sobre cómo habrán de utilizarse sus elementos. La idea es procurar que quien está versado en el uso de la interfaz *Windows* no deba gastar demasiado tiempo extra familiarizándose con una interfaz gráfica que, en esencia, tendría que ser muy similar. Cuanto más autoinformado esté un sistema, cuanto mayor sea el grado de sentido común que se dé a sus componentes, menor será el tiempo que invierta el usuario en acercarse a su funcionamiento y mayor el grado de provecho que del sistema obtenga, así como menor el tiempo que transcurra hasta que la aplicación sea productiva.

Para ilustrar la creación de formularios retomemos nuestra aplicación de *Libros*. Ella contiene las tablas *Libros*, *Autores*, *Autores de Libros*, *Temas*, *Países*, *Ciudades* y *Descriptores de Libros*. Obsérvense las siguientes reflexiones sobre el particular:

- Las tablas *Autores* y *Temas* son independientes de otras tablas. Ellas son necesarias en algunas, pero no necesitan otras para cumplir su propósito. Son responsables de almacenar respectivamente todos los autores posibles y todos los temas posibles, ambos ítems de datos necesarios para completar la información

sobre los libros. Así pues, estas tablas recibirán sendos formularios vinculados.

- La definición de países y ciudades, que obviamente involucra las tablas *Países* y *Ciudades*, aunque puede hacerse por separado, resulta más provechosa cuando se hace en conjunto. Se construirá entonces un único formulario en el que se definan los países (vinculado a la tabla *Países*), y dentro de éste otro formulario (vinculado a la tabla *Ciudades* y llamado sub-formulario) que permita introducir sus ciudades. Veremos que, de forma automática, las dos tablas involucradas resultan asociadas según como dicta el esquema de la BD. Por otra parte, la definición de países y ciudades es independiente de los libros, aun cuando al incluir los libros, se les necesitan.
- La tabla *Libros* recibirá su propio formulario. Además de sus características propias, este formulario deberá incluir para cada libro dos sub-formularios –independiente uno del otro pero dependientes ambos de los libros–, en los que sea posible incorporar los autores del libro y sus descriptores (a partir de las tablas *Autores de Libros* y *Descriptores de Libros* respectivamente).
- Considerando que se trata de una aplicación sencilla que no contiene información sensible o delicada, se requiere construir una única interfaz para todos sus potenciales usuarios, sin hacer distinciones.

En síntesis, nuestro ejemplo amerita la elaboración de los siguientes elementos:

1. Un formulario basado en la tabla *Autores*.
2. Un formulario basado en la tabla *Temas*.
3. Un formulario basado en la tabla *Países* con un sub-formulario vinculado, basado en la tabla *Ciudades*.
4. Un formulario basado en la tabla *Libros,* con un sub-formulario vinculado basado en la tabla *Autores de Libros* y un segundo subformulario, también vinculado, basado en la tabla *Descriptores de Libros*.

Por último, desearíamos que cuando el usuario edite los libros, pueda incorporar fácilmente a la tabla de autores, o temas, o ciudades de países, registros que vayan siendo necesarios para completar la in-

formación. Así pues, procuraremos que desde los libros puedan abrirse directamente los formularios que correspondan a estos datos de forma que se añada lo necesario y que, de vuelta a los libros, pueda utilizársele donde se requiera.

Ahora bien, planificada la estrategia, es menester poner manos a la obra y construir los formularios. Veamos cómo hacerlo inmediatamente.

Para diseñar y almacenar formularios, *Access* proporciona la pestaña homónima que se aprecia en la figura 19. Cuando allí se presiona el botón "Nuevo", se ejecuta un cuadro de diálogo como el que muestra la figura 50.

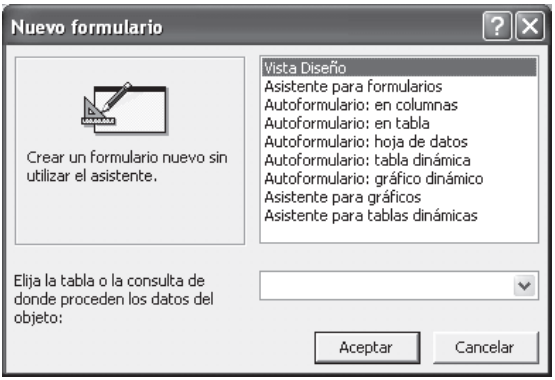


Figura 50. Cómo crear nuevos formularios

El cuadro de diálogo ilustrado en la figura 50 permite crear un formulario desde cero partiendo de una ventana completamente en blanco, o bien crearlo utilizando diversos asistentes que facilitan el trabajo, dependiendo de qué tipo de formulario se desea. El "Asistente para formularios" (segunda opción de la figura 50), automatiza la creación de cualquier tipo de formulario. La opción "Autoformulario: en columnas", crea un formulario basado en una tabla o consulta, cuyos campos serán dispuestos un registro a la vez por pantalla. La opción "Autoformulario: en tabla" crea un formulario basado en una tabla o consulta, cuyos campos serán dispuestos varios registros a la vez en un arreglo tabular en la pantalla. La opción "Autoformulario: hoja de datos" crea un formulario basado en una tabla o consulta, cuyos campos serán dispuestos emulando como se presentan en la visión convencio-

nal de tablas simples (como se ven los datos desde la pestaña "Tablas", en la ventana de base de datos). Las opciones destinadas a las tablas y gráficos dinámicos facilitan la construcción de síntesis de datos, presentadas en forma de referencias cruzadas, trasponiendo filas y columnas o desplegando gráficos de barras, tortas y demás a partir de los datos que se encuentran en la BD.

Supongamos que nos disponemos a crear el formulario decidido para la tabla *Autores*. Comenzaremos utilizando la opción "Autoformulario: en tabla", en vista de que *Autores* cuenta con sólo tres campos, que pueden verse apropiadamente en una ventana de forma tabular. Para ello, antes de presionar el botón "Aceptar" mostrado en la figura 50, debe seleccionarse el nombre de la tabla sobre la que basaremos nuestro formulario, en el campo dispuesto a tal fin. El asistente, entonces, automáticamente, crea un formulario y lo despliega como se aprecia en la figura 51.

En la figura 51 se muestra un típico y sencillo formulario del *Access* en formato tabular. Notemos que se trata de una ventana *Windows*. Como tal, contiene los elementos típicos de toda ventana, a saber: Título de la ventana en el borde superior, en este caso, "Autores". Pulsando y arrastrando sobre este título puede cambiarse de lugar la ventana. Contiene botones de posicionamiento y redimensionamiento que se encuentran como iconos en la esquina superior derecha de la ventana o como menús emergentes en la esquina superior izquierda. En la esquina superior derecha puede, en orden de presión de cada botón, minimizarse, maximizarse o cerrarse la ventana, que son operaciones frecuentes en *Windows*.

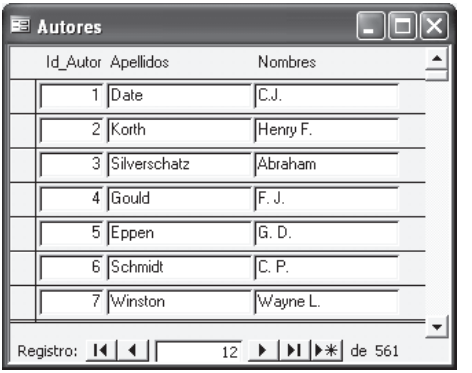


Figura 51. Formulario tabular "Autores"

En la figura 51 se aprecian también: Marco de la ventana bordeándola completamente. Este marco es susceptible de ser ampliado o disminuido utilizando el ratón y la operación de pulsar y arrastrar convencional. Barra de desplazamiento vertical dispuesta en el margen derecho de la ventana. Esta barra hace posible que el usuario desplace su visión del formulario hacia arriba o hacia abajo y su presencia indica la existencia de elementos en la ventana que no se están mostrando en el área visible. No hay barras horizontales, pero ello es simplemente porque en este caso no son necesarias. De haberlo sido, en razón de que el espacio visible de los objetos en sentido horizontal no abarcara todo su contenido, éstas habrían aparecido como lo harían en cualquier otra ventana del *Windows*.

El formulario ilustrado en la figura 51 presenta algunos elementos adicionales: Dado que seleccionamos un formato tabular para la tabla *Autores*, los nombres que hemos dado a sus tres campos aparecen como "Etiquetas", o títulos de columna, en la parte interna superior de la ventana. El contenido de la tabla, expresado en los valores de cada campo en cada registro, en este caso un identificador único para cada autor, sus apellidos y sus nombres, aparecen ahora de manera que se muestran varias filas y cada fila representa a un autor (o registro) diferente. Utilizando del teclado las teclas [RePág], [AvPág], [Inicio], [Fin], [Ctrl]-[Inicio], [Ctrl]-[Fin], las flechas de dirección, entre otras combinaciones, puede "navegarse" a través de todos los registros de la tabla.

Están disponibles todas las acciones que se acostumbran sobre campos y registros en una base de datos. Esto quiere decir que el usuario puede moverse entre registros, detenerse en uno de ellos, posicionarse en uno de sus campos–valores, alterar su contenido, eliminar el registro completo, abrir un nuevo registro e introducir datos inéditos. Esta apertura a todas las operaciones de edición es el comportamiento por defecto de los formularios vinculados en *Access*. Para deshabilitar algunas o todas estas facilidades deben alterarse las propiedades del formulario o bien, como dijimos antes, añadirse segmentos de código fuente al módulo del formulario que le den características personalizadas. Todo lo que se haga sobre los datos en un formulario vinculado quedará almacenado en la base de datos automáticamente, cuando el usuario abandone el registro. No es necesario entonces guardar explícitamente los cambios, no obstante, hay que ser cuidadosos, pues un cambio puede ser la eliminación del valor de un campo hecha por error,

y esta eliminación también quedará guardada, lo cual significa que el dato se eliminará efectivamente, sin advertencia.

Cada registro en un formulario se visualiza entre separadores. Los separadores son líneas horizontales en la ventana que dividen el espacio entre registros. Los separadores superior e inferior de un registro conforman a su izquierda un botón señalador de dicho registro. Cada vez que el usuario se detiene en un registro, este botón señalador se iconiza como una punta de flecha con dirección a la derecha, indicándolo como el registro activo. Si se desea copiar o eliminar un registro, puede darse clic sobre el señalador correspondiente y presionar, por ejemplo, las teclas [Ctrl]-[Ins] o [Suprimir], completando la operación luego de responder afirmativamente a la advertencia que aparecerá en pantalla. De forma similar pueden realizarse operaciones de "cortado" y "pegado" de registros.

El último elemento de un formulario estándar es la sección dedicada al desplazamiento entre registros. Esta sección se ubica en la parte interna inferior de la ventana, etiquetada con la palabra "Registro". En la tabla 3 se describen sus elementos.

Tabla 3. Elementos de un formulario para el desplazamiento entre registros

Elemento	Descripción
⏮	Botón para ir al inicio del conjunto de registros. Coloca el selector de registros en el primero del conjunto.
◀	Botón para ir al registro anterior. Coloca el selector de registros en el inmediato anterior al actual.
12	Cuadro de texto que muestra el Nº del registro actualmente seleccionado. El usuario puede escribir aquí un nuevo número, lo que lo llevará directamente a dicho registro. Estos números no se corresponden con los identificadores de registro, sólo son una secuencia automáticamente generada por el SMBD a medida que los registros entran a la tabla. No obstante, pueden utilizarse alternativamente para efectos de ubicación contextual en el formulario.
▶	Botón para ir al registro siguiente. Coloca el selector de registros en el inmediato posterior al actual.
⏭	Botón para ir al final del conjunto de registros. Coloca el selector de registros en el último del conjunto.
▶*	Botón para agregar un registro. Coloca el selector de registros una posición más allá del último del conjunto, lo cual habilita un nuevo registro para introducir datos inéditos.

de 561	Etiqueta indicadora del número total de registros que contiene el conjunto. Así se conoce, por ejemplo en este caso, que el usuario se encuentra situado en el registro Nº 12 de un total de 561 que contiene el conjunto.

Considerando que la figura 51 se refiere a un formulario de autores, en su interior los campos de la tabla homónima se presentan como "Cuadros de Texto", susceptibles de recibir datos o modificar aquellos que se tienen para el momento. Todas las teclas de movimiento habituales, como el tabulador, [Enter] (o retorno del carro), entre otras, funcionan para desplazarse entre campos. En particular, la tecla de función [F2] pone al sistema en disposición de editar el contenido de un campo de datos. Por ejemplo, si el apellido del autor "Eppen", que ocupa el quinto registro en la figura 51, contuviera un error de transcripción en el nombre, digamos, su nombre se escribiera con una sola letra "G." (y no "G. D.", como muestra la figura), un posible curso de acción para corregirlo sería: (a) Ubicarse en el registro apropiado avanzando o retrocediendo en el conjunto, o bien escribiendo el número 5 y [Enter] en el cuadro de texto del Nº de registro actual (ver tabla 3). (b) Presionar el tabulador dos veces hasta seleccionar el campo "Nombres" del registro. (c) Presionar la tecla de función [F2] para editar el contenido interno del campo. (d) Mover el cursor con las flechas hasta ubicarlo luego del primer punto. (e) Suprimir el espacio y las dos letras que sobran (" D."). (f) Presionar [Enter] o [Tab] para pasar al siguiente campo.

Por otra parte, debe quedar claro que lo mostrado en la figura 51 es la vista del formulario que tendrá el usuario. Se trata del formulario elaborado por el asistente de manera automática y presentado al usuario (no al diseñador) para que opere sobre sus datos. De hecho, el asistente ha guardado automáticamente el formulario, asignándole un nombre equivalente a la tabla vinculada, en este caso, *Autores,* y ahora aparece en la pestaña de formularios de la ventana de base de datos.

Sin embargo, el diseñador no está habitualmente conforme con la creación automática de un formulario y desea hacerle cambios. Un formulario abierto puede alternarse entre la "vista diseño" y "la vista formulario" (que es como *Access* llama a la vista del usuario) seleccionando la opción apropiada en el menú secundario asociado con él. En la vista diseño, el analista puede alterar todos los elementos de un

formulario. En *Access,* el diseño siempre se identifica con un icono que muestra una escuadra, un lápiz y una regla.

La figura 52 muestra la vista diseño del formulario de autores que venimos estudiando, con los elementos que ya incorporó el asistente que le dio forma.

Los elementos resaltados en el figura 52 son comunes a todos los formularios. La figura muestra en particular, para el caso de la tabla *Autores*, etiquetas y cuadros de texto vinculados a sus campos, que son también muy frecuentes. Concentrándonos en esta oportunidad sobre los elementos comunes a todos los formularios, veamos a continuación la descripción de cada uno de los señalados en la figura 52.

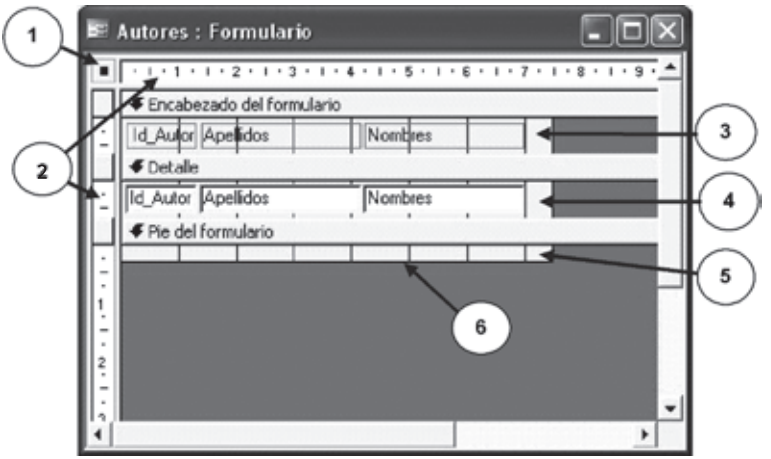


Figura 52. Vista "Diseño" del formulario de autores

1. Botón señalador de todo el formulario. Cada vez que el diseñador desee operar con las propiedades de todo el formulario como un objeto, podrá dar doble clic en este icono (o clic izquierdo del ratón para acceder al menú secundario del formulario).
2. Reglas de ubicación. Las reglas horizontal y vertical mostradas en la figura 52 permiten al diseñador ubicar los elementos internos del formulario relativos a una unidad de medida equivalente a como aparecerán luego en la vista formulario. Si se da clic sobre cualquiera de estas reglas en un punto al que correspondan objetos dentro del formulario, estos objetos se seleccionarán. Esta op-

ción es útil cuando se desea alterar la posición de varios objetos a la vez, preservando la ubicación entre ellos.

3. Sección "Encabezado del formulario". Esta sección, limitada entre dos barras horizontales, representa el área del formulario que por definición será mostrada en todo momento, encabezándolo. Como las otras áreas, puede contener cualquier objeto de datos, etiquetas y gráficos, entre otros. En el caso de la figura 52 se disponen aquí las etiquetas de los campos, pues en vista de que se ha diseñado el formulario para mostrar una tabla de valores, aunque el usuario no pueda ver simultáneamente todos los registros desea ver en todo momento los encabezados de los campos.
4. Sección "Detalle". Esta sección contiene por lo general los elementos de datos vinculados a los registros de la tabla origen. Generalmente es la preocupación central al momento de diseñar un formulario. *Access* controla automáticamente el paso de un registro a otro, según cómo hayan sido dispuestos los campos en esta sección. Al compararse las figuras 51 y 52 se comprende cómo funciona la sección "Detalle". El diseñador dispone cuadros de texto, uno por campo, tal como desea que aparezcan en la vista del usuario, y al haberse definido el formulario como "tabular", *Access* interpreta que en dicha posición relativa, fila tras fila, deberá mostrar todos los registros mientras quepan en la ventana. Dicho en otros términos, en la figura 51 vemos varios registros de autores; sin embargo, en la figura 52 vemos sólo un ejemplo de cómo queremos que aparezcan estos registros; no es necesario y de hecho sería un error disponer en la vista diseño varios registros iguales, a menos que quisiéramos repetir la información a propósito.
5. Sección "Pie del formulario". Como la sección "Encabezado del formulario" representa un área que por definición será mostrada en todo momento, ahora señalando su término. Puede contener cualquier objeto de datos, etiquetas, gráficos y demás. En el caso de la figura 52 no se ha utilizado en lo absoluto.
6. Fondo del formulario. Es la cuadrícula gris claro que aparece detrás de todos los elementos del formulario. Resulta bien importante, pues define el área de la ventana que deberá abarcarse al presentar el formulario. Puede ampliarse o disminuirse a voluntad pulsando y arrastrando sus bordes o esquinas con el ratón.

Cuando un formulario se abre para su diseño, se habilita una barra de herramientas como la que muestra la figura 53. Ella contiene iconos que representan objetos que pueden incluirse en los formularios. La tabla 4 explica cada uno de estos objetos.

Figura 53. Barra de herramientas para el diseño de formularios

TABLA 4. ELEMENTOS EN LA BARRA DE HERRAMIENTAS DEL FORMULARIO

Elemento	Descripción
	Botón de selección. Su función es disponer el cursor del ratón en posibilidad de seleccionar uno o varios objetos del formulario. Los objetos pueden seleccionarse dando clic sobre ellos o formando un rectángulo que los contenga pulsando y arrastrando el ratón. También puede utilizarse la combinación Shift–clic, un objeto a la vez, para seleccionarlos en secuencia. Los objetos seleccionados pueden luego moverse, eliminarse, alinearse vertical u horizontalmente, entre otras operaciones.
	Habilitación de los asistentes. Este es un botón interruptor que puede "prenderse" o "apagarse" según se desee o no mantener habilitados los asistentes. Los asistentes de que se dispone para el diseño de formularios están asociados a algunos objetos de forma tal que cuando el usuario posiciona dichos objetos en el formulario, una secuencia de cuadros de diálogo le ayuda a configurar sus características.
	Etiqueta. Se trata de un objeto que puede contener texto por lo general de naturaleza estática, esto es, invariable con el tiempo. Una etiqueta es una frase que explica o nombra alguna sección u objeto del formulario. En la figura 52, los objetos del encabezado del formulario que contienen los valores "Id_Autor", "Apellidos" y "Nombres", son etiquetas.
	Cuadro de texto. Es tal vez el objeto más utilizado en los formularios. Se trata de un contenedor de datos de cualquier tipo, en el que el usuario podrá incluir información escribiéndola con el teclado. Un cuadro de texto puede o no estar asociado con un campo de tabla; si lo está, entonces el dato que introduzca el usuario será considerado el dato que desea introducir en el respectivo campo vinculado al que hace referencia. Para todos los efectos prácticos opera como una variable a la cual se hace referencia por su nombre. Por ejemplo, los objetos "Id_Autor", "Apellidos" y "Nombres" que se encuentran en la sección "Detalle" de la figura 52, son cuadros de texto vinculados a los campos homónimos de la tabla *Autores*.

Elemento	Descripción
	Grupo de opciones. Se trata de un objeto que enmarca un conjunto preestablecido de opciones de entre las cuales el usuario puede seleccionar una. Cada opción disponible tiene asignado un valor único que lo diferencia de las demás, de manera que aquella selección que realiza el usuario se transforma en el valor del objeto "Grupo de Opciones".
	Botón de alternar. Es un objeto que puede tomar los valores "verdadero" cuando está presionado y "falso" cuando no lo está. Como en otros objetos de su tipo, su nombre representa una variable del formulario, que toma estos dos valores mencionados según se encuentre o no presionado.
	Botón de opción. Es un objeto previsto para recibir la selección del usuario. Comúnmente se utiliza dentro de un "grupo de opciones" o como sustituto del "botón de alternar". Puede o no contener un valor dependiendo de si el usuario lo ha seleccionado o no, pero su valor no está restringido a ser sólo verdadero o falso.
	Casilla de verificación. Un objeto muy similar a los dos anteriores en el sentido que se dispone para recibir la selección del usuario. La casilla de verificación asume el valor "verdadero" si está seleccionada y "falso" en caso contrario.
	Cuadro combinado. Objeto construido a partir de un cuadro de texto, que incorpora una facilidad muy útil adicional: la presentación de los posibles valores que puede contener. Cuando se muestra un cuadro combinado se espera que el usuario le asigne un valor cualquiera, de la misma forma como lo haría con un cuadro de texto, no obstante, si el usuario hace clic sobre la flecha que apunta hacia abajo (en su borde derecho), se despliega una lista de valores de entre los cuales se puede o debe seleccionar uno. De esta forma el usuario no tiene que conocer de antemano exactamente qué valor dar al control; revisando la lista desplegada puede seleccionar el que le parezca más apropiado. Por supuesto, dicha lista debe ser construida por el diseñador; puede formarse estáticamente una vez o dinámicamente a partir de datos en una tabla o consulta y, entre sus características, está el permitir o no que el usuario incorpore un nuevo elemento que no esté en ella.
	Cuadro de lista. Cumple la misma función que el cuadro combinado mencionado antes pero, en este caso, la lista de valores que se propone al usuario no se despliega a su voluntad, sino que el objeto muestra por defecto todos los valores de la lista. El usuario puede recorrer todos los valores valiéndose de barras de desplazamiento que están incluidas en el propio control, y seleccionar uno de ellos, que se convierte en el valor del objeto. Con algo de esfuerzo de programación puede hacerse que el control acepte más de un valor, pero este no es su comportamiento por defecto.
	Botón de comando. Este control no está diseñado para aceptar valores, sino pulsos del ratón. Puede programarse para responder a eventos y normalmente se le emplea para dar al usuario la posibilidad de desencadenar acciones a su voluntad. Por ejemplo, puede programarse para abrir un formulario, imprimir un informe o presentar el resultado de una consulta, entre otras acciones, cuando se le pulse.

Elemento	Descripción
	Imagen. Objeto que enmarca y encapsula una imagen de cualquier tipo. Dentro de un formulario pueden incorporarse imágenes que por lo general otorgan información estática al usuario. Por ejemplo, exhibir la imagen de un libro encabezando el formulario dedicado a ellos sería buena idea, pues cada vez que el usuario la viera, rápidamente entendería de qué se trata e incluso, con el tiempo, sin leer los títulos que se encuentran en el formulario de los libros.
	Marco de objeto independiente. Permite incorporar en un formulario un objeto de cualquier naturaleza creado con una aplicación diferente del Access, vía OLE (por "*Object Linking and Embedding*" u "Objeto Enlazado y Embebido"), que resulta incrustado en el formulario, pero que no depende de los datos contenidos en una tabla o consulta de la base de datos. Por ejemplo, un formulario puede incorporar un documento del Word o un gráfico del AutoCad de forma independiente de la base de datos que, cuando el usuario lo seleccione, invoque a la aplicación que le dio origen para manipularlo. Se trata de una operación delicada, pues requiere la presencia en el computador de un servicio que atienda las peticiones OLE.
	Marco de objeto dependiente. Como en el "marco de objeto independiente", permite incorporar objetos creados y mantenidos por otros programas diferentes del Access, sin embargo, en este caso, los objetos en cuestión sí dependen de datos contenidos en la base de datos y por tanto, entre sus propiedades se encuentra una que permite establecer el origen de datos del que dependen.
	Salto de página. Divide en páginas la presentación de un formulario afectando su desplazamiento vertical.
	Control ficha. Es un contenedor de objetos que los separa por pestañas. Cuando deben incorporarse muchos controles a un formulario, es buena idea separarlos de acuerdo con su tipo y crear una pestaña de la ficha para cada tipo. Esta facilidad permite que los formularios contengan múltiples objetos organizadamente.
	Subformulario/Subinforme. Es un control que se dispone en un formulario o informe para que otro formulario o informe resulte embebido en él. Al formulario o informe embebido en el original se le denomina sub-formulario o sub-informe. Los subformularios y subinformes pueden estar o no vinculados, en términos de datos, con los formularios e informes que los contienen. Si están vinculados, campos del formulario se hacen coincidir con campos del subformulario para obrar el efecto que cuando un registro del formulario cambie, también lo hagan los datos asociados a él en el subformulario. Los subformularios o subinformes sin vinculación o independientes se utilizan para presentar información general, como sería, por ejemplo, un encabezado o cualquier tipo de información estática.
	Línea. Objeto gráfico que permite dibujar una línea en el formulario y darle cualquier orientación.
	Rectángulo. Objeto gráfico que permite dibujar un rectángulo de cualquier tamaño en el formulario.

Elemento	Descripción
	Más controles. Este botón otorga al diseñador acceso a otros objetos (no estándar del *Access*) creados con programas en otros lenguajes. Por defecto, no existen más controles que los anteriores y esta herramienta no produce acción alguna.

Un formulario, las secciones que lo integran y todos sus objetos tienen propiedades que pueden alterarse a voluntad del diseñador. Para acceder a las propiedades de un objeto cualquiera, basta dar doble clic sobre él en la vista diseño de formularios o bien pulsar sobre él con el botón derecho del ratón para luego seleccionar la opción "Propiedades". En total son un número grande de propiedades (un formulario en la versión 2003 de *Access*, tiene 107 propiedades diferentes). Afortunadamente, el SMBD les otorga valores por defecto cuando crea los objetos. El diseñador suele aceptar la mayoría de estos valores y concentrarse en alterar sólo aquellos que intervienen cuando desea dar algún comportamiento especial a su ventana.

El nombre de un objeto, el elemento de datos al que se encuentra vinculado, si al presionarse debe ocurrir algo programado con antelación, su forma, su tamaño, su color, las operaciones de edición que están permitidas, si se encuentra habilitado o no, si debe o no aparecer en la vista del usuario, en fin, la mayoría de las características de un objeto se establecen en sus propiedades. El Apéndice B (en el CD) lista todos los objetos de un formulario con sus propiedades.

Regresando a nuestro ejemplo de los libros, demos por creado el formulario que corresponde a la tabla *Temas*, pues podría hacerse de manera muy similar a como explicamos para el caso de los *Autores* y concentrémonos ahora sobre el formulario que haremos corresponder a la tabla *Países*.

La figura 54 muestra el formulario para los países que deseamos lograr. Se trata de un formulario basado en la tabla *Países* de manera muy similar a los anteriores, que ahora contiene además un subformulario vinculado basado en la tabla *Ciudades*. Cada registro de los países en la primera tabla resulta asociado con los registros de sus ciudades en la segunda. Por ejemplo, en la figura, el país "España" tiene definidas las ciudades "Barcelona", "Bilbao" y "Madrid", de manera que son las únicas que se ven. Si el usuario inserta una nueva ciudad, el formulario entendería que se trata de una nueva ciudad del país "España", contro-

lando automáticamente la inclusión del código de país correspondiente en la tabla *Ciudades*. Si tratara de incorporar un nuevo país utilizando el formulario, los registros de las ciudades aparecerían en blanco, prestos a aceptar sus ciudades.

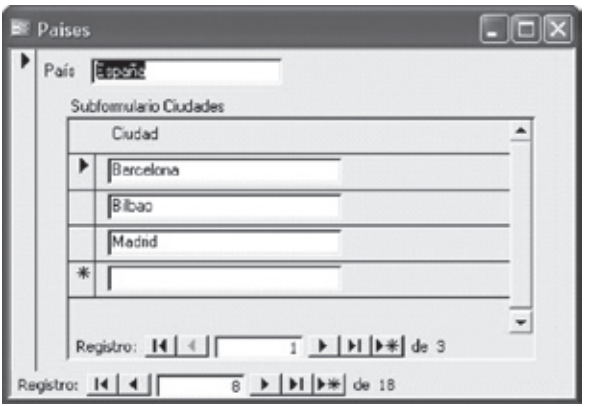


Figura 54. Formulario "Países" y "Ciudades"

El control de los movimientos entre registros de la figura 54 se lleva a cabo tanto en el formulario como en el subformulario, de la misma manera. En el ejemplo, el usuario se encuentra visualizando el registro Nº 8 de 18 totales en la tabla *Países*, y el registro Nº 1 de 3 totales en la tabla *Ciudades*, correspondientes a "España".

El formulario principal es del tipo "En columnas", esto es, muestra un único registro de la tabla *Países* a la vez, mientras que el subformulario es del tipo "Formularios continuos", pues muestra a la vez varios registros de la tabla *Ciudades*.

La figura 55 contiene la vista diseño del formulario *Países*. Puede constatarse que incluye dos controles, a saber: un cuadro de texto cuyo origen del control es el campo "País" de la tabla *Países*, y un subformulario cuyo objeto origen se denomina "Subformulario Ciudades". El subformulario y el formulario se vinculan con la regla siguiente: el campo principal es "País" de la tabla *Países*, y el secundario es "País" de la tabla *Ciudades;* luego, para todos los efectos, Países.País = Ciudades.País. A su vez, dentro del subformulario existe un único control, cuadro de texto, cuyo origen del control es el campo "Ciudad" de la tabla *Ciudades*.

Figura 55. Vista diseño del formulario "Países"

La construcción del formulario principal de la figura 55 utilizó el "Asistente para formularios" descrito a partir de la figura 50. Para la incorporación del subformulario se abrió el formulario en la vista diseño y se incorporó un objeto subformulario/subinforme a partir de la barra de herramientas, tal como se describe en la tabla 4.

Si el botón de activación de los asistentes se encuentra presionado cuando se incorpore un control subformulario/subinforme como el de la figura 55, se despliega un asistente que permite configurar automáticamente sus características, como muestra la secuencia en la figura 56.

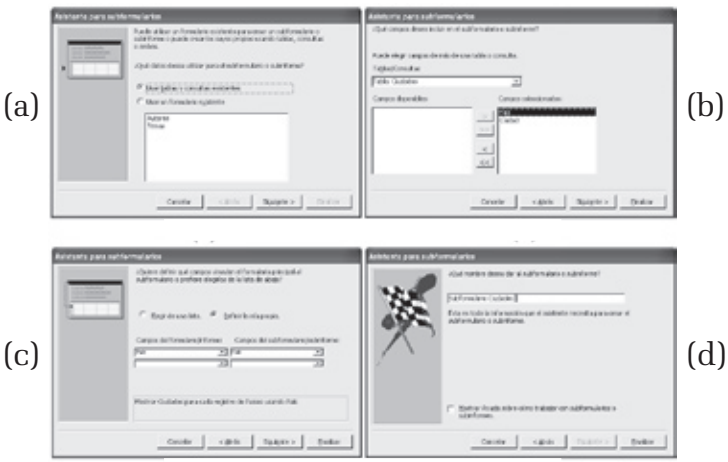


Figura 56. Secuencia de configuración de subformularios utilizando el asistente

La figura 56 despliega la secuencia en que el asistente para la creación de subformularios, solicita información al diseñador. La parte (a) de la figura permite decidir si el subformulario deberá ser construido partiendo de cero, con base en una tabla o consulta, o si, por el contrario, se incorporará como subformulario un formulario que ya exista en la ventana de base de datos. En este caso seleccionamos la primera de las opciones. La parte (b), entonces, muestra la ventana en la que debe escogerse la tabla o consulta sobre la que se desea basar el subformulario y los campos que contendrá. En este caso, la tabla en cuestión es *Ciudades* y los campos son dos: "País" y "Ciudad". La parte (c) de la figura 56 permite decidir de qué manera resultarán vinculados los campos del formulario y subformulario. La vinculación se establece como dictan las reglas de integridad referencial y como se desprende del diseño de claves primarias y ajenas, en este caso, Países.País = Ciudades. País. En el pie puede comprobarse cómo el sistema está "entendiendo" la relación cuando se lee "Mostrar Ciudades para cada registro de Países usando País". Por último, en la parte (d) se coloca el nombre que recibirá el subformulario.

El procedimiento ilustrado en la figura 56 produce un subformulario vinculado, dentro del formulario *Países*, que contiene dos controles asociados con los campos de la tabla *Ciudades*. No obstante, lo que muestra la figura 55 es un subformulario con un solo control (y campo), "Ciudad". Manualmente hemos eliminado el otro control, puesto que es innecesario para efectos de presentación. Efectivamente, no vale la pena presentar dos veces el país (tanto en el formulario como en el suformulario), ya que el usuario sabrá de qué país se trata, pues lo verá en el formulario principal.

Otro aspecto que hemos alterado manualmente es la forma de presentación del subformulario. Por defecto, los formularios se presentan como "hojas de datos", sin embargo, como puede apreciarse en la figura 54, hemos cambiado esta forma por la de "Formularios continuos". Para lograrlo modificamos consecuentemente la propiedad del subformulario denominada "Vista predeterminada" (ver en CD, Apéndice B, Sección B.1). También movilizamos la etiqueta del control, de la sección "Detalle" a la sección "Encabezado" del formulario.

La figura 57 muestra el menú de herramientas para el diseño de formularios.

Figura 57. Menú de herramientas para el diseño de formularios

Resaltan dos de sus botones:

1. Lista de campos. Permite mostrar los campos de la tabla o consulta a la que se encuentra vinculado el formulario. Por ejemplo, los campos de la tabla *Libros* que pueden incorporarse a un formulario (mostrados en la figura 58) se presentan después de presionar este botón, suponiendo que el formulario que se edita en el momento está vinculado a la tabla *Libros*. De esta lista pueden "pulsarse y arrastrarse" hasta el área del formulario, los campos que desean incluirse.

Figura 58. Campos de tabla desplegados desde el menú de herramientas

2. Código. Este botón invoca a una aplicación que comparten todos los programas del paquete *Office*. Se trata del editor de programas o código fuente del "Visual Basic para Aplicaciones". El editor de programas permite escribir instrucciones codificadas en el lenguaje de programación *Visual Basic*, particularmente, la edición "para Aplicaciones", que en este caso se refiere al *Access*. Tratándose de programas asociados con un formulario, el editor se dispone a incorporar código fuente cuyo ámbito de acción es el propio formulario. Más específicamente, el módulo donde se guardarán los programas que se escriban en esta oportunidad será el que corresponde al formulario que se está diseñando. Práctica-

mente todos los objetos de un formulario pueden ser programados a voluntad, pero como adelantábamos antes, comentaremos sólo una pequeña porción de programas necesarios para hacer funcionar un botón de comando.

Tal vez el formulario que introduce mayores complicaciones sea el que nos queda, *Libros*. Las figuras de la 59 a la 62 contienen elementos de dicho formulario, basado en un control ficha.

La figura 59 muestra la primera pestaña del control ficha dispuesto en *Libros*. Esta pestaña se dedica a la información general de cada libro y contiene cuadros de texto vinculados a los campos "Id_Libro", Título", "Año", "Precio" y "Fecha_Adquisición", así como cuadros combinados para los campos "Id_Tema", "País" y "Ciudad".

La figura 59 introduce nuevos elementos: dos botones de comando mostrados como iconos y un cuadro de texto desactivado para la edición. Dando clic sobre el botón de comando que esté a la derecha del control "Id_Tema", el usuario abrirá el formulario destinado a editar la tabla *Temas*. Lo propio ocurrirá si pulsa el botón a la derecha de los controles "País" y "Ciudad", en este caso, abriendo el formulario *Países*. La idea es que al encontrarse el usuario editando información de los libros, no deba abandonar su tarea si se le presenta el caso que necesita un tema, país o ciudad que no encuentre registrado aún en la base de datos. Le bastará entonces presionar el botón apropiado y al abrirse el formulario correspondiente podrá agregar (o modificar) el nuevo valor, sin abandonar la edición de los libros.

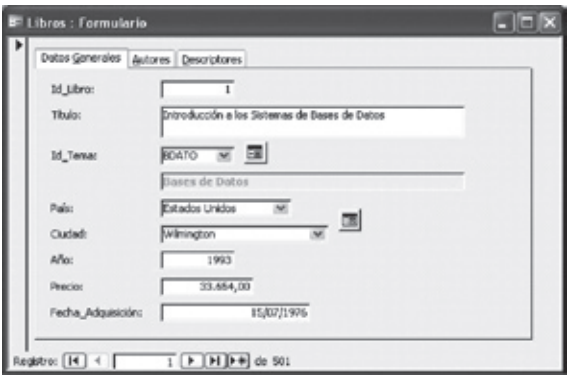


Figura 59. Formulario "Libros", tipo ficha. Pestaña "Datos Generales"

Para garantizar que los nuevos datos o las modificaciones a los datos existentes que se editen abriendo estos formularios sean almacenados antes de que se le necesiten en *Libros*, la lógica de la situación debe satisfacer dos condiciones importantes, a saber: (a) Cuando se abra el formulario auxiliar de temas o países no debe permitirse continuar ninguna otra tarea hasta tanto no se culmine con el formulario abierto y sea cerrado, y (b) Cuando el usuario salga del formulario auxiliar, habiendo o no editado la información deseada, la lista de valores del campo en *Libros* que dio objeto a su apertura debe resultar actualizada con la nueva información.

Ambos botones se diseñaron utilizando el asistente para configurar botones de comando que proporciona *Access*. En dicho asistente se pueden especificar operaciones de muchos tipos, que resulten desencadenadas al dar clic sobre el botón. En este caso, ambos botones implementan la operación de abrir un nuevo formulario (*Temas*, el primer botón y *Países*, el segundo). En la práctica, el asistente incorpora al módulo del formulario una subrutina programada que se ejecutará cuando ocurra el evento "Al hacer clic" asociado al botón (Ver en CD, Apéndice B, sección B.13). La programación del comportamiento de éste y todos los demás objetos programables de formularios e informes se realiza "orientada a objetos y conducida por eventos". Esto quiere decir que los objetos son programables en respuesta a eventos que ocurran con ellos, en tiempo de ejecución.

El botón que obra la apertura del formulario *Temas* fue llamado como "EditaTemas". El evento "Al hacer clic" de este botón queda programado por el asistente en la subrutina VBA que muestra la tabla 5, con una ligera modificación al código resaltada en negritas.

Comentaremos sobre el código fuente mostrado en la tabla 5, lo siguiente:

- El botón se llama "EditaTemas" y la subrutina "EditaTemas_Click()". VBA entiende que el programa de la subrutina se ejecutará si ocurre el evento "Click" asociado con el objeto "EditaTemas". Los nombres de los eventos son palabras reservadas del VBA y se escriben a continuación de un guión de subrayado, luego del nombre del objeto sobre el que debe responderse al evento.

TABLA 5.LISTADO DEL PROGRAMA "AL HACER CLIC" DEL BOTÓN "EDITATEMAS"

```
Private Sub EditaTemas_Click()
On Error GoTo Err_EditaTemas_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = "Temas"
    DoCmd.OpenForm stDocName, , , stLinkCriteria, , acDialog

Exit_EditaTemas_Click:
    Exit Sub

Err_EditaTemas_Click:
    MsgBox Err.Description
    Resume Exit_EditaTemas_Click

End Sub
```

- La subrutina es de carácter privado, por lo cual sólo será conocida dentro del módulo del formulario *Libros*.
- El control de errores de la rutina se opera con los comandos "On Error" y las etiquetas "Exit_EditaTemas_Click:" y "Err_EditaTemas_Click:", pero no tiene participación activa en el propósito de la rutina, cual es abrir el formulario *Temas*.
- La instrucción que produce la acción buscada, es:
  "DoCmd.OpenForm stDocName, , , stLinkCriteria, , acDialog"

Se trata del método "OpenForm" (abrir formulario) del objeto "DoCmd" (ejecute comando), pasándole como parámetros dos variables de tipo cadena de caracteres, "stDocName", que contiene el nombre del formulario que debe abrirse, y "stLinkCriteria", que en este caso contiene una cadena vacía, pero que podría contener un criterio de filtrado (cláusula *WHERE* del SQL) para los registros del formulario que será abierto. Otro parámetro que se pasa al método (añadido nuestro, no del asistente), es la constante "acDialog". Ella indica que el formulario a abrir deberá comportarse como un cuadro de diálogo, esto es, una vez abierto no podrá permitirse ninguna otra operación hasta tanto se le cierre. Las comas intermedias que separan los parámetros deben ponerse aun cuando aquéllos sean omitidos.

Para establecer la segunda parte de la lógica descrita, al botón "EditaTemas" debe incorporársele un fragmento de código VBA, esta

vez íntegramente escrito por el diseñador, de manera que al salir del botón sea actualizada la lista del control correspondiente en el formulario *Libros*. La tabla 6 muestra esta subrutina.

Tabla 6. Listado del programa "Al salir" del botón "EditaTemas"

```
Private Sub EditaTemas_Exit(Cancel As Integer)

    Id_Tema.Requery

End Sub
```

La tabla 6 presenta una subrutina VBA que no contiene código para la detección de errores (por supuesto, podría contenerlo), y cuya función es simplemente recalcular la consulta SQL sobre la que se basa la lista del control "Id_Tema". El método "Requery" del cuadro combinado "Id_Tema" asegura que cualquier cambio que haya producido el usuario editando el formulario *Temas* se vea reflejado ahora en la lista desplegable del cuadro combinado "Id_Tema" del formulario *Libros*.

De forma análoga ocurre para el botón destinado a la apertura del formulario *Países,* por supuesto, cambiando en este caso el valor de la variable "stDocName". Una diferencia más resaltante es ahora que al retornar de la edición del formulario *Países* deben actualizarse dos campos en lugar de uno sólo: el campo "País" y el campo "Ciudad". Ello es necesario puesto que el usuario ha podido alterar tanto la tabla *Países* como la tabla *Ciudades*. El código fuente para la apertura del formulario *Países* se omite por su similitud con el caso anterior, y la tabla 7 muestra el programa que responde al evento "Al salir", ahora para el botón "EditaPaíses".

Tabla 7. Listado del programa "Al salir" del botón "EditaPaíses"

```
Private Sub EditaPaíses_Exit(Cancel As Integer)

    País.Requery
    Ciudad.Requery

End Sub
```

Otro control novedoso del formulario mostrado en la figura 59 es el cuadro de texto que se encuentra desactivado para la edición (ubicado a continuación del control "Id_Tema"), y que parece no corresponderse con ningún campo de la tabla *Libros*. Este objeto se ha llamado "MuestraTema" y su papel es encontrar y mostrar el tema que se corresponda con el identificador del tema que seleccione el usuario en el formulario. En la figura 59, el "Id_Tema" seleccionado es "BDATO", y el tema al que corresponde este identificador es "Bases de Datos". Este cuadro de texto ha sido desactivado para la edición, estableciendo la propiedad "Activado" a "No" (Ver en CD, Apéndice B, sección B.6). La razón para desactivarlo es que no se espera que el usuario pueda alterar la información contenida en él, sólo se espera que la visualice. Adicionalmente se ha cambiado la propiedad "Espesor de la fuente" para dar más grosor a las letras y hacerlas más legibles. El "Origen del control" para este cuadro de texto se muestra en la tabla 8.

TABLA 8. INSTRUCCIÓN PARA EL "ORIGEN DEL CONTROL" DE "MUESTRATEMA"

```
=SiInm(Negado EsNulo([Id_Tema]);
      DBúsq("Tema";"Temas";"Id_Tema='" & [Id_Tema] & "'");"")
```

Nótese que "MuestraTema" es un cuadro de texto independiente, es decir, no vinculado a campo alguno. Su origen viene dado por la resolución de una función o expresión VBA que hace lo siguiente: Verifica que el campo "Id_Tema" no sea nulo. En caso de serlo, simplemente asigna como valor del cuadro de texto una cadena de caracteres vacía. Si resulta no serlo, entonces invoca la ejecución de la función "DBúsq" (en inglés, "*dlookup*"). La mencionada función "DBúsq" devuelve la primera ocurrencia del campo indicado como primer parámetro (en este caso, "Tema"), en la tabla o consulta indicada como segundo parámetro (en este caso, la tabla "Temas"), que cumpla con la condición lógica indicada como tercer parámetro. La condición lógica es una cadena de caracteres que en la figura 59 se evalúa como "Id_Tema='BDATO'". Adviértase cómo se indica una constante cadena (entre apóstrofes) y cómo se concatenan los valores constantes con las variables provenientes del formulario (utilizando el operador "&").

Una última característica que comentaremos para esta pestaña de la ficha en el formulario *Libros* tiene que ver con el comportamiento deseado para el par de campos "País" y "Ciudad". Cuando creamos la tabla *Ciudades* no pudimos establecer que las ciudades se mostraran para cada país particular. No se admiten estas restricciones en el nivel de la base de datos, sin embargo, ahora en la interfaz, deseamos que el usuario seleccione un registro del campo "País" y que las ciudades de ese país sean las únicas que se puedan escoger en el campo "Ciudad". Para lograr esto debemos alterar algunas propiedades de ambos controles en el formulario *Libros*. La tabla 9 muestra una de las alteraciones al evento correspondiente en el control "País".

Tabla 9. Código VBA incorporado al campo "País" en Libros

```
Private Sub País_Change()

    Ciudad = Null
    Ciudad.Requery

End Sub
```

La tabla 9 muestra una subrutina que se ejecuta cuando el usuario cambia el valor del campo "País", es decir, cuando selecciona otro país distinto al que allí estaba. Al cambiar el valor, el campo "Ciudad" se anula y su lista desplegable –recuérdese que se trata de un cuadro combinado– es actualizada, ahora con la información del nuevo país.

El "Origen de la fila" del campo "País" queda inalterado, sin embargo, el "Origen de la fila" del campo "Ciudad" debe ser alterado para que incluya la restricción de mostrar sólo las ciudades del país seleccionado. La figura 60 muestra el asistente para consultas que genera este nuevo "Origen de la fila".

Nótese que la lista desplegable del control "Ciudad" contendrá una sola columna con las ciudades, filtradas de acuerdo con su pertenencia al país seleccionado para el libro.

La forma en que se introduce un parámetro que depende de un objeto dentro de un formulario tiene la sintaxis [Forms]![<Nombre del Formulario>]! [<Nombre del Control>], en este caso, "[Forms]![Libros]![País]". La instrucción SQL resultante en la figura 60, es la siguiente:

PARAMETERS [Forms]![Libros]![País] Text(255);

```
SELECT Ciudades.Ciudad
FROM Ciudades
WHERE Ciudades.País=[Forms]![Libros]![País];
```

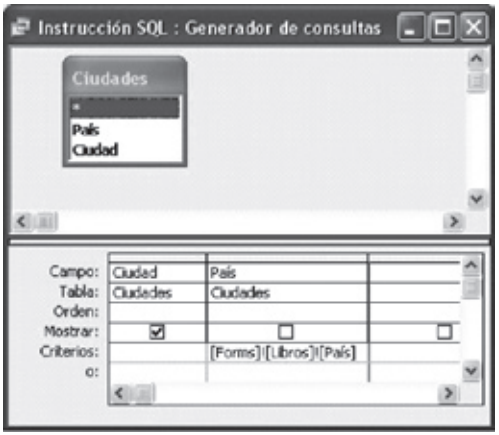


Figura 60. Consulta para filtrar las ciudades de acuerdo con el país

Para adaptar el formato a la nueva lista desplegable deben también alterarse algunas propiedades relacionadas del cuadro combinado "Ciudad" (ver en CD, Apéndice B, Sección B.11). El nuevo "Número de columnas" es 1, el "Ancho de columnas" es 4 cm., la "Columna dependiente" es 1, el "Ancho de la lista" es 4 cm. y, por último, "Limitar a la lista" es ahora "Si".

La figura 61 muestra la segunda pestaña del formulario *Libros*, bautizada como "Autores". Esta pestaña contiene un subformulario vinculado a la tabla *Autores de Libros* de acuerdo con el campo "Id_Libro". Para que el usuario pueda editar directamente los autores si no los encuentra definidos y disponibles en el cuadro combinado "Id_Autor", se incorpora, como antes, un botón cercano a la etiqueta homónima que abrirá para la edición el formulario *Autores*. Las operaciones sobre este formulario son similares a las explicadas con antelación para la pestaña "Información General", en los controles "Id_Tema", "País" y "Ciudad". Por otra parte, para la actualización al salir de los valores de la lista desplegable en el control "Id_Autor", ahora debe darse su nombre completo en el código VBA, pues se trata de un objeto dispuesto no sobre el formulario, sino dentro de un subformulario. Ello se muestra en la tabla 10.

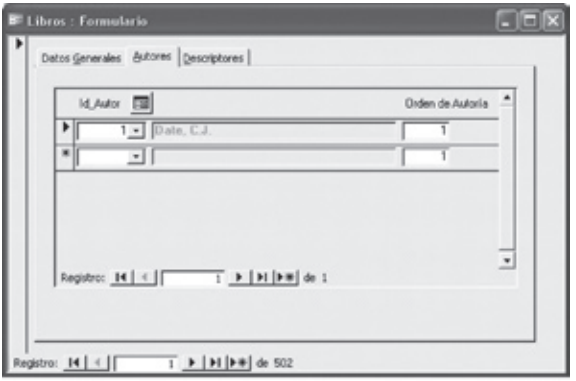


Figura 61. Formulario "Libros", tipo ficha. Pestaña "Autores"

TABLA 10. LLAMADA A UN CONTROL QUE ESTÁ EN UN SUBFORMULARIO

```
Private Sub EditaAutores_Exit(Cancel As Integer)

Forms![Libros![Subformulario Autores de Libros]!Id_Autor.
Requery

End Sub
```

La pestaña "Autores" contiene dentro del subformulario un cuadro de texto explicativo del autor de que se trata el "Id_Autor". El único autor del libro "Introducción a los Sistemas de Bases de Datos" del ejemplo, es Date, C.J., cuyo identificador es el Nº 1.

El cuadro de texto en cuestión, bautizado como "MuestraAutor", tiene el "Origen del control" que se presenta en la tabla 11.

TABLA 11. INSTRUCCIÓN PARA EL "ORIGEN DEL CONTROL" DE "MUESTRAAUTOR"

```
= SiInm(Negado esNulo([Id_Autor]);
DBúsq("Apellidos";"Autores";"Id_Autor=" & Cad([Id_Autor])) &
", " & DBúsq("Nombres";"Autores";"Id_Autor=" &
Cad([Id_Autor]));"")
```

Ahora, si no es nulo el identificador del autor, se ejecuta la función "DBúsq" para encontrar los apellidos del autor cuyo identifica-

dor es el que está en el formulario para el momento, y se le concatena, agregando una coma y un espacio (“, ”), con una nueva ejecución de la función “DBúsq”, que encuentra sus nombres. Nótese que la función “DBúsq” sólo puede devolver el valor de un campo a la vez; nótese asimismo que como ahora el identificador de un autor es un número y no una cadena de caracteres como antes, dicho número no recibe entreapostrofado y debe ser convertido a cadena con la función “Cad()” antes de concatenarse. La condición para el ejemplo de la tabla 11 se evalúa como “Id_Autor=1” en ambas llamadas a la función “DBúsq”, y su resultado, como puede verse en la figura 61, es “Apellidos, Nombres”, es decir, “Date, C.J.”.

Para finalizar con el formulario *Libros*, la figura 62 contiene la tercera pestaña de la ficha, bautizada “Descriptores”. Esta pestaña no contiene novedades a lo ya dicho y no haremos comentarios sobre ella.

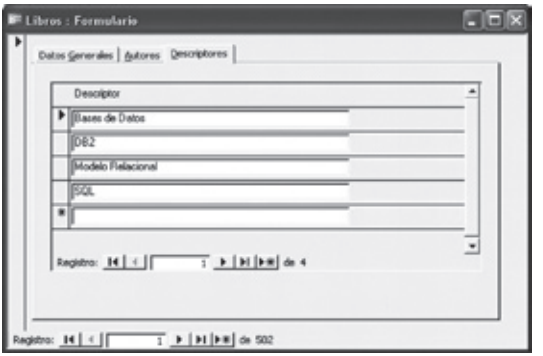


Figura 62. Formulario “Libros”, tipo ficha. Pestaña “Autores”

Los formularios son representantes de la interfaz de entrada de datos de todo sistema de información, sin embargo, para la presentación de datos al usuario, los objetos idóneos son los “Informes”, sobre los cuales versará la próxima sección.

3.7 Presentación de informes

Por “Informe” se entiende un objeto diseñado para imprimirse de forma atractiva y cuyo contenido serán datos dispuestos con caracte-

rísticas particulares, una organización preconcebida y un formato apropiado. Para agregar un informe puede presionarse el botón "Nuevo" en la pestaña "Informes" de la "ventana de base de datos".

Los controles que intervienen en los informes son en esencia los mismos que en los formularios: cuadros de texto, casillas de verificación, cuadros combinados o de listas, gráficos embebidos y otros. No abordaremos sus definiciones aquí, pues ya lo hicimos en la sección anterior. La diferencia entre formularios e informes estriba en que estos últimos se diseñan pensando no en la interacción bidireccional con el usuario (usuario → computador y computador → usuario), sino en la presentación de datos disponibles en un instante, o interacción unidireccional (computador → usuario).

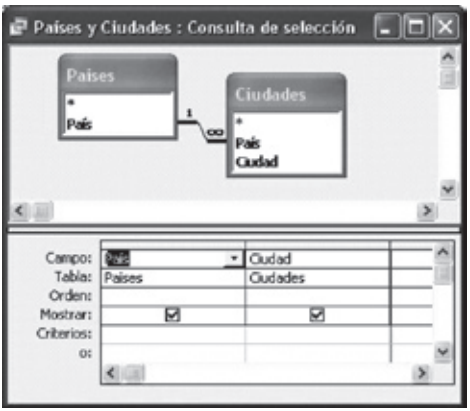


Figura 63. Consulta de Países y Ciudades

Supóngase como un primer ejemplo, que contamos con una consulta como la de la figura 63, titulada *Países y Ciudades*, de la cual se obtiene una lista de los países y sus ciudades.

La figura 64 despliega la secuencia de pasos para crear un informe utilizando el asistente, a partir de la consulta *Países y Ciudades*.

Se despliega una secuencia que comienza por interrogar sobre qué tabla o consulta será construido el informe en cuestión. En este caso, la parte (a) de la figura indica que se selecciona la consulta *Países y Ciudades* y sus dos campos "País" y "Ciudad".

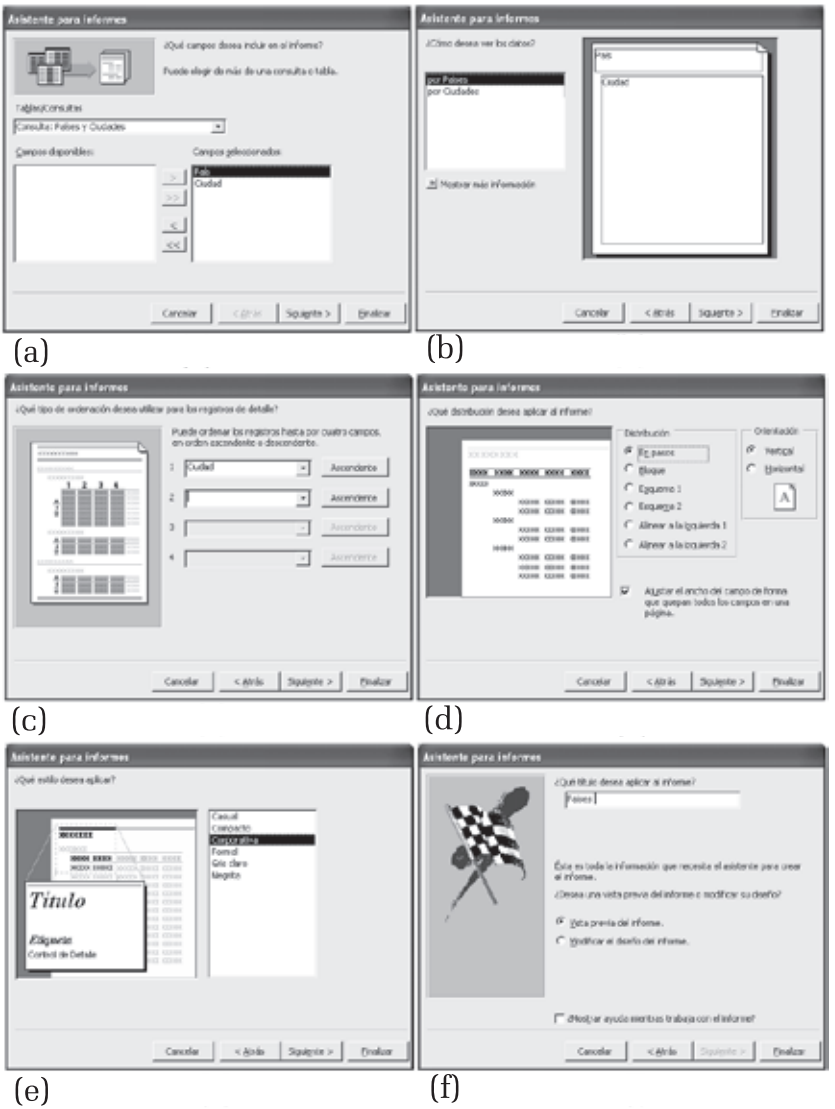


Figura 64. Creación del informe "Países y Ciudades" con el asistente para informes

La parte (b) de la figura sugiere que un informe puede recibir distintos niveles de agrupamiento en los datos. En este caso se ha decidido agruparlos por países de manera que para cada país se muestren sus ciudades. Esta es una de las características más resaltantes de los infor-

mes, la creación de grupos de datos. La parte (c) de la figura 64 permite decidir los campos de la sección interna del grupo, por los cuales se desea ordenar el informe. En este caso se ordenará alfabéticamente por ciudad, dentro de cada país.

La parte (d) se dedica a presentar al usuario una serie de formatos prediseñados para la organización de informes, de entre los cuales debe seleccionar uno. En este caso se selecciona la distribución "En pasos", con orientación vertical de la hoja, que produce un solo encabezado para grupos y datos indentando estos últimos respecto a cada grupo. La parte (e) da distintas opciones de forma o características tipográficas que pueden aplicarse al informe. En este caso se seleccionó el conjunto "Corporativo", que presenta el título del informe en letras grandes como encabezado, dibuja líneas de distinto grosor como separadores, coloca las etiquetas de campo en un encabezado de lista y los datos con una fuente pequeña. Finalmente, la parte (f) de la figura permite al diseñador dar un nombre al informe que ha creado.

Aunque siempre será posible comenzar el diseño de un informe sin utilizar los asistentes, el asistente para informes resulta muy útil para comenzar. Adicionalmente, como estrategia de valor, siempre será buena idea construir una consulta que contenga el mínimo conjunto de datos a presentar en un informe, antes de crearlo.

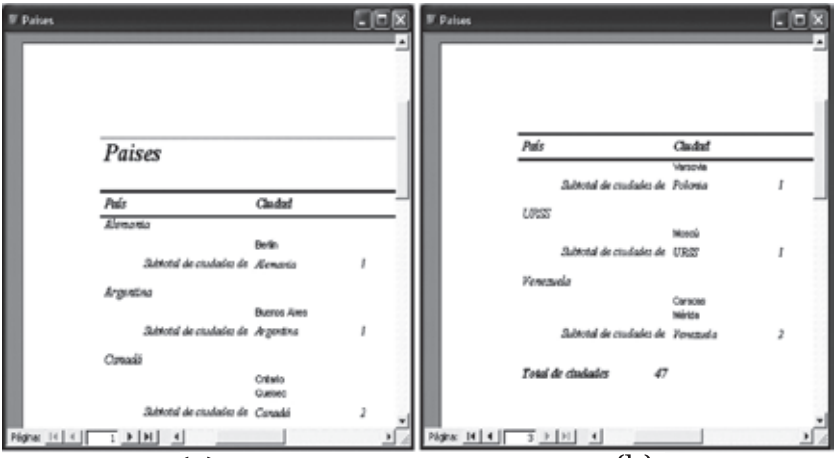
Por otro lado, en no pocas oportunidades el diseñador desea personalizar el informe construido con el asistente. Por ejemplo, desea agregar controles o disponerlos de forma diferente, cambiar tipos, grosores, colores, agregar imágenes o, quizá lo más frecuente, incorporar campos calculados que totalicen los datos dentro de grupos o para el conjunto general de los registros. La figura 65 presenta la primera y última páginas del informe *Ciudades y Países* (a y b), pre-visualizado en pantalla, incluyendo algunos controles añadidos luego de utilizar el asistente para informes.

La forma de pre-visualización de un informe muestra tal como saldrá por el dispositivo impresor del computador[11], para lo cual, el

[11] En inglés se denomina a esta característica *"What you see, what you get"* (WSWG), lo que en español sería algo así como "lo que ves, será lo que obtendrás".

usuario debe tener tal equipo instalado o bien tener acceso a un dispositivo de este tipo en la red.

En la figura 65(a) puede apreciarse que el nombre del "objeto" informe, coincide con el título del informe ("Países") que se muestra encabezando la página. Le sigue otro encabezado, esta vez con las etiquetas de los campos, que da sentido columnar a la página. Ya dentro del informe se despliegan los datos de países primero y, dentro de cada país, las ciudades presentes en la base de datos, cerrando la información de cada país con la cuenta de las ciudades que le son propias. Nótese que aun cuando el resultado de la consulta que da origen al informe repite el país para cada ciudad, el informe no lo hace. En el informe se agrupan las ciudades para cada ocurrencia de un país.



(a) (b)

Figura 65. Pre-visualización del informe de Países y Ciudades

En la figura 65 se ve claramente esta acotación al examinar las ciudades del Canadá: Ontario y Quebec. También vale la pena mencionar que un informe puede tener una extensión de varias páginas, las cuales pueden recorrerse a voluntad utilizando las barras de desplazamiento verticales u horizontales, así como los botones de desplazamiento por páginas ubicados en la parte inferior de la ventana.

La parte b de la figura muestra la última página del informe. Allí, además de lo ya dicho, se presenta ahora el total general de ciudades

del informe (47 ciudades). La figura 66 despliega la vista "diseño" del recientemente creado informe *Países*.

Además de los elementos comunes a formularios e informes (que pueden revisarse en la sección anterior del texto), un informe incorpora algunos objetos que comentaremos a partir del examen de la figura 66.

En primer término comentaremos que un informe construido a partir del asistente no incorpora automáticamente los controles que hemos añadido en la sección "Pie País" ni en la sección "Pie del informe". Dichos controles deben añadirse manualmente.

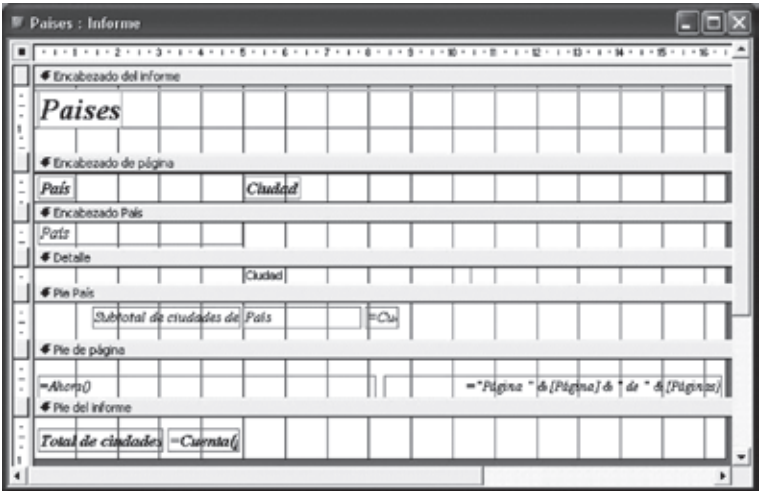


Figura 66. Vista "diseño" del informe "Países y Ciudades"

El informe mostrado en la figura 66 tiene las secciones siguientes:

- Encabezado del informe: Aquí (así como en la sección "Pie del informe"), se disponen controles que se refieren al informe en su totalidad y que serán mostrados una sola vez encabezando el documento. Todo cómputo que se haga en esta sección considera la totalidad de los datos contenidos en el informe. En el ejercicio de la figura 66, esta sección se destina al título del informe, escrito en letras de gran formato, con un control "etiqueta" cuyo valor es la cadena de caracteres "Países".
- Encabezado de la página: Sección prevista para contener información común a cada una de las páginas del documento (al igual

que la sección "Pie de página"). Lo que aquí se incluye se referirá a, y será mostrado en, cada página del informe. En el ejercicio aludido se disponen en esta sección los encabezados de las columnas, utilizando etiquetas con los nombres correspondientes a los datos de países y ciudades.

- Encabezado de grupo (País): Es común encontrar dentro de los informes, a continuación de la sección "Encabezado de página" y antes de la sección "Detalle", los encabezados de los grupos que el diseñador desea conformar para los datos que serán presentados en el documento. En el ejemplo se ha dispuesto un único grupo a partir de los valores del campo "País". Por supuesto, un informe puede contener múltiples grupos definidos en forma jerárquica, esto es, grupos dentro de grupos en anidamiento. La inexistencia de la sección "Encabezado del grupo" no necesariamente implica la inexistencia de un grupo. Pueden definirse grupos sobre los cuales no interese mostrar encabezado alguno, no obstante, en caso que exista un encabezado de grupo, todo lo que allí se disponga se referirá a cada instancia del grupo en particular. Por ejemplo, el cuadro de texto "País", vinculado al campo homónimo de la consulta que da origen al registro y dispuesto en el encabezado del grupo (también llamado "País"), cambiará su valor sólo cuando se hayan agotado las ciudades de cada particular país.
- Detalle: Así como en los formularios, la sección Detalle se destina a contener los controles vinculados con los campos de registros que formarán el informe. Todo lo que aquí se disponga se refiere entonces a cada registro particular de la fuente de datos. Para el ejemplo de la figura 66, el único campo que será desplegado individualmente será "Ciudad". En esta sección también es posible ubicar controles calculados, referidos a cómputos hechos sobre campos de un mismo registro.
- Pie de grupo (País): Con el mismo ámbito de acción que el encabezado, el pie cierra o culmina la información de cada grupo. En el ejemplo de la figura se han dispuesto secuenciadamente en esta sección una etiqueta rotulada como "Subtotal de ciudades de", un cuadro de texto vinculado al campo "País" y un cuadro de texto independiente, cuyos valores serán originados por la expresión "=Cuenta([Ciudad])". Estos controles muestran al

usuario el número de ciudades de cada país. La función "Cuenta" no es la única que puede utilizarse en este lugar, pueden utilizarse muchas otras de agregación como "Suma" o "Promedio", pero sin duda es la que aplica para valores cualitativos como los del ejemplo en cuestión. Este mismo conjunto de controles, con idénticos resultados, pueden disponerse en el encabezado del informe. Aun cuando la expresión no contiene indicación alguna, el procesador del informe entiende, debido a la ubicación de los controles mencionados, que éstos se refieren a cada elemento del grupo, y actúa en consecuencia.

- Pie de página: En esta sección del informe, el asistente ha incorporado automáticamente controles calculados que presentan la fecha de elaboración del informe (utilizando la función "Ahora()") y el número de páginas del documento (utilizando los campos "[Página]" y "[Páginas]", intrínsecos a todo objeto informe), concatenados apropiadamente. Todo aquello que se coloque en esta sección se referirá a, y se repetirá en, cada página.
- Pie del informe: Así como la sección "Pie del grupo" cierra o determina la culminación de un grupo particular, la sección "Pie del informe" determina el cierre o el término del informe en su totalidad. Todo lo que aquí se disponga, así como en la sección "Encabezado del informe", se referirá a la globalidad de los registros. En particular, en el ejercicio al que se refiere la figura 66 hemos dispuesto el control etiqueta con el valor "Total de ciudades" y el cuadro de texto con la expresión "=Cuenta([Ciudad])". Nótese que la expresión es idéntica a la utilizada en el pie del grupo "País", sin embargo, el procesador del informe, como se desprende del examen de la figura 65(b), entiende ahora que el conteo de las ciudades debe hacerse sobre todas ellas y no sólo sobre las de un grupo particular. Esto es debido exclusivamente a la ubicación del control, razón por la cual el diseñador debe tener claridad sobre el lugar en que debe colocar un campo calculado en función del ámbito de acción del cálculo que desea presentar en un informe.

La figura 67 muestra la ventana en la que se configuran grupos y opciones de ordenamiento del informe. A esta ventana se accede desde el menú secundario desplegado al dar clic con el botón derecho del ra-

tón sobre el informe, escogiendo la opción "Ordenar y agrupar". Puede verse que en la ventana de la figura se incluyen dos campos, "País" y "Ciudad". El primero se utiliza para definir un grupo, con encabezado y pie, automáticamente ordenado en forma ascendente, pero el segundo (aun cuando no se aprecia del todo en la figura), sólo se utiliza para efectos de ordenamiento. Esto produce que aparezca en el informe un grupo por cada país y, dentro de cada grupo, las ciudades ordenadas alfabéticamente.

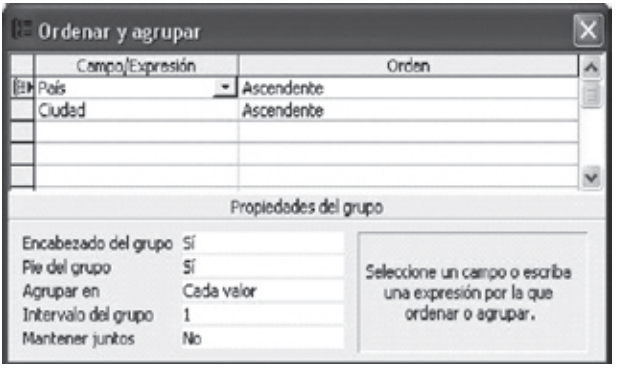


Figura 67. Definición de grupos y campos de ordenamiento

De igual forma que en los formularios, los informes, los subinformes, los demás controles, las secciones internas y prácticamente todos los elementos que le componen, tienen propiedades que pueden alterarse a voluntad del diseñador. Entre estas propiedades también se encuentran aquellas susceptibles de ser programas con VBA en respuesta a eventos que, sin embargo, y considerando la naturaleza más bien estática de los informes, son un conjunto mucho más reducido que para el caso de los formularios. El Apéndice C (en CD) muestra las propiedades de objetos seleccionados del informe. Se omiten allí algunos de los objetos, bien porque recibieron su descripción en la sección anterior dedicada a los formularios, o bien porque por su naturaleza pierden sentido para los informes (tal es el caso de los botones de comando, grupos de opciones y cuadros combinados, entre otros).

La figura 68 presenta un informe más completo que el anterior, que entrega ahora la información de cada libro. Como puede notar el lector, contiene algunos elementos adicionales a los discutidos para el

informe de la figura 65. El documento resultante muestra el número total de ejemplares en la base de datos como parte del encabezado, discrimina los registros agrupándolos por el identificador del tema y sobre cada grupo, calcula el número de libros y su precio promedio. Dentro de grupos, ordena alfabéticamente los libros por su título e incluye, para cada libro, además de la información contenida en la tabla *Libros*, sus autores y sus descriptores.

El informe de la figura 68 ha sido creado utilizando primeramente el asistente para informes, siguiendo los pasos que lista la figura 64, no obstante, contiene una serie de añadidos que lo hacen más complejo que aquél. En primer término, la base del informe no es la tabla *Libros*, sino una consulta basada en dicha tabla, que la combina con la tabla *Temas*, extrayendo el campo "Tema".

El "Origen del registro" del informe "Libros" es una consulta llamada "Libros para el informe", cuyo contenido SQL es el siguiente:

```
SELECT Libros.Id_Libro, Libros.Título, Libros.
Id_Tema, Temas.Tema, Libros.País, Libros.Ciudad,
Libros.Año, Libros.Precio, Libros.
Fecha_Adquisición FROM Temas INNER JOIN Libros
ON Temas.Id_Tema = Libros.Id_Tema;
```

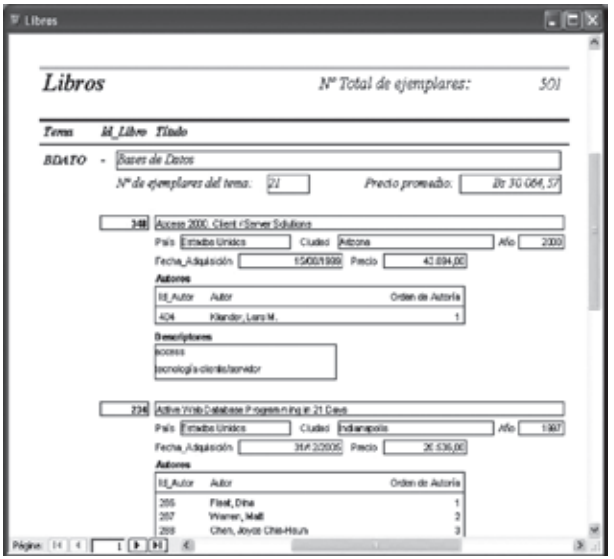


Figura 68. Informe completo de libros

Este origen de datos tiene el propósito de vincular los libros y los temas, antes de construir el informe, para facilitar la presentación de la descripción de cada tema, además de su identificador. Otras modificaciones importantes llevadas a cabo a partir del producto que entrega el asistente para informes, tienen que ver con la forma. Como puede verse en la figura 68, se han enmarcado los campos de datos, se han unificado los tipos de letra para los controles de una misma sección, se han añadido letras negritas en algunos subtítulos y se han alineado y dispuesto convenientemente los controles para mejorar su presentación. El diseño se muestra en la figura 69.

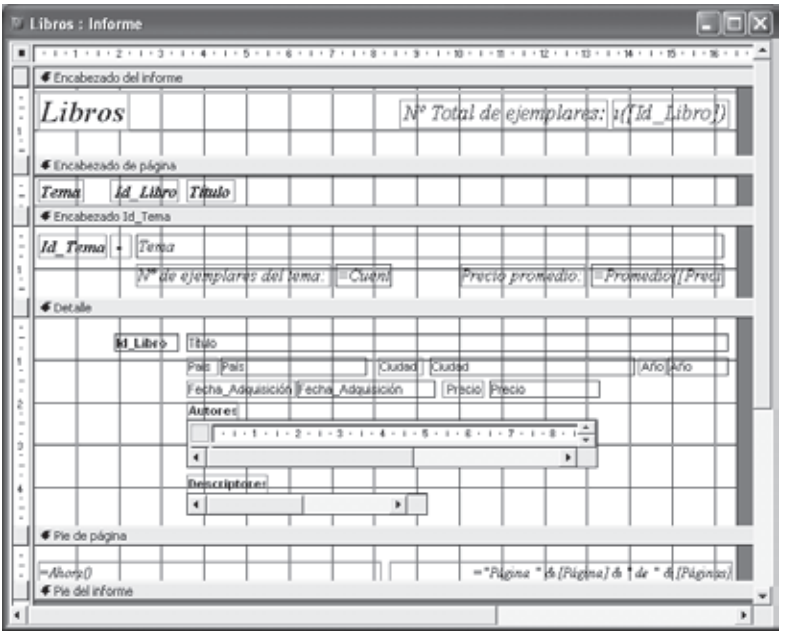


Figura 69. Vista diseño del informe "Libros"

- En el encabezado del informe se ha incluido, además de una etiqueta con el valor "Libros" (que sirve de título al informe), un cuadro de texto independiente y su etiqueta, en el que se calcula el total de libros que contiene nuestra base de datos. La instrucción en la propiedad "Origen del control" del cuadro de texto, es: "=Cuenta([Id_Libro])". En la figura no se aprecia muy bien

su contenido, pues el cuadro ha recibido la orden de mostrarse justificado a la derecha y ha recibido un tamaño acorde con el número que se supone debe mostrar.

- En el encabezado de página sólo hay tres etiquetas para tema, identificador del libro y título respectivamente, pues el resto de la información del documento se presenta en forma continuada para cada registro e incluye sus propias etiquetas.
- En el encabezado del grupo según "Id_Tema" se han incluido, además de la información relativa al tema en sí mismo, dos controles independientes. El primero destinado a contar los libros de cada grupo y el segundo destinado a calcular el precio promedio de los libros del grupo. Este segundo cuadro de texto tiene un "Origen del control" dado por la expresión "=Promedio([Precio])".

Nótese que en la sección detalle se han incluido para cada libro dos subinformes vinculados al original a partir del campo "Id_Libro". El primero de ellos se dedica a presentar los autores y el segundo a los descriptores. El subinforme para los autores parte de una consulta "Autores para informe" cuya instrucción SQL es la siguiente:

```
SELECT [Autores de Libros].Id_Libro, [Autores de
Libros].Id_Autor, [Apellidos] & ", " & [Nombres]
AS Autor, [Autores de Libros].[Orden de Autoría]
FROM Autores INNER JOIN [Autores de Libros] ON
Autores.Id_Autor = [Autores de Libros].Id_Autor;
```

De la instrucción SQL se deduce que se seleccionan el identificador del libro, el identificador del autor, un campo que se ha compuesto de la concatenación de los campos "Apellidos" y "Nombres" del autor y el orden de autoría. Por otro lado, el subinforme para los descriptores se basa en la tabla homónima (no en una consulta) e incluye los campos "Id_Libro" y "Descriptor". Ambos subinformes se muestran respectivamente en las secciones (a) y (b) de la figura 70.

Nótese en la figura 70 que, aunque hemos dicho que en ambos subinformes se incluyó el campo "Id_Libro", en realidad éste no aparece en ninguno de los dos. Efectivamente, el campo en cuestión participa del "Origen del registro" en ambos subinformes para efectos de establecer el vínculo apropiado con el informe original "Libros", no obstante, en vista de que los datos de los subinformes se mostrarán en el marco

de la información de los libros y esta información contiene ya el identificador de cada libro, el campo "Id_Libro" que podría colocarse en ambos subinformes ha sido eliminado de su presentación (más no, por supuesto, del origen de datos).

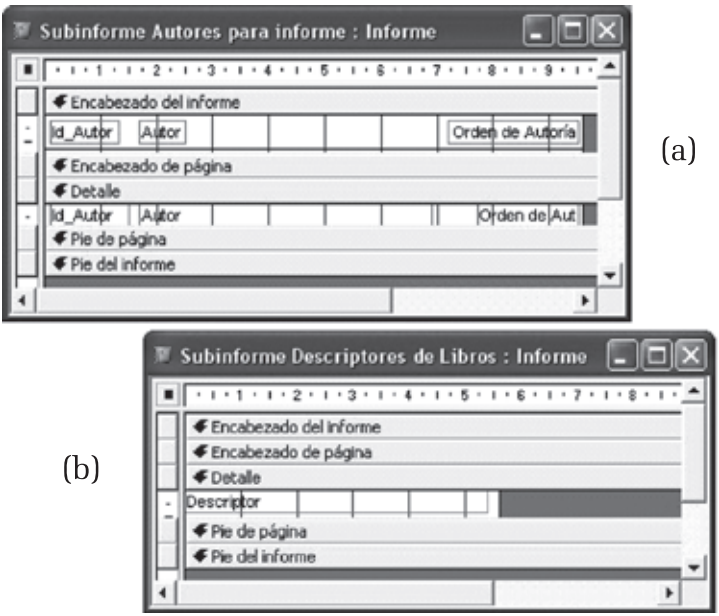


Figura 70. Vista diseño de los subinformes del informe "Libros"

Por último, y para finalizar esta sección dedicada a los informes, expondremos ahora un procedimiento considerablemente más laborioso cuyo resultado es definitivamente muy apreciable: la elaboración de gráficos y su presentación en informes.

Supongamos que deseamos presentar un informe que contenga discriminada por temas, la inversión total en libros que ha registrado nuestro sistema. La forma más sencilla de comenzar este informe es construyendo una consulta que, agrupando por temas, sume los precios de todos los libros. Sin embargo, pensemos que consideramos esta lista muy precaria y queremos añadir un gráfico que muestre la información de manera más atractiva.

El tipo apropiado para presentar tal información es un gráfico circular o "de torta", que debería contener el pedazo del pastel de la

inversión que consume cada tema. El primer problema a resolver es que la lista de todos los temas, aun cuando éstos se encuentran agrupados para sumar los precios de los libros asociados, es demasiado extensa para construir un gráfico circular automático (como puede comprobarse, la base de datos contiene libros de 48 temas diferentes). Entonces, nuestra primera acción será construir un juego de consultas para obtener una lista con los temas que, por ejemplo, montan más de 300 mil bolívares invertidos, incluyendo un último registro que englobe a todos los temas que montan menos de esta cantidad, en un solo rótulo, digamos, "Otros". Así obtendremos una tabla de valores totales en cuanto a la inversión por tema, pero con sólo unos pocos registros, como por ejemplo, los mostrados en la figura 71.

Figura 71. Resumen de la inversión en libros por tema

Para obtener el resultado mostrado en la figura 71 son necesarias varias consultas previas. Veamos una descripción de cada una en la tabla 12.

TABLA 12. CONSULTAS NECESARIAS PARA LA LISTA RESUMIDA DE TEMAS Y PRECIOS

Instrucción SQL	Comentarios
`SELECT Id_Tema, Sum(Precio) AS InvTotal FROM Libros GROUP BY Id_Tema HAVING Sum(Precio)>300000;`	Esta primera consulta, diseñada y almacenada con el nombre "Inversión en libros por tema mayor 300", calcula la suma de los precios de libros (renombrada como "InvTotal") agrupando para cada "Id_Tema", pero sólo para aquellos registros en los que dicha suma sea superior a los 300 mil bolívares.
`SELECT Id_Tema, Sum(Precio) AS Inversión FROM Libros GROUP BY Id_Tema HAVING Sum(Precio)<=300000;`	Esta segunda consulta, muy parecida a la primera, diseñada y almacenada con el nombre "Inversión en libros por tema menor 300", calcula ahora la suma de los precios de libros (renombrada como "Inversión") agrupando para cada "Id_Tema", pero sólo para aquellos registros en los que dicha suma sea inferior o igual a los Bs. 300 mil.
`SELECT "OTROS" AS Id_Tema, Sum(Inversión) AS InvTotal FROM [Inversión en libros por tema menor 300];`	Esta tercera consulta toma los valores de la segunda y, considerándolos un solo grupo, suma el campo Inversión, llamándolo "InvTotal". Por otro lado, agrega lo que aparenta ser un campo, llamado Id_Tema, que contendrá un valor constante, cadena de caracteres ("OTROS"). La intención es que el formato de la consulta resultante tenga los mismos campos que la primera, pero con un solo registro que muestra el par conformado por la cadena "OTROS" y la suma de todos los precios de libros que, considerados por temas, tienen precios inferiores o iguales a los 300 mil bolívares. A esta consulta almacenada se le ha llamado "Inversión en libros por tema menor 300 TOTAL".
`SELECT * FROM [Inversión en libros por tema mayor 300] UNION SELECT * FROM [Inversión en libros por tema menor 300 TOTAL];`	Ahora, finalmente, se unifican las consultas primera y tercera formando una sola que produce los resultados mostrados en la figura 71. A esta consulta se le ha llamado "Inversión en libros resumen" y, como resulta evidente, es una consulta de unión.

Una vez diseñada la fuente de datos, el siguiente paso es elaborar el informe. Para ello, partiendo esta vez de un informe en blanco, es decir, sin utilizar el asistente, podemos lograr un documento que en la vista diseño se parece al de la figura 72. En dicha figura vemos un informe con apenas tres elementos: una etiqueta con el título del informe, un cuadro de gran tamaño contentivo de un gráfico (que se muestra como una torta de tres pedazos) y un subinforme cuyo origen de datos corresponde a la consulta de la figura 71.

El subinforme incluye en la sección "Pie del informe" un control calculado como "=Suma([Inversión])", cuyo propósito es que la tabla a mostrar en el documento sume todos los precios de los libros agrupados en temas.

Figura 72. Vista diseño del informe "Resumen de Inversión en Libros"

Para diagramar el gráfico que se muestra en la figura 72, estando abierto el informe en la vista diseño, se selecciona la secuencia de opciones "Insertar" → "Gráfico…" del menú principal del *Access*. Cuando se escoge esta alternativa se desencadena un asistente para gráficos que sigue los pasos ilustrados en la figura 73.

El primer paso (a) es la selección de la fuente de datos sobre la que será construido el gráfico, en este caso la consulta almacenada con el nombre "Inversión en libros resumen". El segundo paso (b) permite seleccionar los campos de la fuente de datos que participan en la elaboración del gráfico, en este caso "Id_Tema" e "InvTotal". El tercer paso (c), presenta una serie de alternativas a escoger sobre el tipo de gráfico que se desea construir. En nuestro caso hemos seleccionado el tipo "Gráfico circular en 3D".

El cuarto paso (d) muestra una suerte de ejemplo del gráfico que ha sido interpretado por el asistente. En este caso, el asistente, por defecto, interpreta que la serie de datos a graficar la conforman los "Id_Tema"

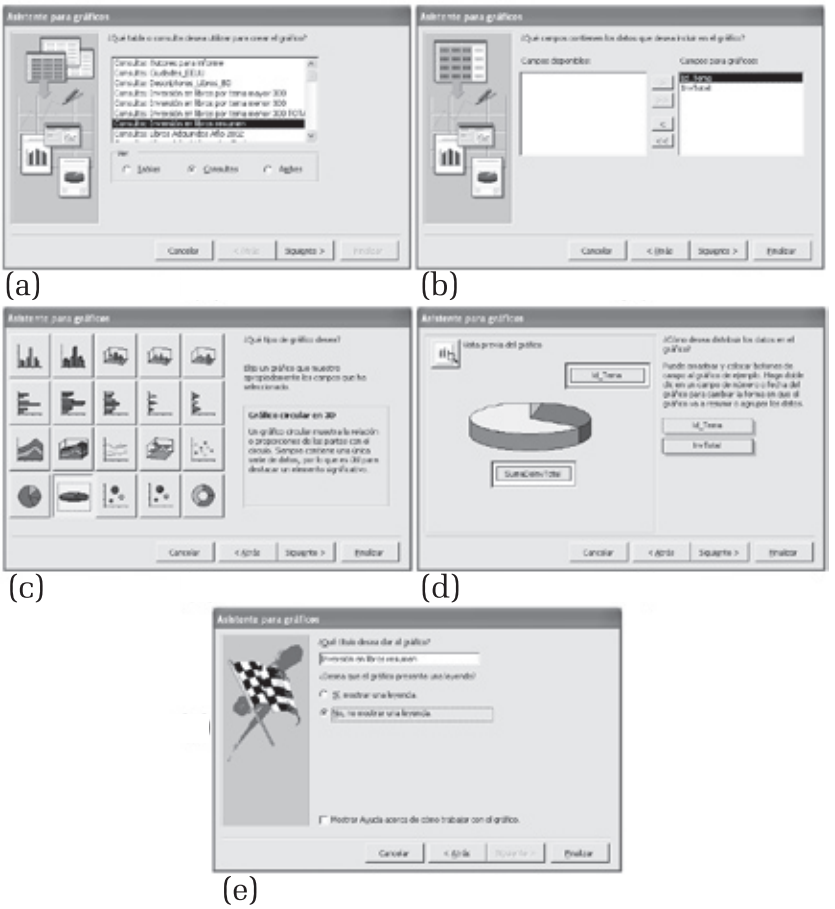


Figura 73. Pasos del asistente para gráficos

y que los valores numéricos que forman las proporciones que dará a la torta, provienen del campo "InvTotal". Normalmente, el asistente adopta una función agregada de dominio (como suma o promedio) que aplicará al campo numérico, sin embargo, en nuestro ejercicio no requerimos de función alguna, por lo cual hemos dado doble clic sobre el cuadro "InvTotal" y escogido la opción "Ninguna", ordenando así al asistente que los valores a distribuir sean directamente los del campo "InvTotal", sin sumarizarse.

El último paso (e) contempla la definición del nombre que daremos al gráfico y si se deberán o no mostrar etiquetas de la serie de valores. El nombre sugerido es el mismo que el del informe y hemos

seleccionado no mostrar etiquetas, ya que esperamos incluir la serie de valores en el pastel. Para hacer esto último, una vez ubicado el gráfico en el área del informe y alterado su tamaño hasta obtener la cobertura deseada, debe darse doble clic sobre él para acceder a sus propiedades.

Son variadas las propiedades de un gráfico y, cuando se accede a ellas, el menú de opciones del sistema cambia. Una en particular es de relevancia, pues ha sido alterada en el ejercicio que nos ocupa: disponer las etiquetas de la serie de valores en el propio gráfico e incluir el porcentaje de participación de cada tema en la inversión total. En el menú principal de *Access*, ahora hay una opción "Gráfico" y, dentro de ésta, una "Opciones de Gráfico" que al seleccionarse despliega una ventana como la de la figura 74.

Como puede apreciarse en la figura 74, para lograr que datos y porcentajes se dispongan en el interior del gráfico circular, basta seleccionar en la pestaña "Rótulos de datos" las opciones "Nombre de la Categoría" y "Porcentaje".

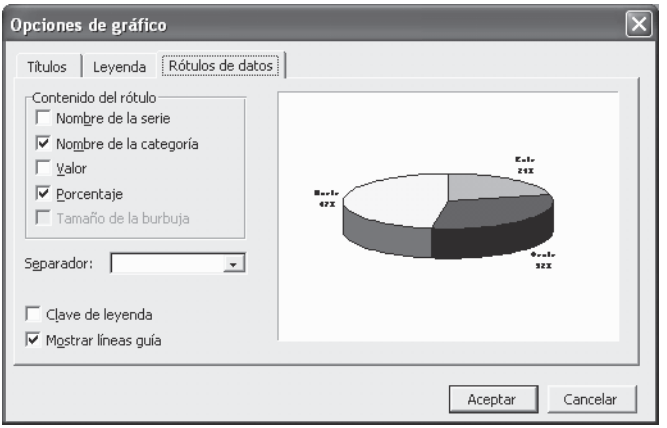
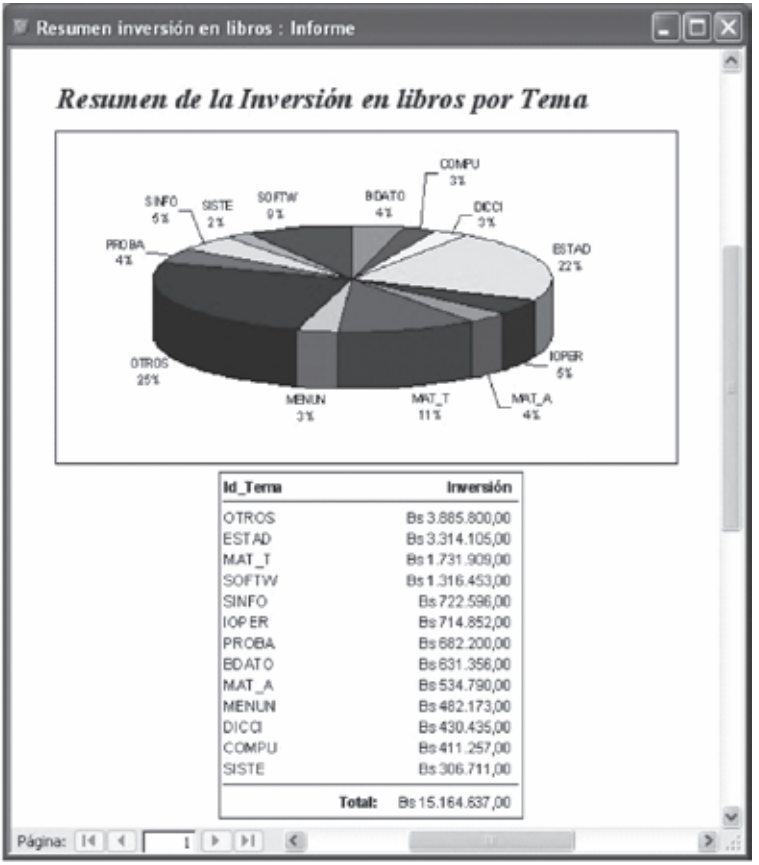


Figura 74. Opciones de gráfico. Pestaña "Rótulos de datos"

El gráfico, así como la tabla del subinforme, se construyen dinámicamente a partir de los datos del sistema cada vez que se despliega el informe. Note el lector la gran ventaja que representa contar con una herramienta en la cual se diseña una sola vez un gráfico y luego se le despliega tantas veces como se desee, de manera que el informe resultante siempre contiene el gráfico actualizado con los datos más recientes.

Ventaja ésta que sólo puede lograrse en un ambiente de bases de datos y sistemas de información. Una vez creado, configurado y guardado el informe, la figura 75 lo despliega tal como aparecerá impreso.



Id_Tema	Inversión
OTROS	Bs 3.885.800,00
ESTAD	Bs 3.314.105,00
MAT_T	Bs 1.731.909,00
SOFTW	Bs 1.316.453,00
SINFO	Bs 722.596,00
IOPER	Bs 714.852,00
PROBA	Bs 682.200,00
BDATO	Bs 631.356,00
MAT_A	Bs 534.790,00
MENUN	Bs 482.173,00
DICCI	Bs 430.435,00
COMPU	Bs 411.257,00
SISTE	Bs 306.711,00
Total:	Bs 15.164.637,00

Figura 75. Informe "Resumen de la Inversión en libros por Tema"

3.8 Interfaces de órdenes. Menús

Un menú es un objeto al que todo usuario de ambientes gráficos en computadora está habituado. Se trata de un área de la aplicación en la cual se disponen opciones, a veces iconizadas, a veces textuales y a veces de ambas formas que, vía ratón o teclado, pueden ser selec-

cionadas, convirtiéndose en la manera como el *software* responde a su voluntad.

En el caso de los sistemas de información, el menú cumple el doble propósito de facilitar la operatividad del sistema, así como de organizar de manera coherente las distintas actividades que el usuario puede llevar a cabo.

El primer paso en la construcción de todo menú de opciones es la planificación de las facilidades que deberán proporcionársele al usuario para que el sistema cumpla con la finalidad encomendada. El menú, aun cuando puede diseñarse en cualquier momento, se incluye en realidad al final de todo el trabajo, cuando los elementos componentes del sistema ya han sido materializados.

Supongamos, para el ejemplo de los libros que seguimos, que deseamos lograr una interfaz de órdenes jerárquica que permita al usuario las acciones que se muestran en la figura 76. El menú a construir contiene cuatro opciones principales, las tres primeras del tipo submenú (con cuatro, tres y tres opciones de acción respectivamente), y la cuarta del tipo opción de acción, directamente.

Figura 76. Diseño del menú principal del sistema "Libros"

La idea es que accediendo al submenú "Editar Datos", el usuario encuentre concentrados todos los formularios que le permiten incorporar o modificar los registros de la base de datos. Haciendo lo propio con el submenú "Consultar Datos" pueda desplegar tres de las consultas que hemos preparado con antelación. Dentro del submenú "Imprimir

Informes" encuentre tres de los documentos prediseñados para ser impresos. Y, por último, dando clic sobre la opción "Salir" abandone la sesión de trabajo con el sistema.

Lo primero que debemos hacer antes de llevar a la práctica el menú diseñado es preparar las acciones que estarán disponibles. Para ello se construyen "macros" que ejecuten cada acción deseada.

Las macros son poderosos objetos que agrupan comandos y los ejecutan en lotes cuando se les invoca. Se almacenan en la ventana de base de datos, en la pestaña "Macros". Para el caso debemos construir un total de once macros distintas. Una para cada apertura de formulario asociada con las opciones de "Editar Datos" (cuatro en total), una para cada apertura de consulta asociada con las opciones de "Consultar Datos" (tres en total), una para cada apertura de informe asociado con las opciones de "Imprimir Informes" (tres en total), y una última que desencadene la salida del sistema.

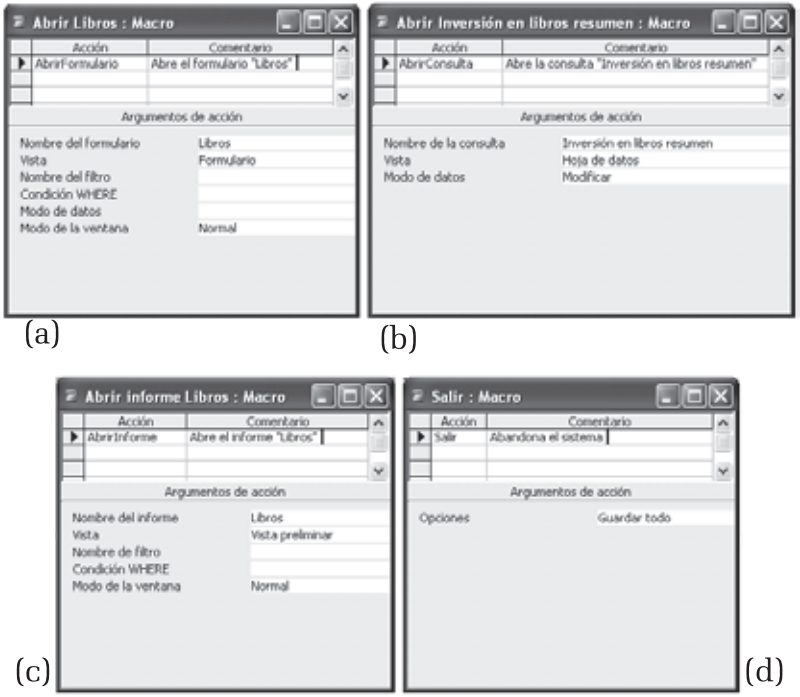


Figura 77. Macros creadas para ejecutar acciones comunes

La figura 77, secciones (a), (b), (c) y (d), muestra cuatro ejemplos de la creación de macros, una para cada tipo de las necesarias en el ejercicio. Supondremos que las restantes se crean de manera similar y por tanto las omitiremos. Para crear una macro debe oprimirse el botón "Nuevo" de la pestaña "Macros" en la ventana de base de datos.

La figura 77(a) muestra la configuración de una macro cuya acción es abrir el formulario "Libros". La figura 77(b) muestra una macro que ejecuta la acción de abrir una consulta denominada "Inversión en libros resumen". La figura 77(c) ejemplifica una macro que abrirá el informe titulado "Libros", y la macro mostrada en la figura 77(d) produce la acción de abandonar el sistema. La única de las macros que recibió un parámetro distinto a los previstos por defecto fue la de la figura 77(c). Por defecto, el parámetro "Vista", en una macro que abre un informe, es "Imprimir". No obstante, en la macro que esquematiza la figura lo hemos alterado para que el informe no sea impreso inmediatamente, sino que se presente primero en una vista preliminar.

Una vez creadas las macros que necesitamos, el siguiente paso es crear el nuevo menú de opciones del sistema y añadir los elementos previstos. Del submenú "Herramientas", a partir del menú principal del *Access*, se selecciona la opción "Personalizar". En la pestaña "Barra de herramientas" de la ventana que se despliega a continuación, dando clic en el botón "Nueva", se crea una nueva barra titulada "Menú MisLibros" y se selecciona para ser mostrada, tal como sugiere la figura 78(a).

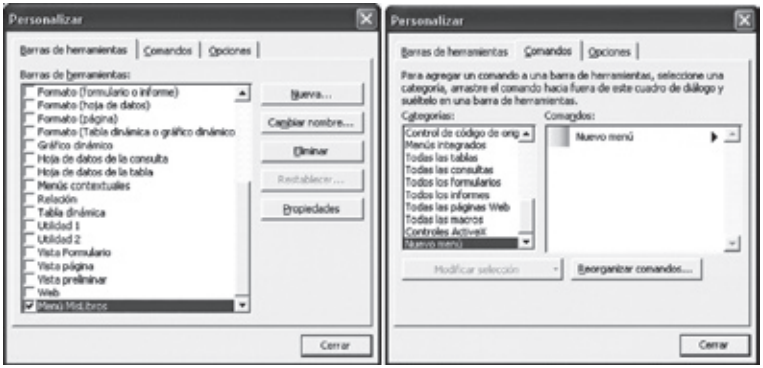


Figura 78. Creación de una nueva barra de herramientas

Al momento, en algún lugar de la pantalla aparecerá una nueva barra de herramientas flotante con el nombre que le hemos asigna-

do y completamente vacía. Para agregar submenús a la nueva barra, desde la pestaña "Comandos" de la ventana "Personalizar", en la categoría "Nuevo menú" [ver figura 78(b)], se toma y arrastra la opción "Nuevo menú" y se suelta sobre la barra de herramientas flotante que se encuentra en la pantalla. Este proceso se repite dos veces más, puesto que la barra de herramientas que estamos construyendo necesita tres submenús.

El último de los elementos principales de la nueva barra de herramientas ya no es un submenú, sino una opción que ejecuta la acción de salir del sistema. Para ubicar este último elemento principal en la barra, desde la pestaña "Comandos", ahora en la categoría "Archivo" (la primera de la lista), se toma y arrastra la opción "Personalizado" hasta la última posición de la barra de herramientas flotante. De esta forma se completa el primer nivel en la jerarquía del nuevo menú, el cual debe parecerse al mostrado en la figura 79(a).

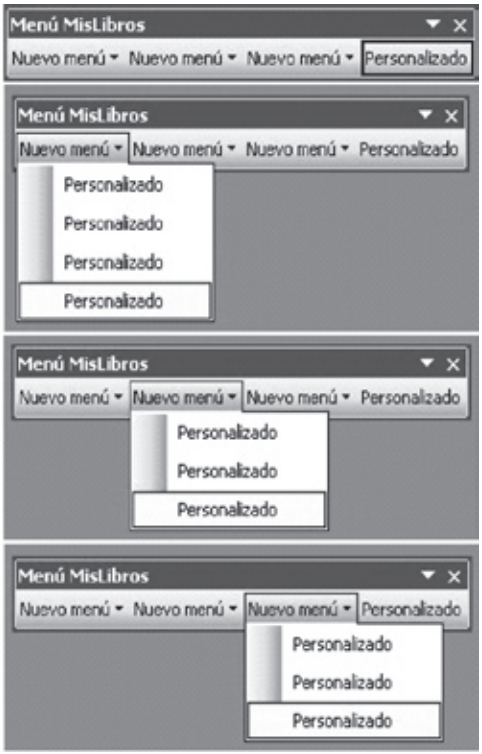


Figura 79. Estructura del "Menú MisLibros" de una nueva barra de herramientas

Para completar el segundo nivel de la jerarquía del menú se requieren cuatro opciones para el primer submenú y tres para cada uno de los dos submenús restantes. Cuidadosamente se arrastra la opción "Personalizado" hasta soltarla, ya no sobre la barra de herramientas principal, sino dentro de cada submenú particular. La figura 79, en las secciones (b), (c) y (d) muestra la estructura definitiva que debemos alcanzar en la nueva barra de herramientas. Falta ahora particularizar cada ítem del menú.

La figura 80 muestra las características personalizables de los elementos de un menú. La sección (a) de la figura despliega las características para los submenús, en particular para el submenú "Editar Datos", y la sección (b) hace lo propio con la opción de acción, en particular la opción "Salir". Para personalizar un elemento del menú, el diseñador debe encontrarse en posición de llevar a cabo dicha tarea (Menú Principal → Herramientas → Personalizar). Visualizando el menú creado, como se muestra en la figura 79, haciendo clic con el botón secundario del ratón sobre el elemento que desea personalizar y escogiendo en el menú emergente la opción "Propiedades", se despliegan las ventanas a las que se refiere la figura 80.

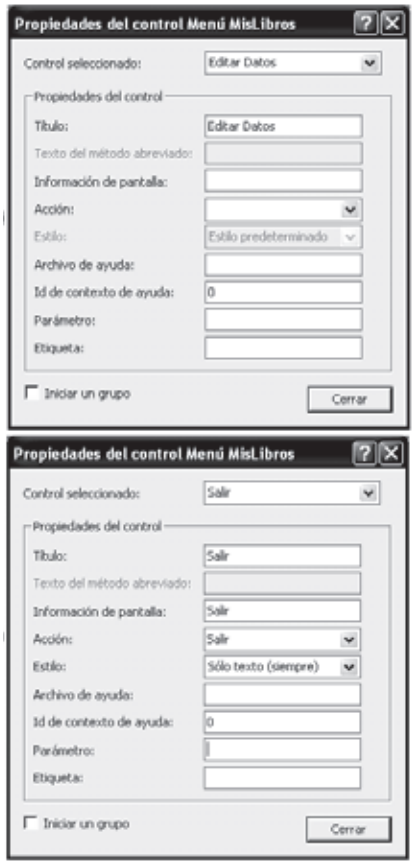


Figura 80. Características de los elementos de un menú

Nótese que la única característica que se ha alterado para el submenú "Editar Datos" es el título, claro está, debido a que un submenú no realiza acción alguna, sólo es un contenedor de otras opciones y submenús. Por otro lado, cuando se trata de una

opción de acción, como es el caso en "Salir", además del título debe indicarse el nombre de la macro que contiene la acción, en este caso, la macro se llama también "Salir".

Las restantes opciones del nuevo "Menú MisLibros" se personalizan de igual manera como se ha ejemplificado con la figura 80. Debe lograrse entonces una barra de herramientas flotante como la que se aprecia en la figura 81.

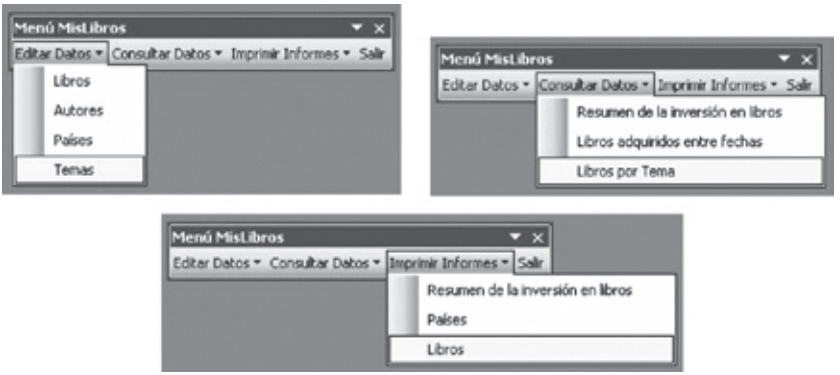


Figura 81. "Menú MisLibros" con sus características completas

El sistema entiende lo que hemos creado como una barra de herramientas flotante que contiene opciones, no obstante, para que este elemento sea un menú debe ocupar un lugar de preferencia fijo (y no flotante) en la ventana y ser considerado como tal. Esto se consigue simplemente tomándolo del título y arrastrándolo hasta el lugar deseado, por una parte, y por la otra, ajustando su propiedad "Tipo de barra de herramientas" a "Barra de menús", invocada dentro de la pestaña "Barra de herramientas" que muestra la figura 78(a). Se sugiere colocarlo después del menú normal del *Access* y antes de cualquier otro menú que esté desplegado para el momento.

Cuando ya se esté conforme con todos los elementos que componen el sistema, el único menú que deberá mostrarse será el menú personalizado que se haya creado (en este caso el "Menú MisLibros"), ocultando también la ventana de base de datos. Esta última configuración es importante pues el diseñador seguramente no desea que el usuario actúe sobre el sistema de manera diferente a como él ha previsto al construir los menús. Así pues, deberán deshabilitarse todas las

opciones habituales del *Access*, incluida la ventana de base de datos, y dejar disponibles las opciones que proporciona el menú que se ha creado. Esto se consigue en la opción Herramientas → Inicio, como muestra la figura 82[12].

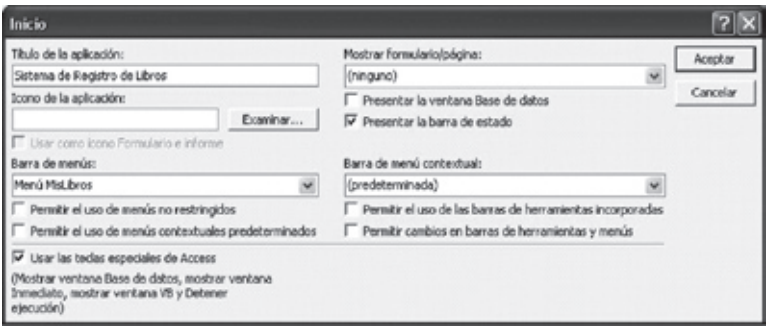


Figura 82. Opciones de inicio de la aplicación

Una aplicación como la que hemos creado, cuando sea liberada al usuario deberá lucir de forma similar a como se ilustra en la figura 83.

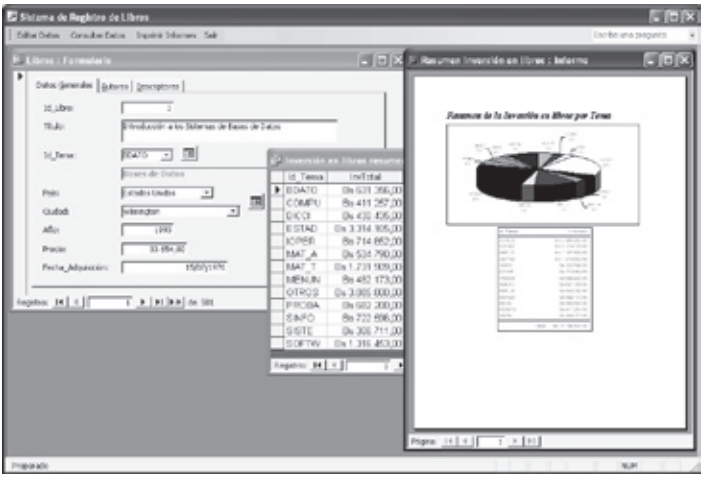


Figura 83. La interfaz del Sistema de Registro de Libros

[12] Para abrir la aplicación obviando las características de inicio, en lugar de dar doble clic sobre el icono que represente la aplicación en el escritorio de Windows debe darse la combinación Shift–doble clic.

3.9 Publicación de datos en la Internet

La Internet se ha transformado en un medio potente y apetecible de comunicación entre las personas, en consecuencia, toda organización moderna desea tener presencia en la red. Ahora bien, dicha presencia en la Internet sólo se logra construyendo un sitio Web que luego se publique en un servidor. Quienes a través de la red encuentren y visiten el sitio Web creado, estarán captando la información que la organización les proporciona.

Un sitio Web tiene cuando menos dos formas de ser concebido: estáticamente o dinámicamente. Se puede tener presencia estática en la red si se configura un sitio Web con información de texto, imágenes u otra de manera que para ser actualizada tengan que modificarse archivos físicos en el servidor. Por otra parte, para obtener presencia dinámica en la Internet es necesario que el sitio Web esté conectado con alguna base de datos, de manera que los contenidos que se publiquen se obtengan dinámicamente a partir de consultas en dicha base de datos. En otras palabras, un sitio Web dinámico cuenta con bases de datos de las cuales se extrae y presenta la información al mismo momento de ser consultada. Se denominan dinámicos precisamente debido a que sin necesidad de que quien mantenga el sitio altere archivos físicos, la información que se despliega puede cambiar de un instante al siguiente si ha cambiado la fuente de datos. Así, sólo organizaciones que cuentan con sistemas de información y bases de datos para el apoyo de sus actividades pueden lograr presencia dinámica en la Internet.

Como este texto versa sobre la creación de sistemas de información y bases de datos, es lógico que dediquemos algunas líneas a la tarea de poner en la Internet datos extraídos de nuestros sistemas.

En general, el trabajo asociado con la Internet es en sí mismo una especialidad. Para poder acceder a ella como usuario sólo se necesita una computadora conectada a la red mediante un ISP (*Internet Service Provider* o Proveedor de Servicios Internet) y un navegador Web. No obstante, para acceder a ella como proveedor de contenidos, esto es, publicando información propia, hace falta mucho más que eso. Para publicar contenidos en la Internet se requiere un servicio de alojamiento de páginas Web, es decir, se necesita que alguien proporcione un servidor de páginas Web en el que sea posible copiar los contenidos que se desean publicar y al que el resto del mundo tenga acceso. Dicho servidor

debe ser bien administrado para que se desempeñe correctamente, esté permanentemente conectado y proporcione los mecanismos de seguridad que garanticen que no se violentan los contenidos a él confiados. Todo servicio de este tipo requiere entonces de un equipo de trabajo e incluye *hardware* y *software* apropiados. No es materia de esta obra la configuración de servicios Web, así que sólo nos referiremos a ello tangencialmente y daremos por hecho que el lector cuenta con un proveedor de alojamiento Web estable y correctamente configurado.

Supondremos además que las herramientas de *Microsoft* para la publicación de contenidos, específicamente el servidor de páginas Web IIS (por *Internet Information Server*), la tecnología ASP (por *Active Server Pages* o Servidor de Páginas Activas), *Front Page* (el diagramador de páginas Web de *Microsoft*), ODBC, así como el habitual HTML (por *Hyper Text Markup Language* o Lenguaje de Hipertexto), están disponibles.

Volviendo a la idea original, se trata entonces de buscar una forma de aprovechar los datos que se capturan con el sistema de información y su base de datos para publicarlos de alguna manera en la Web y que proveedores, clientes, amigos, con o sin restricciones, puedan consultarlos en el ciberespacio. Son muchos los ejemplos en los que puede ser útil la publicación de información dinámica en la Internet: una librería que pone en la red el catálogo de sus libros y consigue con ello más clientes, una ferretería que publica su inventario de partes a la venta y consigue con ello que los interesados le visiten con mayor frecuencia y para fines específicos, una universidad que publica en la red las carreras que ofrece y las fechas importantes para inscribirse en ellas, una oficina gubernamental que desea tramitar documentos oficiales a través de la red, proporcionando con ello un mejor servicio a los ciudadanos, una persona que, como nosotros, tiene un buen registro de sus libros y desea compartirlo con sus amigos... y un sinfín más.

Todo sitio dinámico requiere de una estrategia. Puede ser que se desee obtener datos del mundo exterior y almacenarlos en bases de datos propias. Puede ser que sólo se desee mostrar datos que ya se encuentran en las propias bases de datos. Puede ser, claro está, que se desee una combinación de ambos. Capturar datos de la Internet y almacenarlos en bases de datos propias implica una comunicación bidireccional entre el usuario del sitio Web y el propietario de la base de datos. Esto significa que se necesita mucho más control y mucha más atención sobre los problemas de seguridad. No se sabe de antemano qué

datos llegarán y si éstos pueden o no contener información maliciosa que dañe los sistemas. Por otra parte, el segundo caso es más sencillo y las consideraciones de seguridad, aunque siempre importantes, son de menor complejidad. Al nivel introductorio que pretendemos aquí, discutiremos sólo el segundo caso.

Supongamos que nuestro sistema de registro de libros ya está terminado y funcionando. Deseamos ahora poner a disposición de amigos, familiares o de nosotros mismos una pequeña parte de la información con que contamos utilizando la Internet. Nos gustaría tener un sitio Web en el que el visitante pudiera consultar todos los libros que están registrados en el sistema, sus autores y aquellos libros filtrados de acuerdo con algún año de publicación. Apenas tres cosas sencillas que servirán como ejemplo de muchas otras que podrían hacerse. Entonces, a grandes rasgos, una de las muchas formas en que podríamos proceder es la siguiente:

1. Adicionar una clave de seguridad a la base de datos "MisLibros.mdb".
2. Crear un registro DSN (por *Data Source Name* o Nombre de Origen de Datos) de Sistema, vía ODBC, en la computadora que servirá de puente a la conexión con la base de datos (que puede ser o no la misma computadora que contenga la base de datos, en nuestro caso será la misma).
3. Activar y configurar correctamente el IIS como servidor de páginas Web en la computadora que contendrá nuestro sitio (esta tarea la daremos por realizada en vista que escapa del alcance del texto).
4. Diseñar y crear nuestro sitio Web de Libros, utilizando *Front Page,* de manera que se conecte a la base de datos vía el registro DSN creado en 2.
5. Publicar el sitio y comprobar su funcionamiento.

3.9.1 Adicionar una clave de seguridad a la base de datos

Para completar este primer paso debe abrirse el *Access* sin cargar la base de datos. Una vez abierto *Access,* las opciones "Archivo" → "Abrir" deben ser invocadas, de manera que en el cuadro de diálogo resultante se seleccione el archivo de base de datos ("MisLibros.mdb") y se le abra en modo exclusivo (ver figura 84).

Abierto el archivo de base de datos en modo exclusivo, las opciones del menú principal del *Access* "Herramientas" → "Seguridad" →

Figura 84. Cómo abrir un archivo de base de datos en modo exclusivo

"Establecer contraseña para la base de datos…" conducen a otro cuadro de diálogo como el de la figura 85, en el cual el diseñador puede introducir la clave que desee repitiéndola para garantía de no equivocarse (supongamos "1234"). De ahora en adelante, el archivo sólo se abrirá para el usuario "admin" y la clave "1234".

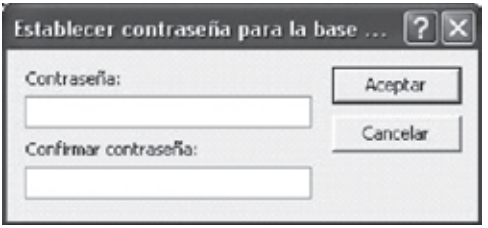


Figura 85. Cómo establecer una contraseña para la base de datos

Existen otras formas de dar seguridad mediante claves a los archivos de bases de datos. Una de las importantes, que requiere reflexión y diseño, es definiendo grupos, usuarios y asignando permisos sobre los distintos objetos de la base de datos; sin embargo, por simplicidad nos conformamos aquí con asumir que el usuario del sistema es uno solo ("admin") y su clave para ingresar es "1234".

3.9.2 El registro DSN del sistema

El segundo paso se da ya no dentro del *Access*, sino en las herramientas administrativas del panel de control de *Windows*. Allí se encuentra una aplicación para establecer los orígenes de datos (ODBC), como ilustra la figura 86(a), que al ser invocada ejecuta una ventana como la mostrada en la figura 86(b).

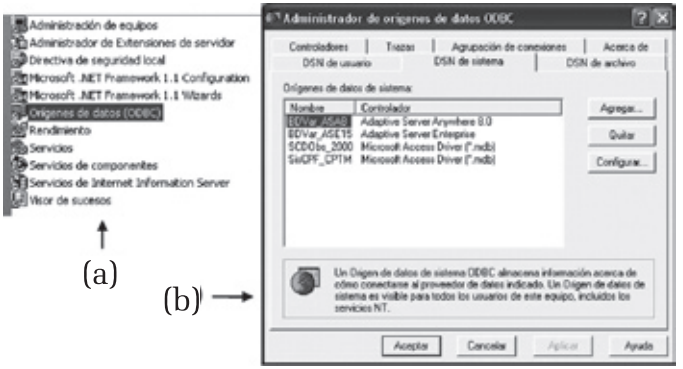


Figura 86. Orígenes de datos ODBC

Dando clic sobre el botón agregar de la ventana en la figura 86(b) se obtiene la lista del conjunto de manejadores (*drivers*, en inglés) preestablecidos para todos los SMBD que se encuentren instalados en el sistema y que soportan la tecnología ODBC. "Microsoft Access Driver (*.mdb)" es uno de ellos, y precisamente el que debemos seleccionar, como sugiere la figura 87.

Al finalizar el proceso de la figura 87 aparece otra ventana en la que se configuran los detalles de la base de datos *Access* a incorporar como nuevo registro DSN del sistema. Las características para el caso de la base de datos "MisLibros.mdb" se muestran en la figura 88.

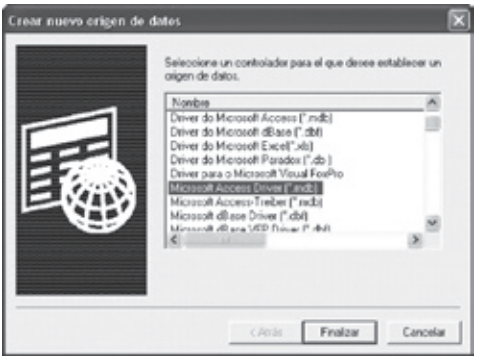


Figura 87. Selección del ODBC de Access

La sección (a) de la figura 88 contiene la ventana básica donde establecer las características del manejador ODBC para Access. El nombre escogido para identificar el registro DSN del sistema es "MisLibros". Su descripción es "Sistema de Registro de Libros". La base de datos a la cual se referirá es, obviamente, MisLibros.mdb e incluye el camino completo que debe hacer el sistema para ubicarla en el disco duro (esta operación puede hacerse utilizando el botón "Seleccionar…").

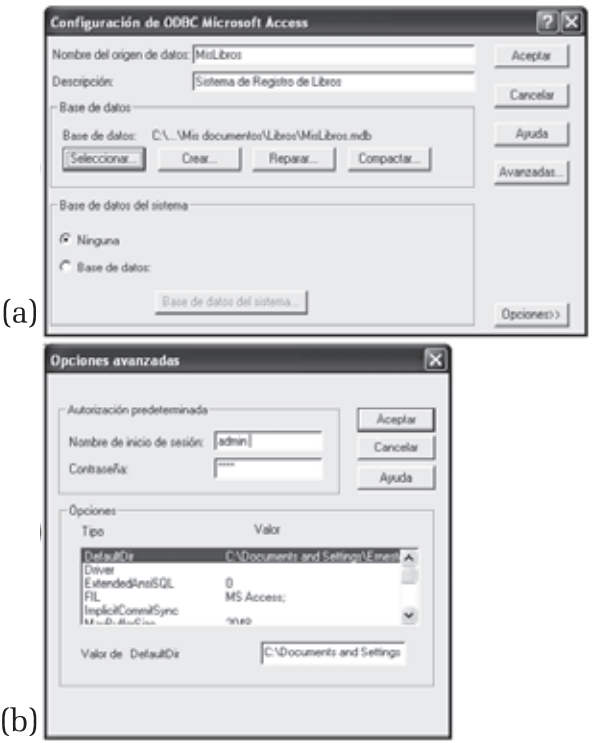


Figura 88. Configuración ODBC de una BD Access

La sección (b) de la figura 88 contiene opciones de configuración avanzadas que se despliegan presionando el botón "Avanzadas…" en la ventana de la sección (a). Allí nos ocupamos de establecer al usuario como "admin" (nombre del inicio de sesión) y la contraseña como "1234" (en ambos casos sin comillas). Al aceptar sobre ambas ventanas se crea el nuevo DSN del sistema, que deberá aparecer en una ventana como la mostrada previamente en la figura 86.

3.9.3 Diseño y creación de un sitio Web utilizando Front Page

En lo sucesivo trabajaremos sobre *FrontPage*. Para comenzar debemos crear un espacio donde residirá nuestro sitio Web. Desde *FrontPage* seleccionamos las opciones Archivo → Nuevo → Página o Web… Se despliega entonces una ventana de navegación en la que optamos por "Sitio Web vacío" (ver figura 89).

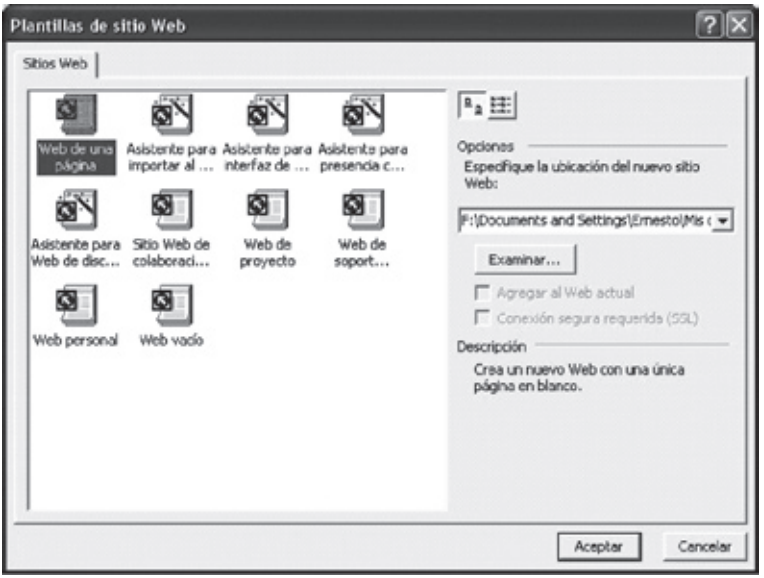


Figura 89. Carpeta donde reside un sitio Web, desde FrontPage.

Como plantilla seleccionamos "Web de una página" y escogemos la carpeta en la que deseamos crear el sitio valiéndonos del botón "Examinar…". Supongamos haber escogido una carpeta llamada "MisLibros" dentro de la carpeta "Mis documentos".

Al aceptar la localización del nuevo sitio Web y considerando que se compone de una única página (index.htm), *FrontPage* crea todos los elementos necesarios para comenzar a incorporar información dinámica en él. Para verificar los elementos que fueron creados dentro de la carpeta seleccionada puede utilizarse el Explorador de *Windows* o bien habilitar en *FrontPage* las opciones Ver → Lista de carpetas. El árbol a partir del directorio de trabajo luce tal como reza la figura 90.

Figura 90. Árbol de carpetas para un nuevo sitio Web, desde FrontPage

La página Web llamada "index.htm" es considerada generalmente como la inicial o de partida de todo sitio Web. Para establecer la estructura del sitio la abrimos con un doble clic desde el árbol de carpetas en *FrontPage* y escribimos sobre ella de manera que, para el ejemplo que nos hemos propuesto desarrollar, vista en el navegador de Internet luzca como la figura 91. Para desplegar dicha página, luego de alterar su contenido debe guardarse en *FrontPage* como se acostumbra. A continuación, con el navegador de que dispongamos (para nosotros *Internet Explorer*, también de *Microsoft*) se ubica la dirección electrónica "http://<Dominio del Servidor>/MisLibros/" (en este caso http://localhost/MisLibros/).

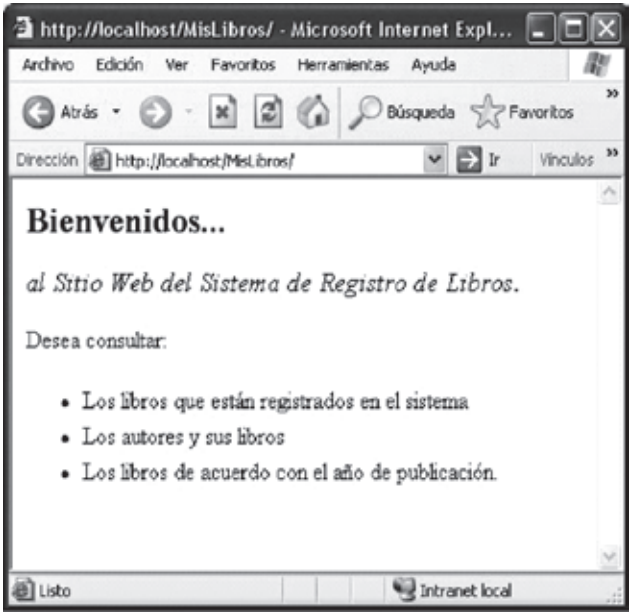


Figura 91. Versión inicial de la página principal del sitio Web de Libros

De vuelta a *FrontPage*, antes de poder acceder a la base de datos es necesario decirle de cuál DSN del sistema deberá valerse para obtener los datos que serán publicados. Para ello, la selección de opciones es Herramientas → Configuración del Web. Al hacerla se despliega la ventana de la figura 92(a), inicialmente en blanco.

Al dar clic en el botón "Agregar…" de la figura 92(a) se despliega la ventana que se esquematiza en la figura 92(b). Allí debe seleccionarse la opción "Origen de datos del sistema en el servidor Web", presionar luego el botón "Examinar…" y, en la ventana que se desplegará, escoger "MisLibros". Una vez escogido el origen de datos, lo que puede verificarse en el cuadro de texto que contiene la cadena "MisLibros" en la figura, se acepta la selección. Esto produce el cierre de la ventana mostrada en (b) y el retorno a la ventana mostrada en (a), ahora con "MisLibros" en el área blanca. Si todo ha salido bien, se habilitan los botones "Modificar…", "Quitar" y "Comprobar". Utilizando este último, debe verificarse que la conexión con la base de datos se ha establecido correctamente, en cuyo caso aparece ahora la frase "MisLibros" precedida por una marca de chequeo, tal como luce en la figura 92(a). Para abandonar esta facilidad se da clic sobre el botón "Aceptar".

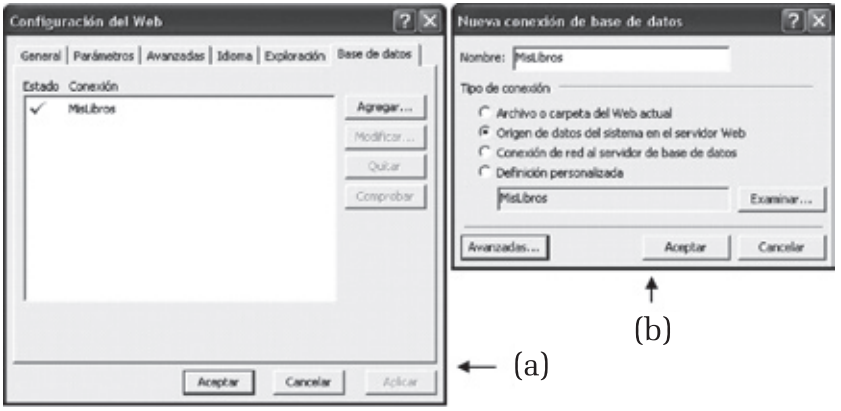


Figura 92. Añadido de una conexión DSN al sitio Web de Libros

Por otro lado, la mejor forma de publicar información en la Web utilizando *Access* es crear las consultas (o vistas) que se llevarán a la Internet, dentro del SMBD, y utilizarlas luego, una vez establecida la conexión ODBC, para extraer la información deseada.

La primera de las opciones que deseamos dar al usuario, ver "Los libros que están registrados en el sistema", se resuelve de manera muy simple con una consulta como las que hemos hecho en anteriores oportunidades. Se toma la información de los libros y se agrega la descripción de los temas. Esta consulta se llama en *Access* "www_Libros_Sistema" y se describe en el primer renglón de la tabla 13. La segunda opción para el usuario, "Los autores y sus libros", tal como la hemos concebido requiere en realidad dos consultas diferentes para completarse. En efecto, cuando el usuario dé clic sobre el hipervínculo correspondiente en la página principal, se ejecutará una consulta llamada en *Access* "www_Autores_Sistema" que lista todos los autores que contiene la base de datos, ordenados alfabéticamente por apellidos y nombres, y cambiando "Id_Autor" del tipo numérico al tipo cadena, para facilidad posterior del *FrontPage*. Cada registro de esta lista deberá contener a su vez otro hipervínculo que conduzca a la lista de los libros que corresponden al autor sobre cuyo identificador se dio clic. Esta segunda consulta, cuyos registros el sitio Web hará coincidir con los de la primera, se llama en *Access* "www_Autores_Libros". En resumen se trata de una consulta Web en dos pasos, primero los autores y, del autor seleccionado, sus libros. Las consultas requeridas en *Access* se listan en los renglones segundo y tercero de la tabla 13.

Tabla 13. Consultas a crear en Access para el sitio Web de Libros

Nombre	Sintaxis SQL del *Access*
www_Libros_Sistema	SELECT Libros.Id_Libro, Libros.Título, Temas.Tema, Libros.País, Libros.Ciudad, Libros.Año, Libros.Precio, Libros.Fecha_Adquisición FROM Temas INNER JOIN Libros ON Temas.Id_Tema = Libros.Id_Tema;
www_Autores_Sistema	SELECT Str([Id_Autor]) AS Id, [Apellidos] & ", " & [Nombres] AS Autor FROM Autores ORDER BY [Apellidos] & ", " & [Nombres];
www_Autores_Libros	SELECT Str([Id_Autor]) AS Id, [Autores de Libros].Id_Libro, Libros.Título, [Autores de Libros].[Orden de Autoría] FROM Libros INNER JOIN [Autores de Libros] ON Libros.Id_Libro = [Autores de Libros].Id_Libro ORDER BY Str([Id_Autor]), Libros.Título;

La tercera de las opciones que daremos al usuario de nuestro sitio Web, consultar "Los libros de acuerdo con el año de publicación", no necesita en realidad otra consulta en *Access*. Utilizaremos para ello la misma consulta "www_Libros_Sistema" que muestra la tabla 13 en su primer renglón. La diferencia estribará en que los registros devueltos serán filtrados, de acuerdo con un año que introduzca el usuario, en un formulario apropiadamente dispuesto en el sitio Web. Veamos a continuación cómo se implementan en *FrontPage* las facilidades descritas.

La primera opción (consultar "Los libros que están registrados en el sistema") necesita la creación de una nueva página Web. Para ello debe darse clic sobre el icono "Crea una página normal nueva" en la barra del *FrontPage* [figura 93(a)].

(a) (b) (c)

Figura 93. Iconos de la barra de herramientas del FrontPage

En la nueva página que ha sido creada y desplegada en el área de trabajo escribimos el título que deseamos darle, por ejemplo, "Consulta de los libros del sistema". Disponiendo el cursor en la línea siguiente invocamos desde el menú principal de *FrontPage* las opciones Insertar → Base de datos → Resultados…, a lo cual el aplicativo responde con la ejecución de un asistente que nos ayudará a incorporar al sitio Web una tabla contentiva de los resultados de una consulta a la base de datos. Los pasos que deben darse en el asistente se encuentran secuenciados en la figura 94.

El primer paso es la selección del origen DSN del que se extraerán los datos, en este caso "MisLibros". El segundo paso es la selección de la tabla o consulta que será objeto de publicación, en este caso se trata de la vista "www_Libros_Sistema". El tercer paso es seleccionar los campos de la vista que serán mostrados en la página Web, así como establecer criterios, si los hay, u ordenamiento si se desea (estos dos últimos al presionar el botón "Mas opciones…"). En este caso no es necesario alterar lo sugerido por el asistente. El cuarto paso se destina a dar forma a la tabla de la página Web, y el quinto a establecer el número

de registros que serán mostrados en cada página. Aceptamos también las alternativas que propone el asistente.

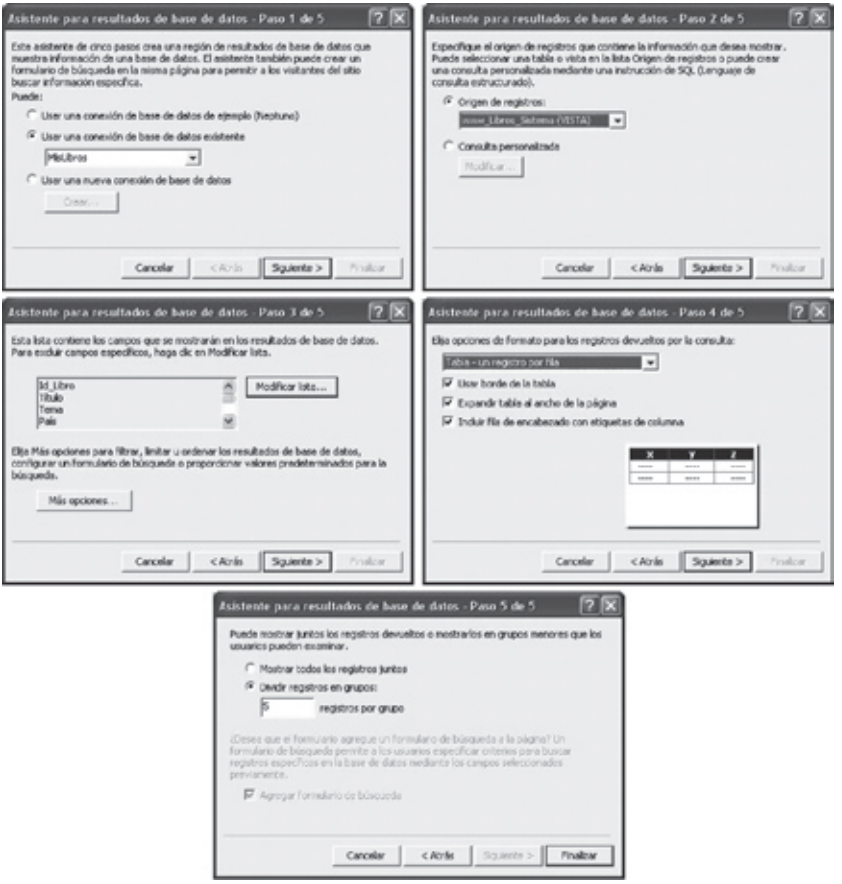


Figura 94. Pasos para incorporar un resultado de la base de datos al sitio Web

Al completar los pasos de la figura 94, *FrontPage* debe insertar automáticamente un objeto Web que denomina "Región de resultados de base de datos". Este objeto encapsula todas las características que hemos decidido utilizando el asistente.

En la línea siguiente a esta región de resultados podemos incluir una frase que, hipervinculada, nos devuelva a la página inicial (para facilitar la navegación al usuario). Esta frase puede ser algo como "Volver

al inicio". Una vez escrita la marcamos por completo y presionamos el botón "Insertar hipervínculo" [ver figura 93(b)], lo que nos conducirá a la ventana de inserción del hipervínculo que se muestra en la figura 95.

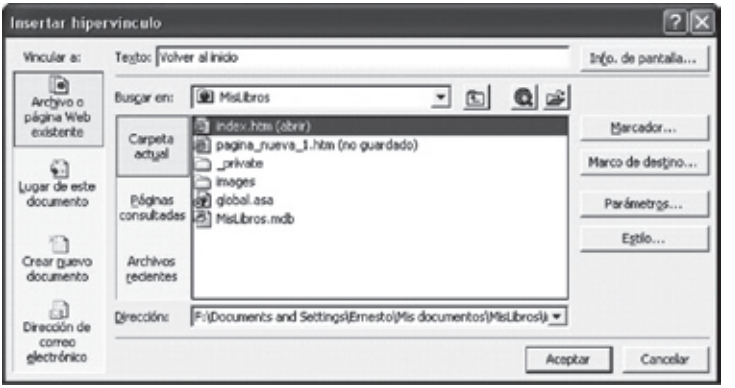


Figura 95. Inserción de un hipervínculo en FrontPage

De entre todos los objetos que pueden ser el destino de un hipervínculo se selecciona la página "index.htm" que representa el inicio del sitio Web. Al hacerlo, el sistema captura la dirección completa de la página en el disco y la ubica en el cuadro de texto "Dirección:". Sólo queda solo aceptar el hipervínculo como sugiere la figura 95.

En este momento, el área de trabajo *FrontPage* debe lucir de forma similar a la figura 96. Nótese que aún el nombre de la nueva página que hemos creado es aquel asignado por *FrontPage*. Esto se debe a que todavía no hemos guardado nuestro trabajo. Para hacerlo, basta dar clic en el icono "Guardar" de la barra de herramientas [ver figura 93(c)]. Al guardar, en vista de que el sistema sabe que esta página contiene elementos de base de datos, se sugiere la extensión "ASP" para el archivo. Efectivamente, una página que contiene códigos para la conexión a bases de datos utilizando tecnologías de *Microsoft*, debe ser reconocida como del tipo *Active Server Pages* para que el servidor de páginas activas, inmerso en el IIS, la reconozca apropiadamente. Cambiemos allí el nombre sugerido por "Libros.asp".

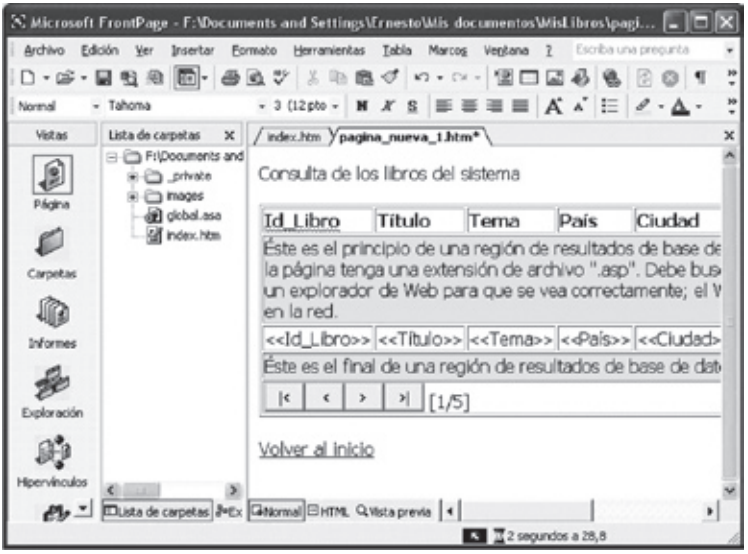


Figura 96. Área de trabajo del FrontPage

Una vez terminada la primera página que muestra resultados de "MisLibros" falta vincularla desde la página principal del sitio Web. Ubicamos entonces en el área de trabajo *FrontPage* la página "index.htm" y, señalando por completo la opción "Los libros que están registrados en el sistema", incorporamos como hipervínculo una referencia a la página "Libros.asp".

Para probar nuestra primera página, habiendo guardado los últimos cambios se accede a "index.htm" del sitio "MisLibros" con el navegador de Internet. La página principal del sistema debe parecerse a la figura 91, ahora contando en la primera línea de las opciones con un hipervínculo que conducirá a la consulta de libros (esto es, a la página "Libros.asp"). La respuesta a la escogencia de este hipervínculo se muestra en la figura 97.

Nótese en la figura 97 cómo el navegador ha resuelto exitosamente la conexión a nuestra base de datos y, en consecuencia, muestra una tabla con todos los registros de libros que están en ella. Son 101 páginas entre las que el usuario puede navegar con los botones de avance y retroceso dispuestos al final de la tabla. También aparece al término del área un hipervínculo para retornar a la página principal del sitio Web.

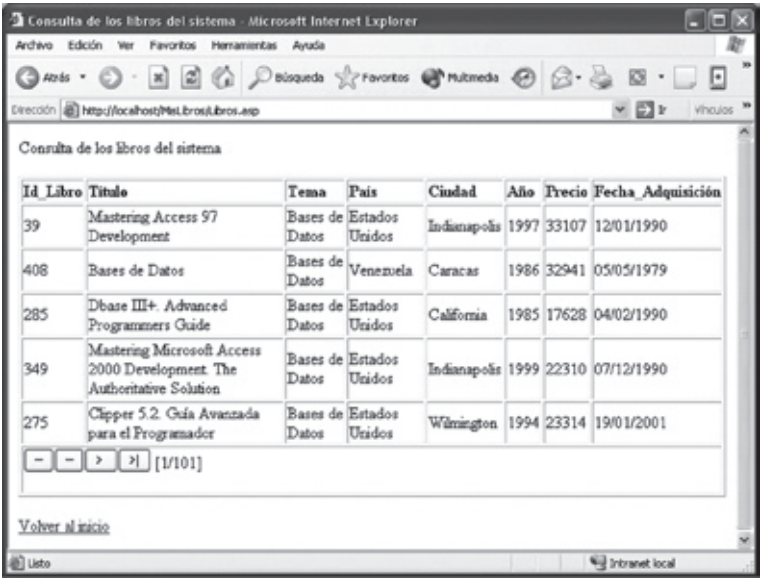


Figura 97. Vista en el navegador de la página "Libros.asp"

La segunda opción (consultar "Los autores y sus libros") amerita la creación de dos nuevas páginas Web. La primera de ellas contendrá los autores y se construye de la manera que hemos explicado con antelación, pero ahora, en lugar de invocar la consulta *Access* "www_Libros_Sistema" se debe invocar "www_Autores_Sistema". Supongámosla ya creada, denominada "Autores.asp" y vinculada a la principal desde la segunda opción.

Lo nuevo consiste en lograr que de la tabla resultante de autores se desprendan hipervínculos, por ejemplo, en sus identificadores, de manera tal que dando clic en alguno de ellos se pueda acceder a los libros que corresponden a dicho autor. Para lograr esto debe ser creada una nueva página que contendrá información de los libros discriminada por autor, basada en la consulta *Access* "www_Autores_Libros", y que será el destino del hipervínculo dentro de la tabla de autores.

Procedemos a crear esta nueva página que llamaremos "Autores_Libros.asp", escribiendo su título e insertando nuevamente un área de resultados de base de datos. En este caso debemos desviarnos un tanto de las opciones normales del asistente para la inserción de resultados, pues deseamos que los resultados de los libros no sean todos los posi-

bles, sino sólo aquellos que se corresponden con el autor seleccionado en la página que le precede. Debemos entonces establecer un criterio de igualdad entre el campo de los resultados ("Id") de la consulta ("www_Autores_Libros") y el parámetro que esperamos del hipervínculo (denominado también "[Id]"). La figura 98 muestra las ventanas requeridas del asistente.

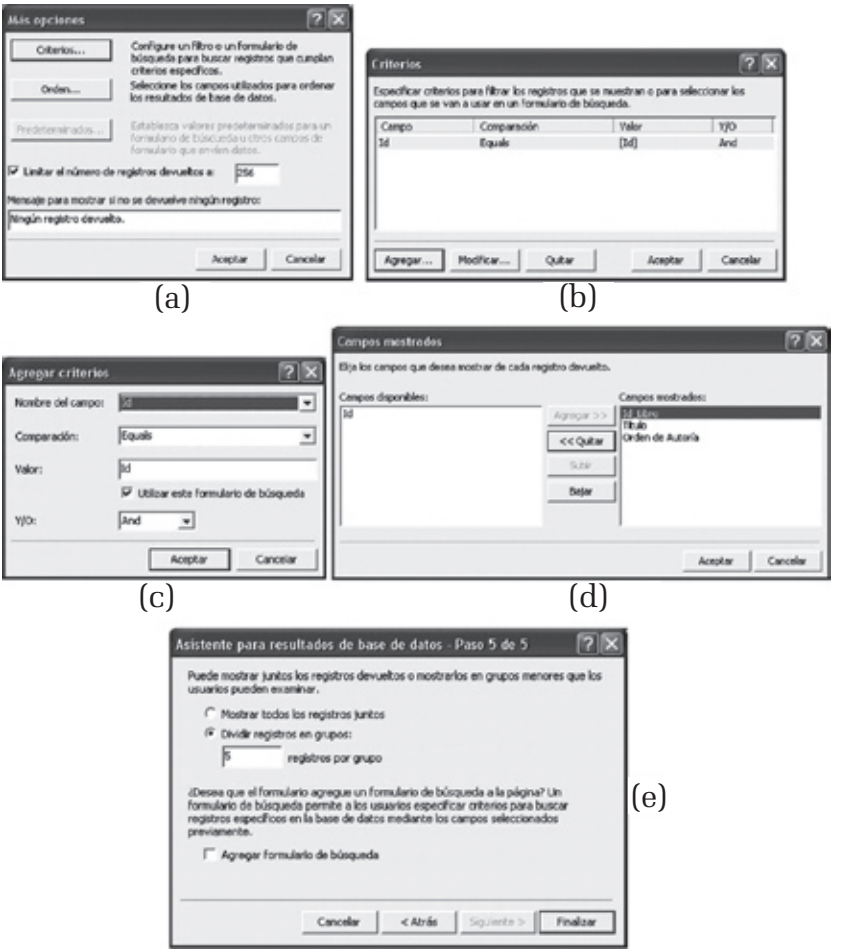


Figura 98. Variaciones del asistente para la inclusión de resultados de base de datos cuando éstos deben ser mostrados a partir de criterios parametrizados

En el paso Nº 3 del asistente, ilustrado con la figura 94, dando clic al botón "Mas opciones…" se obtiene la ventana mostrada en la figura 98(a). Allí, al pulsar el botón "Criterios…" se abre una ventana en blanco como la de la figura 98(b). En la ventana "Criterios" debe pulsarse el botón "Agregar…", llegando así a la ventana esquematizada en la figura 98(c). En esta ventana puede aceptarse lo propuesto por defecto para el criterio, esto es, igualdad entre campo y parámetro "Id", utilizando formulario de búsqueda para el operador lógico "And". Al aceptar estas opciones se incorpora un criterio como muestra la ventana de la figura 98(b).

Por otro lado, en el mismo paso Nº 3 de la figura 94, al dar clic sobre el botón "Modificar lista…" pueden seleccionarse los campos de la fuente de datos que deberán mostrarse. En este caso obviamos mostrar el identificador del autor, pues el usuario ya sabrá de qué autor se trata dado que él mismo lo seleccionará para ver sus libros. Esta acción se ilustra en la figura 98(d). Para finalizar, suponiendo que no deseamos ver en nuestra página un formulario de búsqueda, deshabilitamos la opción "Agregar formulario de búsqueda", como ilustra la figura 98(e).

Completando estos pasos adicionales y culminando el asistente para resultados de base de datos, ahora nuestra página tiene una nueva región de resultados que espera recibir un parámetro "[Id]", y que mostrará todos los libros correspondientes al autor cuyo identificador sea igual a este parámetro.

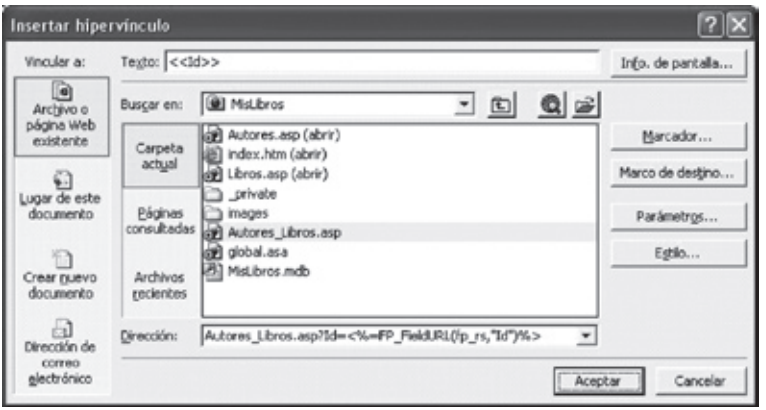


Figura 99. Incorporación de un hipervínculo pasando un parámetro

Debemos hacer algo más sobre la página original "Autores.asp". Visto nuestro diseño de las consultas, es necesario incorporar un hipervínculo al campo "Id" de la tabla de autores. Para ello lo señalamos en su totalidad y agregamos el mencionado hipervínculo de la forma acostumbrada. La página que deberá abrirse siguiendo este hipervínculo será "Autores_Libros.asp", pero esta vez con un parámetro, como sugiere la figura 99.

Los parámetros se establecen de manera muy sencilla desde el botón homónimo en la ventana de la figura 99. La figura 100 muestra las ventanas ya llenas, que permiten configurar el parámetro que se desea para el hipervínculo.

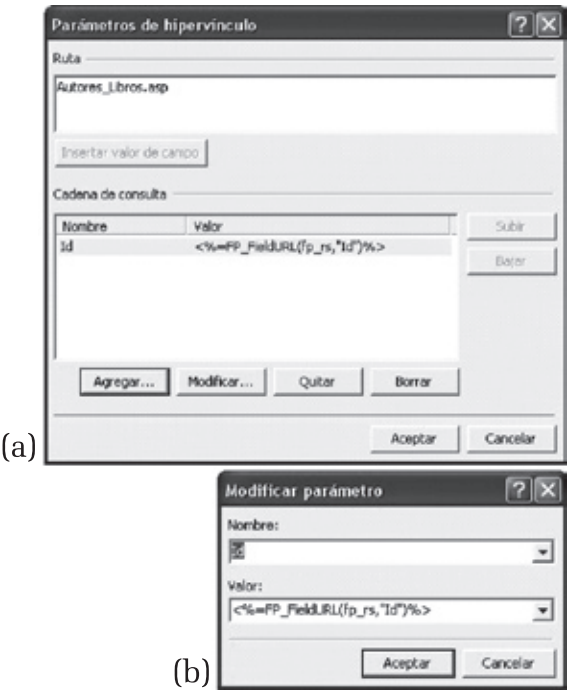


Figura 100. Agregando un parámetro al hipervínculo

La parte (a) de la figura 100 permite agregar el parámetro con el botón correspondiente. Al agregar se despliega la ventana que se ilustra en la parte (b) de la figura. Allí debe escogerse el campo "Id" en el cuadro combinado "Nombre", dejando que *FrontPage* establezca el código

HTML apropiado en el campo "Valor". Aceptadas ambas ventanas se ha completado el añadido de un parámetro y debe entonces aceptarse la creación del hipervínculo, el cual luce un tanto más complejo de lo normal, como puede apreciarse en la figura 99 en el cuadro combinado "Dirección:".

Este trabajo, necesario para dar el comportamiento que deseamos a la consulta de autores, debe lucir entonces como muestra la figura 101.

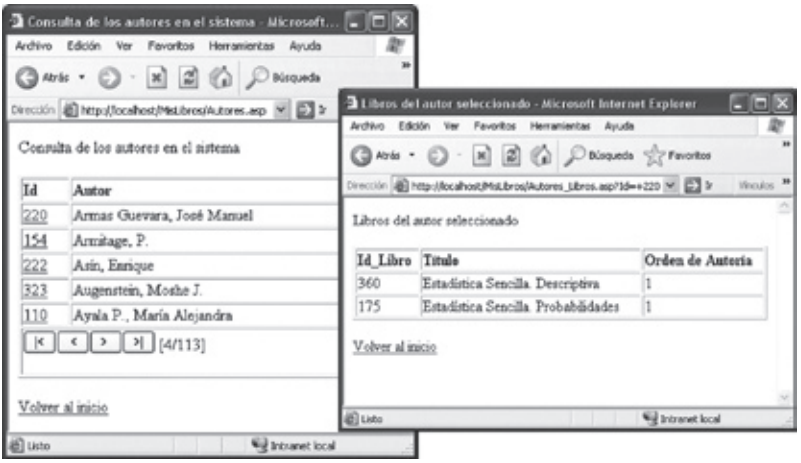


Figura 101. Consulta de autores y, dado el autor, sus libros

La tercera opción (consultar "Los libros de acuerdo con el año de publicación") requiere también de dos páginas Web. La primera de ellas para proporcionar al usuario una forma de seleccionar el año para el que desea ver los libros, y la segunda para responder la consulta, filtrando de acuerdo con el año que él haya suministrado.

La primera página Web es de tipo normal, esto es, HTML, con un formulario construido desde *FrontPage* para recoger el valor buscado. Para lograrla se crea una nueva página y se le nombra como "Pide_Valor.htm" titulándola "Formulario de Solicitud".

Estando abierta en el área de trabajo se inserta un nuevo formulario desde el menú principal del *FrontPage* como: Insertar → Formulario → Formulario. Esto produce la creación de un área (de formulario) que incluye los botones "Enviar" y "Restablecer" necesarios para procesar la información que se introduzca en él. Dejando los botones en la ter-

cera línea del área del formulario escribimos en su primera línea un mensaje indicativo de lo que el usuario debe hacer, en este caso algo como "Introduzca el año para el que desea ver los libros:". En la segunda línea incorporamos un objeto "cuadro de texto" haciendo Insertar → Formulario → Cuadro de texto. A continuación del formulario, como hemos acostumbrado, disponemos un hipervínculo para regresar a la página inicial del sitio. La nueva página debe entonces lucir en *FrontPage* como sugiere la figura 102.

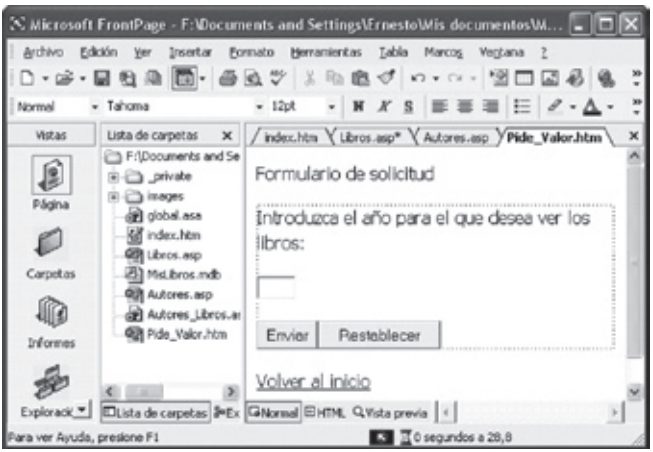


Figura 102. Diseño de una página Web con formulario para solicitar

Cuando damos doble clic sobre el cuadro de texto que contendrá el año, accedemos a sus propiedades. Allí cambiamos su nombre por "ValorA" y establecemos su tamaño a 4 caracteres, como muestra la figura 103.

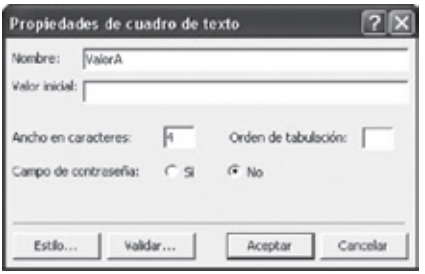


Figura 103. Propiedades de un cuadro de texto de formulario en FrontPage

Dejemos por un momento el formulario y discutamos ahora la creación de la página de resultados necesaria para mostrar los libros por año. Llamaremos a dicha página "Libros_Valor.asp". Esta página de resultados contendrá, como hasta ahora, un título, digamos "Consulta de libros por año", un área de resultados de base de datos y un hipervínculo que la retornará al formulario desde donde fue llamada. Todo esto se ejecuta como hemos explicado antes, el único cambio en este caso es que el área de resultados debe esperar un parámetro llamado "ValorA", que será igualado con el campo "Año" de la consulta (como hiciéramos para los autores de libros). Así pues, en el asistente para resultados de base de datos seleccionamos ahora la consulta "www_Libros_Sistemas", pero en el paso Nº 3 de la figura 94 agregamos un criterio, como indica la figura 104.

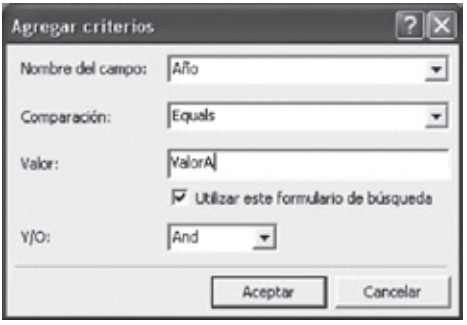


Figura 104. Criterio para los libros dando el año como parámetro

Volviendo al formulario, es necesario que indiquemos al sistema qué debe hacer cuando el usuario presione el botón "Enviar" (ver figura 102). Estando en el formulario de la página, con el botón derecho del ratón accedemos al menú secundario del objeto. Allí seleccionamos la opción "Propiedades del Formulario…", con lo que desplegamos la ventana de la figura 105(a).

En la ventana de la figura 105(a) se selecciona la opción "Enviar a otra:". Pulsando entonces en el botón "Opciones…" se despliega la ventana de la figura 105(b). Allí se escribe el nombre de la página Web que recibirá la información, en este caso "Libros_Valor.asp", que contiene resultados de BD parametrizados en torno al "ValorA".

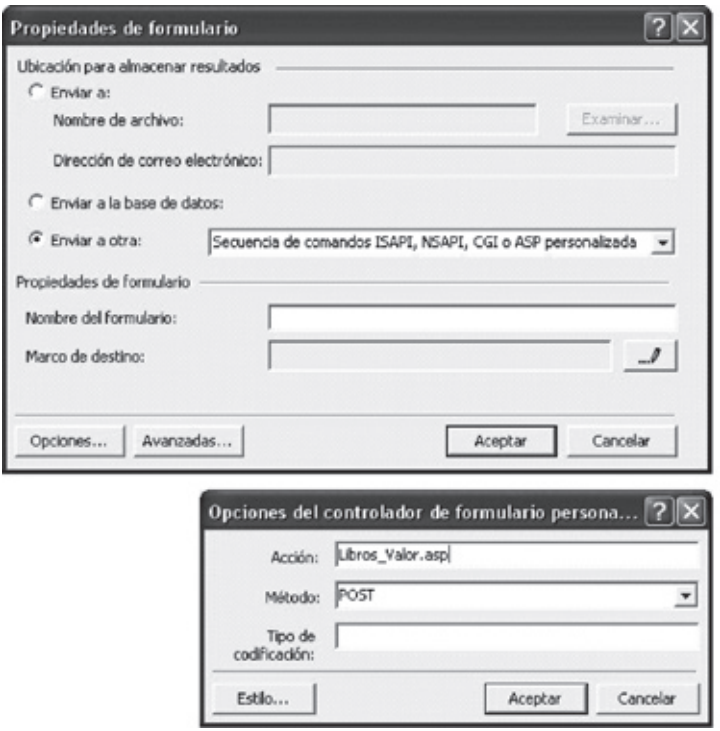


Figura 105. Acciones al enviar la información recabada con un formulario (ValorA)

El resultado es un despliegue como en la figura 106. Falta sólo incorporar los hipervínculos a las páginas apropiadas en la página principal "index.htm" y habremos construido un sencillo sitio Web dinámico para nuestro Sistema de Registro de Libros.

FrontPage es un aplicativo muy potente, que en realidad es un generador automático de código *Visual Basic* y ASP, código HTML y código SQL, para intercambio de datos entre bases de datos y la Internet; por consiguiente, las páginas que hemos construido con los asistentes, en realidad son páginas Web convencionales que incluyen líneas de códigos en todos estos lenguajes. Para una referencia completa de la sintaxis de todas ellas se reproducen íntegramente en el Apéndice D, denominado Sitio Web del Sistema de Registro de Libros (en CD).

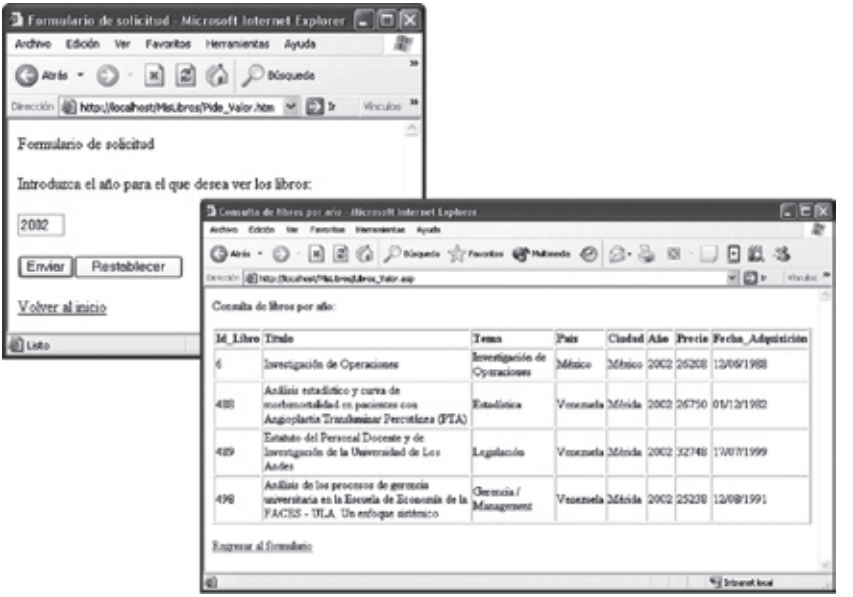


Figura 106. Consulta de libros por año en el Sitio Web de Libros

Epílogo

Hemos cubierto en un nivel introductorio los principales temas que atañen al desarrollo de sistemas de información y bases de datos. De manera considerablemente sintética rozamos desde las ideas generales de sistemas pasando por los aspectos particulares de sistemas de información y los fundamentos de los modelos de datos Entidad – Asociación y Relacional, hasta cómo se llevan a la práctica tales ideas en un Sistema Manejador de Bases de Datos concreto como *Microsoft Access*.

Habrá notado el lector que no se trata de un texto sencillo o trivial. La obra se apoya (sobre todo en la última parte) en el supuesto de que quien sigue con atención sus líneas lo hace acompañado de una versión del programa *Microsoft Access* y complementa muchos de los pasos sugeridos con su propia experiencia en el uso de la aplicación. Es claro que para ello no sólo ha necesitado conocer en algo estos temas, sino también una buena dosis de curiosidad e inclinación natural hacia la disciplina.

Clasificamos el texto como "introductorio" en el sentido de que "introduce a la práctica" (de SI, BD y *Access*), pero no lo es tanto en cuanto a lo conceptual. No obstante, estamos convencidos de que el material que acaba de discutirse será de gran ayuda para aquellos que deseen aproximarse por primera vez a esta desafiante ciencia.

Por otra parte, recomendamos al lector profundizar cuando menos en tres líneas más, a saber: la línea metodológica y de modelado, la

programación avanzada de bases de datos y las nuevas tendencias en el desarrollo de sistemas multiusuarios.

En cuanto a la metodología y opciones para el modelado de procesos y sistemas, hay muchos avances. Uno de los más importantes es el modelado orientado a objetos conocido como UML (por *Unified Modeling Language*, en inglés, o Lenguaje de Modelado Unificado, en español). Mayores señas pueden encontrase en http://www.uml.org/.

En cuanto a la programación avanzada de bases de datos recomendamos al lector profundizar con Visual Basic para Aplicaciones, especialmente en la tecnología ADO (por *ActiveX Data Objects*, en inglés, u Objetos de Datos Activos) y sus derivados como ADODB y similares (mayores señas en http://msdn2.microsoft.com/). Con esta tecnología podrá lograr sistemas aún más profesionales que los que permiten las características estándar del *Access*. También conseguirá poner a su alcance el acceso a bases de datos desde otros lenguajes de programación como PHP o Java, además del Visual Basic (mayores señas de PHP en http://www.php.net/ y de Java en http://java.sun.com/).

En cuanto a las nuevas tendencias en sistemas multiusuarios, será muy útil profundizar en el desarrollo a dos capas (Cliente–Servidor) y a tres capas (Cliente–Servidor de Aplicaciones–Servidor de Bases de Datos). Nadie en particular suscribe estos avances, por lo cual no damos mayores señas, pero, especialmente el desarrollo a tres capas, ha venido recibiendo mucha atención de la comunidad informática en los últimos años, por lo cual no será difícil al lector encontrar algún material que le introduzca en el tema.

Adicionalmente, un último aspecto sobre el que necesitará ahondar el lector, particularmente si le corresponde enfrentar la práctica de SI en niveles de alta exigencia, es en el problema de la implementación. Resulta evidente que la realización de SI y BD no es tarea sencilla. Involucra a múltiples personas y equipos de trabajo. Todo ello, sin embargo, puede perderse por completo si la implementación es deficiente. Por consiguiente, se trata de una tarea delicada, principalmente humana, que debe ser llevada a cabo cuidadosa y planificadamente.

En organizaciones privadas con jerarquías de mando muy fuertes y donde las instrucciones de dueños y gerentes tienen mucho peso en la labor del personal, la implantación de un sistema de información tiene buen camino andado si cuenta con la buena voluntad de personas con autoridad. Por otro lado, en organizaciones públicas en las cuales por lo

general actúan grupos de poder y colectivos con intereses particulares no siempre confesables, la implantación de sistemas de información puede llegar a convertirse en un verdadero reto y ser una labor de alto riesgo.

Vencer en la implantación de un sistema de información no es sinónimo de alcanzar el éxito económico en el proyecto, tampoco significa entregar todos los productos terminados. *Vencer*, en este caso, es en realidad lograr que la organización, humana por naturaleza, acepte nuestro sistema en todas sus capacidades y lo incluya en su labor diaria. *Vencer* significa lograr que el sistema "penetre las venas de la organización", la cual se sentirá a sí misma "informatizada" y no dará nunca más marcha atrás.

Concluimos recordando al lector que el camino en el que se ha aventurado al recorrer estas líneas apenas comienza, y que el futuro es prometedor.

Ernesto Ponsot Balaguer
Universidad de Los Andes
Mayo 2008

Contenido

3 Capítulo

TÍTULOS DE ESTA COLECCIÓN

- *Marketing es servicio al cliente*
  Marlene Peñaloza

- *El Presupuesto Público*
  Fabricio Paredes

- *Sistema de costos por proceso. Teoría y práctica*
  María Stella Quintero

La presente edición de 500 ejemplares
se imprimió el mes de mayo de 2008
en Centro Editorial Litorama C.A.
www.litorama.com.ve / info@litorama.com.ve
Mérida-Venezuela

Información técnica:
Programas: Adobe Indesing CS2 y Adobe Illustrator CS2
Impresión: papel Saima antique y cartulina Sulfato
Fuentes tipográficas: Melior y Helvéticas