CAPÍTULO 10

RECURSOS DE **DATOS**

En el ámbito de una organización, toda la tecnología descrita en los tres capítulos anteriores sólo tiene sentido si permite lograr una mayor rentabilidad, dar mejores servicios a los clientes, hacer más eficientes los procesos de negocios, permitir nuevas operaciones, etcétera. Para ello, nos falta el último elemento esencial en un contexto organizacional y que se refiere al almacenamiento y recupero de datos, y la obtención de información.

En este capítulo veremos las características generales del manejo de los recursos de datos y los conceptos de procesamiento transaccional e inteligencia de negocios, cómo las organizaciones manejan sus datos y obtienen información.

10.1 CONCEPTOS

El procesamiento transaccional, también denominado OLTP (del inglés *On Line Transactional Processing*), se refiere a lo vinculado al procesamiento de las operaciones corrientes. En el Capítulo 4 hemos visto sus principales características, vinculado con el nivel operativo. En cuanto a la administración, genera nuevos datos que deben ser organizados y almacenados (como, por ejemplo, nuevas facturas de venta), también modificaciones a datos existentes (como, por ejemplo, el cambio de datos legales y/o impositivos de un cliente) y, ocasionalmente, eliminaciones (como, por ejemplo, alguna operación errónea, aunque una buena práctica de control interno es que las anulaciones o eliminaciones o modificaciones de operaciones se realicen a través de nuevas transacciones como podría ser una nota de crédito).

Desde hace muchas décadas existen los sistemas de administración de bases de datos, pero mucho más recientemente se hicieron accesibles a las pequeñas y medianas empresas. Antes de su utilización, la organización y administración de datos se realizaba con lo que se pasó a denominar el contexto tradicional de archivos. En este caso, cada entidad (persona, objeto, transacción o evento) constituía un archivo lógico distinto (por

ejemplo, archivo lógico de clientes o de artículos o de movimientos de stock). El archivo tenía diferentes registros lógicos (en nuestro ejemplo cada registro tenía los datos de un cliente); todos los registros lógicos tenían normalmente el mismo diseño (datos y longitudes de cada uno de ellos) y todos los datos de cada registro lógico estaban juntos físicamente en el medio de almacenamiento. Frecuentemente varios registros lógicos formaban un registro físico que era la unidad que se manejaba para la lectura o grabación física del medio de almacenamiento, como ya hemos visto en el Capítulo 8. Cada registro lógico tenía un conjunto de datos denominados habitualmente campos (por ejemplo, nombre del cliente, domicilio legal, domicilio de entrega, teléfonos, etcétera). Cada dato o campo tenía una longitud en bytes determinada y si no había dato real, se dejaba en blanco o en ceros, pero ocupando lugar físico tanto en la memoria como en el medio de almacenamiento. Los datos podían ser de distinto tipo, pero básicamente había datos alfanuméricos conformados por letras y/o números y datos numéricos. Según la organización de archivo elegida, esos archivos lógicos podían ser utilizados de distintas formas y podían plantear distintas restricciones al procesamiento de las transacciones. Como veremos en el Punto 10.4, el uso de archivos tradicionales generaba distintos inconvenientes.

Los sistemas de administración de bases de datos (SABD) solucionaron casi totalmente los problemas que generaban los archivos tradicionales. Simplificadamente, una base de datos es físicamente un archivo (puede ser más de uno) que contiene muchos archivos lógicos y que se denominan tablas. Cada tabla tiene las características básicas de un archivo lógico tradicional, es decir, que está compuesto de registros lógicos y cada registro lógico tiene distintos campos o atributos. En el Punto 10.5 analizaremos todas sus características.

En el Capítulo 5 también se introdujo inteligencia de negocios (o BI, del inglés *Business Intelligence*). Es un conjunto de esquemas de procesamiento que veremos más detalladamente en el Punto 10.6. Se relaciona muy directamente con los niveles táctico y estratégico de una organización, aunque, en menor proporción, también se vincula con el nivel operativo. La característica básica de este tipo de procesamiento es el análisis de grandes cantidades de datos previamente almacenados, es decir, principalmente recuperación resumida y agrupada.

10.2 MODELADO DE DATOS

Los datos que manejan los sistemas de información, no sólo son importantes para llevar a cabo las operaciones cotidianas, sino también pensando en un procesamiento ulterior que permita a la organización obtener valiosa información para la toma de decisiones. Por ejemplo, para las operaciones corrientes podemos ignorar el tipo de cliente, el canal por donde se realizó una venta o como un nuevo cliente conoció a la empresa; sin embargo, seguramente esos datos pueden ser fundamentales para un análisis de marketing.

Cuando diseñamos un sistema de información es entonces primordial tener en cuenta los datos que debemos manejar y almacenar. No obstante, al empezar el análisis de los datos es difícil que podamos llegar al nivel de detalle final. A los efectos de ir refinando el análisis de datos a medida que se avanza en el diseño del sistema de información, recurrimos al modelado de datos que permite ir definiendo una representación que formaliza y que vincula la realidad con nuestro diseño.

Si consideramos el nivel de abstracción, podemos clasificar los modelos de datos en:

- *Modelo de datos conceptual:* un modelo conceptual (alto nivel) permite identificar, globalmente, las entidades de datos (por ejemplo clientes, proveedores, empleados, etcétera), sus principales campos o atributos, sus interrelaciones y las restricciones de integridad. Por su característica conceptual, no depende de la

implementación real, ya sea que se usen archivos tradicionales o bases de datos. Es el más cercano a como un usuario percibiría la estructura de datos. Es muy útil durante el análisis y diseño de un problema dado. Como veremos más detalladamente en el siguiente punto, la herramienta de uso más generalizada es el diagrama de entidad-relación (DER).

- *Modelo de datos lógico:* el modelo lógico (también denominado modelo de implementación) incorpora las operaciones. Si bien son entendibles por los usuarios finales, están relacionados con la implementación en algún sistema de administración de base de datos. No es dependiente de un SABD en particular, pero sí del modelo que utilice el SABD. Por ejemplo, podemos mencionar el modelo relacional, jerárquico, de red y orientado a objetos.
- *Modelo de datos físico:* se refiere a la implementación física de las estructuras de datos a bajo nivel, como se almacenan en el medio de almacenamiento y a los métodos de acceso a los datos, por lo que tiene una alta dependencia del SABD a utilizar.

En la definición y relación de los datos hay restricciones de integridad de los mismos. Las restricciones pueden ser de contenido o dominio (se definen los valores válidos que puede tomar el contenido de un atributo, por ejemplo, el dato sexo puede tomar valores M o F, indicando masculino o femenino), de restricciones entre campos (valores válidos pero en relación a otros campos, por ejemplo, que para aceptar la licencia por maternidad, el sexo debe ser F –femenino–) o de relaciones, también denominadas de integridad referencial (por ejemplo, no puede eliminarse un artículo de la tabla respectiva, si hay movimientos que referencian ese artículo en alguna otra tabla; estas restricciones se aplican u operan de determinada manera, cuando se hacen inserciones, modificaciones y eliminaciones de registros en una base de datos).

10.3 HERRAMIENTAS DE MODELADO DE DATOS

Dentro de las herramientas para trabajar el modelado de datos, el diagrama de entidad-relación (DER) es la de uso más difundido.

La **Figura 10.1** grafica las entidades representadas por cuadrados y las relaciones entre ellas, representadas por líneas. Dentro de los cuadrados, se identifican los atributos de la entidad, encabezados por el o los atributos que se denominan clave principal o primaria y que permiten diferenciar los distintos registros de la entidad, de tal manera que no haya una posible duplicación de clave principal entre dos o más registros de la entidad (tabla). Por ejemplo, si en la entidad empleado –que tiene como atributos el número de documento de identidad, el apellido y nombre, el domicilio, el salario, etcétera– utilizamos como clave principal el apellido y nombre, podría darse la posibilidad que dos personas tuvieran el mismo apellido y nombre; pero sería imposible que dos personas tuvieran el mismo número de documento de identidad, por lo que la clave principal debería ser este último atributo.

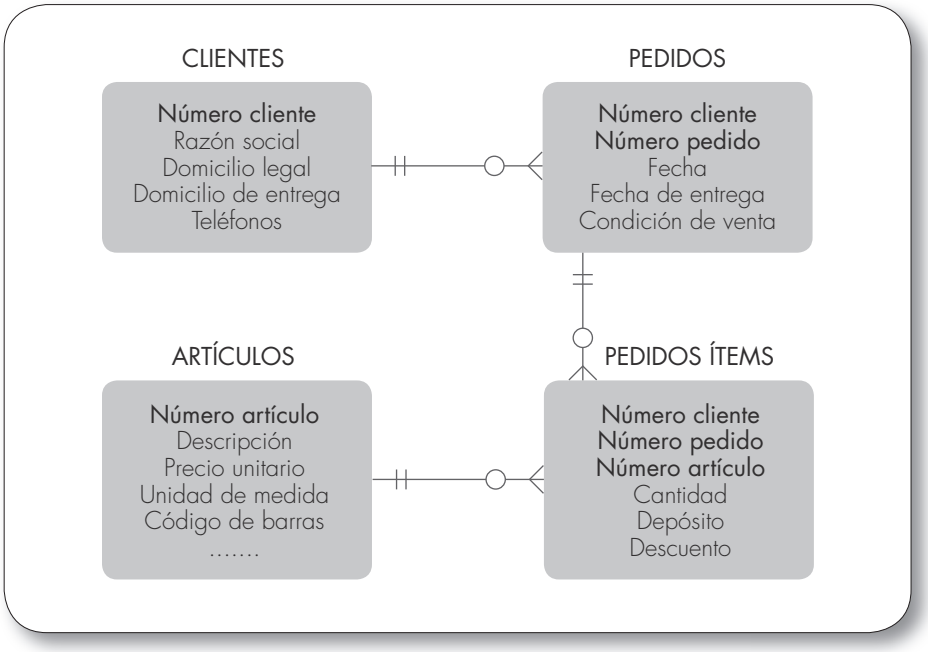
Terminando la línea en la entidad, unas marcas de cada lado de la línea de unión indican cómo es la relación entre estas dos entidades (cardinalidad). Por ejemplo, un cliente puede tener 0, 1 o más facturas o notas de venta. Un pedido tiene un cliente, pero puede tener 0,1 o varios artículos.

En el siguiente ejemplo, un cliente puede no tener ningún pedido (símbolo círculo) o tener uno o más de uno (símbolo pata de gallo), pero un pedido necesariamente debe tener un cliente y sólo uno. De igual manera, un artículo puede no tener ningún pedido o tener uno o más de uno (en distintos pedidos), pero un artículo pedido tiene un artículo. Finalmente, un pedido debe tener uno o más artículos. Una simple línea cruzada indica una relación uno a uno. Estos símbolos pueden combinarse para representar las

relaciones entre dos entidades. De cada lado se indica la mínima y la máxima cantidad de repeticiones que puede tener la relación del lado de la entidad respectiva, por lo que vamos a tener dos símbolos en cada extremo de la línea.

Figura 10.1

Ejemplo de diagrama de entidad-relación



En el libro de Ernesto Chinkes, *Modelado de Sistemas de Información*, puede verse este tema con más detalle y ejemplos.

10.4 ARCHIVOS TRADICIONALES

Antes de la aparición de los sistemas de administración de bases de datos (SABD o DBMS del inglés *Data Base Management System*), las características del manejo de datos eran muy distintas. Para diferenciarlo genéricamente se denominó método de archivos tradicionales.

Una diferencia sustancial era que cada archivo lógico contenía datos de una única entidad. Había archivos maestros de artículos, clientes, proveedores, empleados, cuentas contables, etcétera y archivos de movimientos o transacciones como los de cuentas corrientes, asientos, liquidaciones, etcétera. Pero cada uno tenía datos sólo de una de esas entidades. Un registro lógico contenía todos los datos de un cliente, por ejemplo. Para mejorar el rendimiento de la computadora, los registros lógicos se podían agrupar en registros físicos (normalmente un registro físico podía contener varios registros lógicos que se leían o grababan como una unidad).

Si bien aparecieron algunos utilitarios que permitían evitar el desarrollo de un programa para algunas funciones simples de manipulación o informe, el proceso de actualización y manejo de datos se realizaba con programas ad hoc. Además, los programas tenían que definir internamente todos los detalles del archivo lógico, como la longitud total del registro lógico, el diseño de cada uno (datos que componen cada registro lógico), cantidad de registros lógicos por registro físico, etcétera.

Si se deseaba modificar el diseño de un archivo lógico, necesariamente implicaba que había que modificar la parte pertinente de todos los programas que usaran ese archivo lógico, aunque sólo lo usara para leer algún dato (como, por ejemplo, el nombre del artículo).

Por este motivo, frecuentemente, distintos equipos de desarrollo definían sus "propios" archivos lógicos para la aplicación sobre la que trabajaban, sin considerar si ya había archivos lógicos definidos para una determinada entidad y, en consecuencia, podían encontrarse archivos de clientes distintos para una aplicación de ventas y para otra aplicación de garantías o servicio posventa. De igual manera, se definían distintos archivos de artículos para aplicaciones de facturación y producción. Lo peor del caso es que, generalmente, las codificaciones de clientes y artículos u otros datos como, por ejemplo, el código de presentación (unitario, por docena, etcétera) o el depósito, no coincidían en los distintos archivos, por lo que, además, era imposible consolidar datos referentes a un artículo o cliente, por ejemplo. En la **Figura 10.2** se puede ver gráficamente cuatro aplicaciones (por ejemplo, producción, ventas, compras y contabilidad), cada una con sus diferentes programas y con sus propios archivos; en el caso de las tres primeras aplicaciones, cada una con su propio archivo de artículos, por ejemplo.

Adicionalmente, si, por ejemplo, un cliente cambiaba su domicilio, se requería que se actualizaran más de un archivo; habitualmente uno quedaba actualizado y otro no.

No hay razones técnicas que implicaran que necesariamente debía ocurrir todo esto, pero las características del proceso de desarrollo de aquella época, la falta de uso de la reutilización, la forma de trabajo en equipos, los tiempos y herramientas de desarrollo, hacían que, en la práctica, todos estos problemas fueran moneda corriente.

Resumiendo, en un entorno tradicional de archivos, los principales problemas que aparecen son: a) la alta dependencia entre programas y datos (que dificulta desarrollo y mantenimiento), b) la poca flexibilidad resultante, c) la alta dificultad para compartir datos entre aplicaciones y su consecuente baja integración, d) la redundancia con la consiguiente alta posibilidad de inconsistencia de los datos redundantes y e) una escasa seguridad en el acceso, exhibición y administración de los datos.

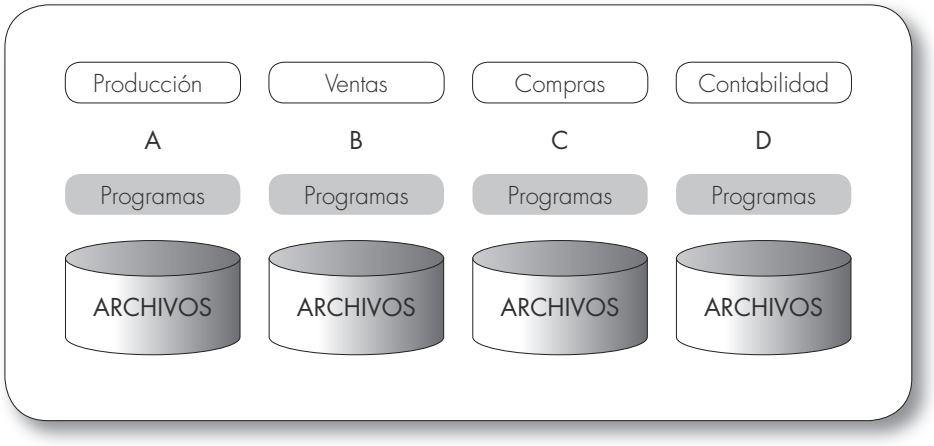


Figura 10.2
Entorno tradicional de archivos

10.4.1 Tipos

Según las características de la organización y del diseño del software, puede haber mucha cantidad de archivos lógicos con datos muy diversos. Sin embargo, casi la totalidad de los archivos pueden ser clasificados en archivos maestros o archivos de transacciones.

Un archivo maestro es una colección de datos referidos a una entidad, por ejemplo, en una empresa tendremos archivos maestros de artículos, clientes, vendedores, proveedores, cuentas contables, empleados, etcétera. Por ejemplo, el archivo de clientes tendrá por cada uno el número de cliente, el nombre, el domicilio legal, el domicilio de entregas, los números y datos impositivos, etcétera. Incluso podemos tener otros

archivos asociados; si conservamos las cuentas de correo electrónico a quien debemos enviar las comunicaciones generales, las facturas, los resúmenes de cuenta, etcétera, por un mejor diseño seguramente tendremos un archivo aparte con el número de cliente, el tipo de cuenta de correo y la cuenta de correo electrónico. Lo mismo puede ocurrir con las diferentes listas de precio de los artículos o con la composición de un producto terminado.

En cambio, los archivos de transacciones o movimientos conservan las distintas transacciones o movimientos que se refieren o realizan las distintas entidades. Por ejemplo, el archivo de cuenta corriente tendrá las facturas, notas de débito, notas de crédito y cobranzas o pagos de los clientes o proveedores; un archivo de movimientos de stock almacenará las entradas y salidas de cada artículo; otro guardará las liquidaciones de sueldo de cada empleado, o los asientos contables.

Por lo general, hay opciones del software que se enfocan al mantenimiento de los archivos maestros, que permiten agregar, modificar y eliminar registros y/o datos. Otras opciones posibilitan la realización de transacciones que son registradas en los archivos correspondientes, como, por ejemplo, facturas, órdenes de compra, órdenes de pago, etcétera. Por lo general, el resto de las opciones del software se referirán a la obtención de informes.

10.4.2 Métodos de organización y acceso

Cualquiera sea la forma en que se organicen los datos, existen dos formas de acceso.

El acceso secuencial implica que los datos se irán accediendo uno a continuación de otro, a los efectos de realizar el procesamiento. Por ejemplo, si queremos obtener un informe de los empleados de la organización, tendremos que ir accediendo a los datos de cada empleado secuencialmente hasta terminar de procesar a todos. Un tema a tener en cuenta es el orden en que se desea obtener el informe, por número de documento, por apellido y nombre, etcétera y si se desea incluir a todos los empleados o sólo a los de un sector o categoría, por ejemplo.

El acceso directo resulta indispensable para la mayoría de las transacciones. Si deseamos facturar un artículo a un cliente que se encuentra delante de la computadora, es difícil imaginar un procesamiento que vaya leyendo los artículos hasta llegar al deseado y luego, leyendo todos los clientes hasta encontrar los datos del indicado. Para ello, es necesario que se puedan obtener los datos necesarios en sólo unos pocos accesos al medio de almacenamiento (por ejemplo, disco magnético).

En el contexto tradicional de archivos existen distintas organizaciones de archivo utilizables para organizar el almacenamiento y recupero de los datos en los archivos lógicos. A su vez, existen variantes de algunas de ellas. Como no es propósito de esta publicación, ahondar en este tema, sólo veremos en forma genérica y global las organizaciones de archivo secuencial, secuencial indexada y directa (o relativa).

En la organización secuencial, los registros lógicos se encuentran uno a continuación del otro, ordenados de acuerdo a uno o más campos denominados clave, por ejemplo, número de cliente. Para llegar a un cliente, hay que pasar por todos los anteriores. Si se busca por número de cliente, sólo habrá que pasar por todos los de número menor, pero si buscamos por nombre, tendremos que pasar por todo el archivo, desde el primero, hasta encontrar el deseado, que podría ser el último. Como vemos, esta organización permite solamente el acceso secuencial. Si queremos agregar o eliminar un cliente, tendremos que generar un nuevo archivo lógico, eliminando o agregando el cliente, de acuerdo al ordenamiento del archivo.

En el caso de la organización secuencial indexada, además de los datos, en el archivo se almacenan índices. Los registros de datos también se guardan ordenados por la clave, pero un registro de índices indica que de tal a tal registro están en tal ubicación

y así sucesivamente. A medida que la cantidad de registros de datos aumenta, un sólo registro de índices no alcanza y se arma un segundo nivel de índices, manteniendo siempre un único registro inicial de índices, que ahora apunta al nivel inferior de índices que apunta a los registros de datos. Si resulta necesario por la cantidad de datos del archivo, se van generando varios niveles de índices. De esta manera, un registro determinado puede ser recuperado con uno o más accesos a índices y luego al registro de datos (por lo menos dos accesos). También, siguiendo el índice de menor nivel, se pueden leer todos los registros secuencialmente. La forma en que se organiza un archivo de este tipo permite fácilmente agregar registros dentro de la secuencia, con unos pocos accesos. Por permitir tanto el acceso secuencial como directo, es una organización muy utilizada y con diferentes variantes de implementación.

En un archivo con organización directa (o relativa), cada posible clave tiene un lugar reservado dentro del archivo. Por ejemplo, el cliente 1 tiene la posición 1, el cliente 10 la posición 10 y así sucesivamente. Si el cliente 2 no existe, la posición 2 está vacía. Para acceder a un cliente determinado, simplemente leemos el registro de igual número, en un sólo acceso directo. Si queremos leer todos los clientes secuencialmente, sencillamente leeremos los registros por número consecutivo, saltando las posiciones vacías. Es la organización de acceso directo más rápido (un sólo acceso), pero tiene varios inconvenientes: 1) la clave debe ser numérica y correlativa, y 2) la cantidad de registros activos debe ser alta en comparación con la cantidad de claves posibles, para no desperdiciar tanto espacio de almacenamiento; por ejemplo, sería inadmisible si tenemos 300 empleados en el archivo de nómina de personal y la clave es el número de documento que puede ser de 00000001 a 99999999 (300 claves activas sobre 99.999.999 posiciones posibles totales). Para solucionar este problema, hay una variante que permite generar una nueva clave numérica acotada por un algoritmo, pero que implica la aparición de sinónimos (claves originales distintas que dan una igual clave transformada) y que impide el acceso secuencial. Sin embargo, dadas las características de una clave original numérica correlativa, el uso de esta organización es muy limitado.

10.5 BASES DE DATOS

Una base de datos (BD o DB del inglés *Data Base*) es una colección de datos almacenados en un formato estandarizado. Físicamente es uno o más archivos pero cuya forma de organizar el contenido de datos no guarda ninguna relación con los esquemas de almacenamiento de los archivos tradicionales.

Un sistema de administración de bases de datos (SABD o DBMS) es un software que posibilita definir una base de datos, guardar los datos, permitir un lenguaje de consulta (SQL), generar informes y crear formularios o pantallas para ingresar datos.

Existen distintos tipos de bases de datos (modelos lógicos), las jerárquicas, de red, relacionales (y multidimensionales) y orientadas a objetos. Actualmente los modelos corrientemente utilizados son los relacionales y los orientados a objetos.

El modelo jerárquico fue el primero y se caracterizaba por implementar una relación de datos de uno a muchos (similar a la representación de relaciones jerárquicas); un nodo padre puede tener muchos nodos hijos, pero no al revés. Las relaciones entre registros se podían ver como un árbol invertido. Consecuentemente, el acceso a los datos también tiene ese sentido jerárquico.

El modelo de red permitió implementar relaciones de datos de muchos con muchos y, por lo tanto, el acceso a los datos puede tener varias trayectorias; un nodo hijo también puede tener varios nodos padres. Todas esas relaciones deben plasmarse en forma física (índices o apuntadores) y, por lo tanto, puede insumir gran cantidad de tiempo y espacio de almacenamiento.

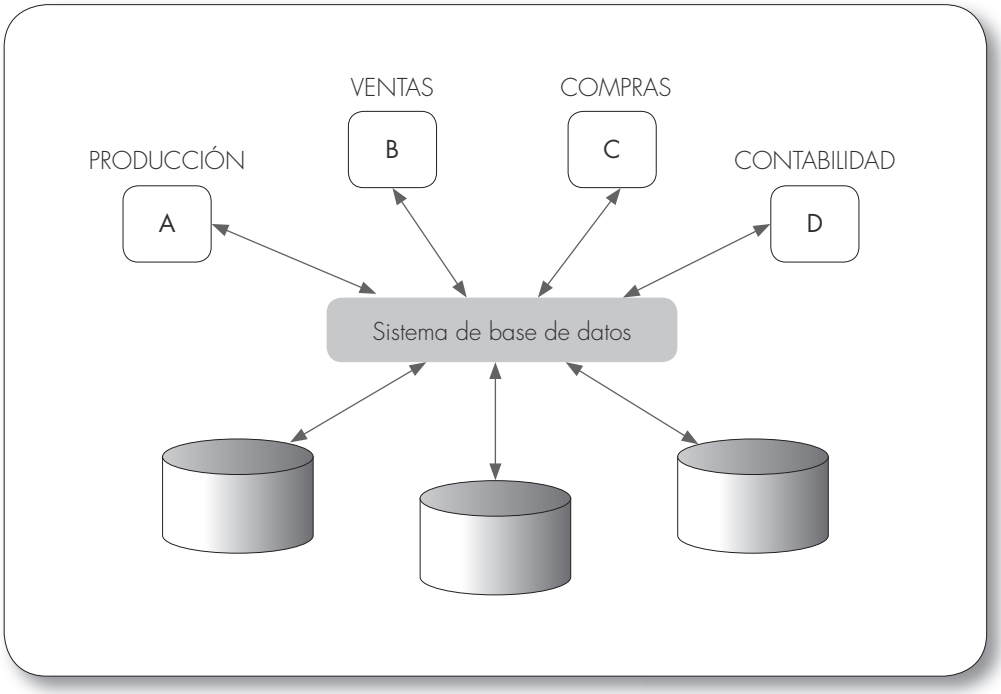
En la década de los años 70, Edgar F. Codd en su trabajo *Un modelo relacional de datos para grandes bancos de datos compartidos*, formuló las bases del modelo relacional. Se basó en el uso de "relaciones". Básicamente, se definen grupos de datos como tablas (o relaciones) y cada tabla guarda los atributos en columnas (campos). Las tablas no se conectan físicamente entre sí; las conexiones se establecen a través de datos comunes o coincidentes entre distintas tablas. El lugar y la forma en que se almacenan los datos no tiene relevancia como en los otros modelos. La flexibilidad y eficacia fueron las principales razones para que este modelo se impusiera rápidamente sobre los anteriores.

Una base de datos multidimensional se diferencia de una relacional básicamente en forma conceptual; el diseño y forma de uso de las relacionales es más eficiente para el procesamiento transaccional, mientras que las multidimensionales (formando cubos multidimensionales), como veremos en el siguiente punto, buscan eficiencia para el uso de herramientas de inteligencia de negocios.

El modelo orientado a objetos puede verse como una extensión del modelo relacional, donde se aplican las características de orientación a objetos. Con este modelo, se pueden definir operaciones, procedimientos o métodos sobre los datos, integrando la definición de la base de datos (por ejemplo, el código para agregar un registro nuevo a la tabla). Los programas de aplicación pueden invocar esos métodos (de acuerdo a la interfaz establecida en la definición del método).

En la **Figura 10.3** puede verse, en contraposición a la **Figura 10.2**, cómo las cuatro aplicaciones se relacionan con el SABD; sin "ver" los detalles físicos y lógicos de la administración de las tablas, registros lógicos y campos.

Figura 10.3
Entorno de base de datos



La forma de almacenamiento de los datos es muy diferente a los archivos tradicionales. En efecto, los distintos datos que componen un registro lógico (también denominado tupla), no se encuentran necesariamente juntos físicamente, sino que pueden estar guardados en distintos lugares del medio de almacenamiento (pero se encuentran vincula-

dos entre sí). Por esta característica, si debe agregarse un campo a una tabla, no resulta necesario modificar ningún programa, salvo que deba usar este nuevo dato. Ya sea en forma manual o a través de un programa, es muy sencillo agregar un campo nuevo al diseño de una tabla y no hay que hacer nada más que modificar sólo los programas que deba almacenar o recuperar ese nuevo dato. Además, se pueden almacenar otros tipos de datos como sonidos, imágenes y videos. Por otra parte, los datos que no están, no ocupan lugar y los que están, ocupan el lugar justo y necesario (por ejemplo, si el nombre de un cliente tiene 15 caracteres, sólo ocupará esos 15 bytes y no la longitud máxima definida, como sucede en los archivos tradicionales; si el domicilio de entrega es el mismo que el domicilio legal y en consecuencia se deja en blanco, el dato no está y no ocupa lugar).

De igual manera, si se agrega una nueva tabla en la base de datos, no resulta obligatorio ni necesario modificar los programas que utilicen las otras tablas de esa base de datos.

Los programas no necesitan leer todas las tablas de la base de datos, ni tenerlas internamente definidas o sus diseños. Y cuando leen una tabla, tampoco necesitan leer todos los campos, pudiendo indicar la lista de campos que se quieren leer o bien indicando que se lean todos los datos, pero sin tener que poner el nombre de cada campo. Cuando se graban los datos en una tabla, tampoco es necesario que se graben todos los campos de la tabla; si se graba una modificación, el resto de los datos permanecerá sin ninguna modificación y con los datos que tenía anteriormente; y si es una inserción en la tabla, los campos para los que no se indique contenido, serán valores nulos (ausencia de valor) o el valor que se haya establecido como valor predeterminado (para cada campo se puede indicar un valor predefinido en la definición del campo en la tabla).

Para los archivos tradicionales, cada programa tenía que contener el diseño completo del registro lógico, aunque no se usaran todos los datos, ya que el archivo lógico sólo contenía los datos. En una base de datos, además de los datos y los índices que permiten vincular los campos de un mismo registro lógico y los que permiten los accesos secuenciales y directos para cada tabla, también se almacenan los diseños completos de todas las tablas que se encuentran en esa base de datos, conformando lo que se denomina diccionario de datos. Todos estos datos también pueden ser leídos por un programa. Para cada tabla incluida en la base de datos, se guarda, entre otros datos, el nombre de la tabla; para cada uno de sus campos, el tipo de dato, su longitud, valor predeterminado, si es obligatorio, reglas de validación, relación con otros campos de otras tablas, etcétera y, además, el o los índices que permiten accesos directos más rápidos. Finalmente, en las bases de datos también se pueden guardar procedimientos que pueden ejecutar las aplicaciones, deliberadamente llamados para su ejecución o en forma automática, como los denominados *stored procedures* (o procedimientos almacenados, que permiten centralizar procesos en la base de datos y que los distintos programas ejecuten iguales procedimientos, por ejemplo, validaciones de datos) o los *triggers* (o disparadores, que son procedimientos que se ejecutan automáticamente cuando se produce una operación de inserción, modificación o eliminación).

Los SABD pueden tener otras características muy atractivas y de gran utilidad como, por ejemplo, que las bases de datos pueden ser distribuidas (partes almacenadas en distintos lugares físicos), pueden tener réplicas (copias) automáticas, copias que pueden funcionar como resguardo automático frente a la caída de una de ellas, de una manera similar al esquema RAID para discos magnéticos y de manera combinada para lograr mayor rendimiento (para la lectura, las dos bases de datos son iguales y puede leerse cualquiera de las dos) y el manejo de transacciones.

Una transacción es un conjunto de cambios (inserciones, modificaciones y eliminaciones) que se realizan sobre la base de datos con motivo de una operación y que se tratarán como una unidad. Por ejemplo, al realizar una factura o nota de venta

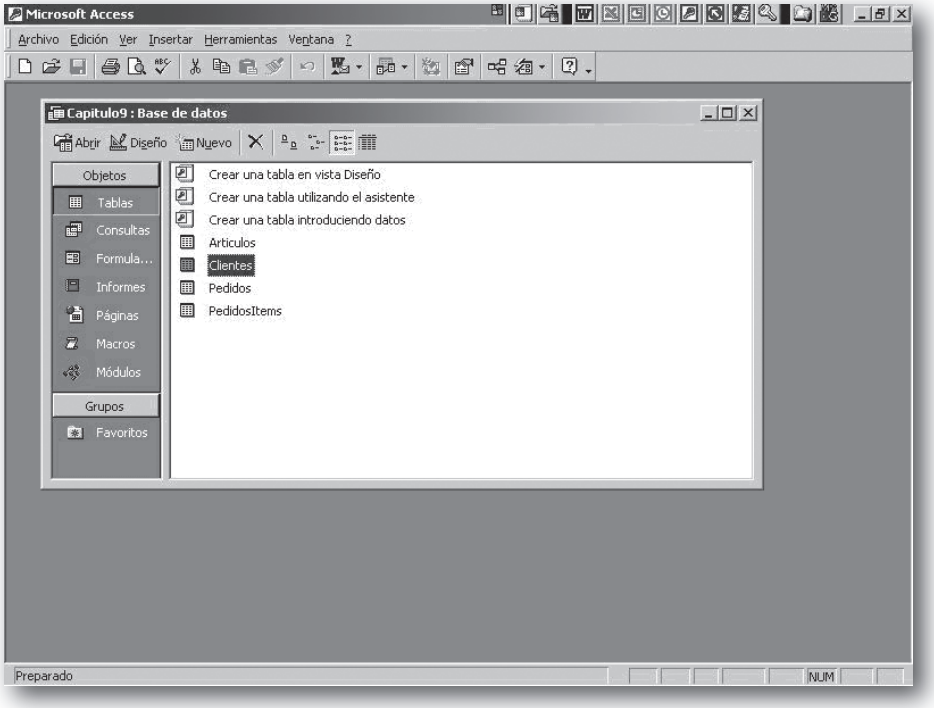
se deberá grabar la factura en la tabla de cuentas corrientes, actualizar el saldo del cliente, grabar las salidas de cada uno de los artículos que componen la factura, actualizar los saldos de los artículos, actualizar el pedido, etcétera. En medio de esas inserciones y actualizaciones de datos puede ocurrir un corte de suministro eléctrico, la falla de la computadora o del disco o cualquier accidente que impida culminar de hacer todos los cambios requeridos. Un programa puede indicar el comienzo y el fin de las actualizaciones de la base de datos que componen la transacción (factura), de manera que si no se culmina exitosamente de realizarlas, ya sea en forma automática por el SABD o bien por indicación del programa, todos los cambios realizados dentro de la transacción hasta el momento, serán revertidos (operación de *rollback*). De esta manera, aseguramos que, al realizar una factura, o bien todas las actualizaciones están realizadas o no está ninguna (todos los cambios y agregados de datos se tratan como una unidad). Imaginemos las dificultades que se ocasionaban cuando las actualizaciones de una factura quedaban por la mitad (primero había que ver hasta dónde habían llegado a realizarse).

Otro problema posible de integridad de los datos aparece si distintos usuarios desean modificar los mismos datos al mismo tiempo. Este acceso concurrente puede ser manejado directamente por el SABD o por los programas, a través de los denominados bloqueos optimistas y pesimistas.

En las **Figuras 10.4** y **10.5** se pueden ver las tablas del ejemplo del Punto 10.3 y la definición de una de dichas tablas, utilizando MS Access.

Figura 10.4

Tablas del ejemplo del Punto 10.3 en MS Access



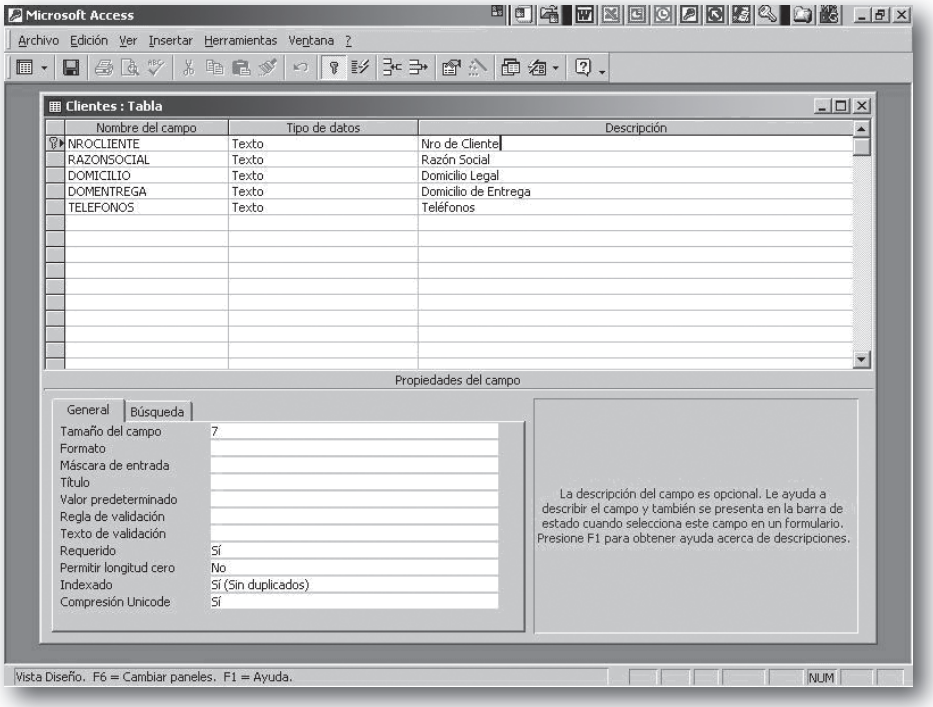


Figura 10.5

Definición de una tabla del Punto 10.3 en MS Access

Hay distintos sistemas de administración de bases de datos. Algunos son de uso gratuito como MS SQL Server Express, MySQL o PostGreSQL, semi-gratuitos como el Microsoft Access (viene incluido en Microsoft Office) y pagos como MS SQL Server, Oracle, DB2 (IBM) o Sybase.

Cada uno de ellos tiene sus ventajas y desventajas, según las necesidades y según las opiniones, básicamente referidas al costo, la *performance*, las limitaciones, la plataforma sobre la que corre, la facilidad de instalación, configuración y uso, las funcionalidades que contempla (incluyendo seguridad) y los productos complementarios que dispone. De todos modos, parafraseando el dicho, hay dos clases de SABD: aquellos de los que la gente se queja y aquellos que nadie usa.

Los SABD tienen distintos componentes básicos, adicionales y desarrollados por terceros. Los más habituales son (ver **Figura 10.6**, pág 212):

a) Motor

Es el componente que realiza todas las operaciones físicas sobre la base de datos. Es el componente que más incide en el rendimiento. También se encarga de controlar las restricciones de integridad ya descritas en el Punto 10.2.

b) Diccionario de datos

Contiene las definiciones de todas las tablas de la base de datos, como ya ha sido señalado. Esta información se guarda habitualmente en tablas de sistema y frecuentemente ocultas, aunque pueden ser leídas por los programas de aplicación.

c) Procesador de consultas

Es un componente esencial de un SABD. El lenguaje de consultas estructurado SQL (del inglés *Structured Query Language*) es la forma más habitual o a veces única, para poder realizar operaciones sobre la base de datos, tanto en forma directa por parte de un usua-

rio, como a través de los programas de aplicación que también utilizan estas sentencias de almacenamiento o recuperación de datos. Es un lenguaje no procedural.

Existe un SQL estándar, aunque los distintos desarrolladores de SABD siempre realizan agregados o cambios que diferencian levemente la sintaxis del SQL para ese SABD en particular. Si bien los cambios no son grandes, pero pueden incluir por ejemplo el nombre con que se identifica un tipo de dato numérico, de fecha o de texto.

El lenguaje SQL tiene básicamente dos grandes tipos de sentencias: las sentencias de definición de datos, DDL (del inglés *Data Definition Language*) y las sentencias de manipulación de datos, DML (del inglés *Data Manipulation Language*). Las primeras permiten definir tablas, campos, e índices (por ejemplo, CREATE, DROP, ALTER), mientras que el segundo grupo permiten generar "vistas lógicas" de los datos combinando una o más tablas y también insertar, modificar y eliminar registros o conjuntos de registros (por ejemplo, SELECT, INSERT, UPDATE, DELETE). Las "vistas lógicas" son conjuntos de datos de una o más tablas (pueden ser sólo algunos datos de algunas tablas combinadas por campos comunes), que son definidas por una sentencia SQL y que son "vistos" como si fueran una tabla real específica, que el motor construye en el momento, que es transitoria y desaparece al terminar la aplicación o cuando ésta decide cerrar esa vista.

Por ejemplo:

```
• Sentencia para crear la tabla ARTÍCULOS del ejemplo (DDL)
/* Script para la creación de Bases de Datos SQL Server */
/*  */
/* Objeto:  Tabla Artículos */
CREATE TABLE ARTICULOS(
             NROARTICULO VARCHAR(20) NOT NULL DEFAULT '',
             DESCRIPCION VARCHAR(50) NULL DEFAULT '',
             PRECIO FLOAT NULL DEFAULT 0,
             UNIDADMEDIDA VARCHAR(2) NULL DEFAULT '',
             CODBARRAS VARCHAR(13) NULL DEFAULT ''
)
/* Clave Primaria */
ALTER TABLE ARTICULOS WITH NOCHECK ADD
CONSTRAINT IA PRIMARY KEY CLUSTERED
(NROARTICULO)

• Sentencia para obtener una vista lógica de todos los Artículos (DML)
SELECT * FROM ARTICULOS ORDER BY NROARTICULO

      (el * indica que se obtengan todos los campos)

•  Sentencia para obtener todos los Artículos con Unidad de Medida = KG, aunque sólo los datos NROARTICULO y DESCRIPCION (DML)
SELECT NROARTICULO, DESCRIPCION FROM ARTICULOS WHERE UNIDADMEDIDA = 'KG'
ORDER BY DESCRIPCION

 (sólo se obtienen los campos NROARTICULO y DESCRIPCIÓN de
      todos los artículos, cuya UNIDADMEDIDA sea KG)
```

d) Generador de reportes

Tan importante como el almacenamiento y la organización de los datos, es la recuperación de los mismos para satisfacer las necesidades de los usuarios. Este componente permite que, de una manera ágil y sencilla y sin necesidad de líneas de código, un usuario pueda definir reportes o listados con los datos necesarios.

Normalmente, el reporte gira alrededor de una consulta que es analizada por el procesador de consultas y ejecutada por el motor, devolviendo el conjunto de datos solicitado por la sentencia de consulta. El generador de reportes formatea dichos datos con la presentación definida por el usuario. Normalmente, la presentación es definida indicando la disposición de los datos en el informe, los títulos, el nivel de detalle, el ordenamiento y los totales que deben incluirse, además de logos, líneas, colores, etcétera que puedan resultar necesarios para destacar el contenido o mejorar su apariencia estética.

e) Generador de formularios

De una manera sencilla (como arrastrar y soltar), este componente puede crear formularios simples para el ingreso y la búsqueda de datos. También utilizará el procesador de consultas, el diccionario de datos y el motor para su funcionamiento.

f) Generador de aplicaciones

Básicamente, una aplicación es un conjunto de opciones de ingreso de datos, procesamiento de los mismos y presentación de datos e información en informes o listados por pantalla y/o en forma impresa. Hemos visto que otros componentes pueden generar formularios de ingreso de datos, procesamientos de diferentes tipo e instancias y formatear datos e información de salida en reportes de acuerdo a las necesidades del usuario. Sin embargo, una aplicación reúne estas opciones de una manera integrada, ágil y sencilla de usar, a través de menús, barras de herramientas, ayuda contextual, etcétera, típicas de cualquier software que utilicemos.

Este componente justamente permite reunir esas opciones individualmente realizadas, con estos elementos, y conformando una aplicación organizada y desarrollada con los componentes y herramientas del SABD.

Por supuesto que una aplicación realizada de esta manera, resultará simple si la comparamos con otras aplicaciones del mundo de los negocios, pero pueden ser útiles para obtener ventajas en la utilización de la tecnología de la información para actividades especiales para las que no se haya desarrollado un software específico o para actividades de poca frecuencia de uso o urgentes, para las cuales la ineficiencia en la utilización de recursos de estas aplicaciones y su bajo nivel de seguridad, no sea demasiado importante en comparación con la rapidez y bajo costo de su generación.

g) Comunicación e integración

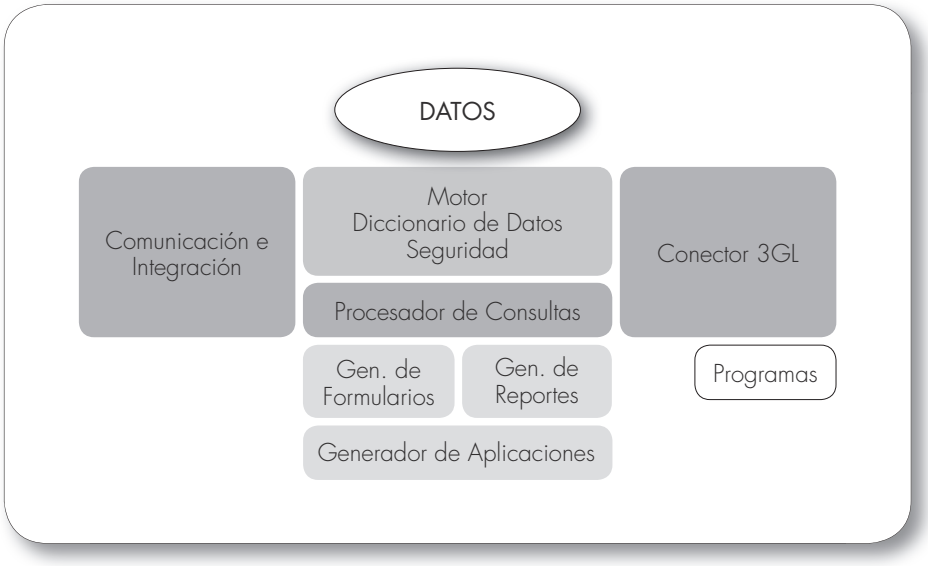
Algunos SABD tienen algunas características especiales que permiten almacenar y recuperar datos de distintas bases de datos físicas o lógicas (réplicas), que pueden estar en distintas computadoras e incluso en distintos lugares físicos. Además, hay componentes que permiten que los lenguajes de programación puedan conectarse con el procesador de consultas, diccionario de datos y con el motor de la base de datos. Las aplicaciones desarrolladas con lenguajes de programación, generalmente utilizan sentencias SQL para armar consultas que les permiten grabar los datos en las diferentes tablas, así como obtener conjuntos de datos o "vistas" lógicas de los datos almacenados que simplifican notablemente las tareas de manejo de datos de las aplicaciones.

h) Seguridad y utilidades

Por supuesto, no pueden faltar componentes que permitan administrar seguridad en cualquier tipo de acceso a la base de datos, así como privilegios o permisos a usuarios para realizar determinadas acciones (ver **Figura 10.6**). También los SABD pueden implementar las restricciones de integridad vistas en el Punto 10.2 de tal manera que distintos programas respeten las mismas reglas. El manejo de transacciones descrito también permite elevar el nivel de integridad de la base de datos.

Adicionalmente, existen módulos que permiten la administración de usuarios de la base de datos, las clásicas tareas de resguardo y recupero (*backups*), así como herramientas de monitoreo de rendimiento o para mostrar la estrategia de procesamiento frente a una sentencia SQL (que permite a desarrolladores mejorar los tiempos de ejecución de consultas complejas).

Figura 10.6
Componentes



Una descripción sencilla de las ventajas de los SABD sería decir que elimina o reduce, sustancialmente, los problemas enunciados al analizar el entorno tradicional de archivos.

Por las características explicadas, existe una alta independencia entre la disposición física de los datos y como los programas acceden a ellos. Como se pueden agregar campos a las tablas existentes y nuevas tablas sin que los programas deban ser necesariamente modificados, el uso de SABD facilita enormemente el desarrollo de aplicaciones y su posterior mantenimiento, permitiendo una gran flexibilidad en la administración de las estructuras de datos.

La simplicidad en la utilización de las bases de datos y en la comprensión de su definición disminuyó notablemente las dificultades que encontraban distintos equipos de trabajo para compartir datos entre distintas aplicaciones y, consecuentemente, se redujo la redundancia de datos (siempre puede haber alguna redundancia pero seguramente con más consistencia y uniformidad en las duplicaciones) y aumentó la capacidad de integración, al resolverse problemas de codificaciones diferentes para los mismos artículos o clientes, códigos de operaciones diferentes para las mismas operaciones o códigos iguales para operaciones distintas, por ejemplo. De esta manera pueden integrarse datos de distintas operaciones y aplicaciones, permitiendo obtener información de alcance más amplio sobre artículos, clientes, operaciones, etcétera.

Las funcionalidades relacionadas con la seguridad que proveen los SABD permiten el desarrollo de distintas aplicaciones, con criterios y reglas de seguridad e integridad más uniformes que en el entorno tradicional de archivos, donde cada aplicación definía y manejaba esquemas de seguridad que podían ser muy diferentes entre sí. Por ejemplo, la posibilidad de leer sólo algunos datos de una tabla hace que el resto de los datos de la tabla no sean transferidos a lo largo de la red ni que se encuentren en memoria principal de la computadora del usuario.

10.6 INTELIGENCIA DE NEGOCIOS

En el Capítulo 5 ya hemos visto las principales características de las distintas herramientas que se agrupan como inteligencia de negocios. Estos software tienen como objetivo la exploración y explotación de datos para dar soporte a la toma de decisiones. Las principales herramientas son los reportes, el análisis multidimensional (también denominado frecuentemente OLAP, del inglés *On Line Analitical Processing* o procesamiento analítico en línea), los tableros de control o comando y cuadro de mando integral (BSC o *Balanced Scorecard*) y la minería de datos (en inglés *Data Mining*).

Ahora que ya hemos visto las características de las bases de datos, podemos analizar algunos aspectos técnicos.

Como ya hemos señalado, datos de distintas fuentes se compilan formando un almacén de datos (en inglés *Data Warehouse*). En su mayoría, esos datos provienen del procesamiento transaccional de la organización (OLTP). Los procesos para incluir esos datos en el *Data Warehouse* se denominan ETL (del inglés *Extract*, *Transform* and *Load*), es decir extracción, transformación y carga. Estos procesos se encargan de obtener los datos necesarios, de transformarlos de acuerdo a las necesidades posteriores de procesamiento de inteligencia de negocios y para que puedan ser más integrados y de cargarlos en el almacén de datos (ver **Figura 10.7**).

Los datos que se incluyen en un *Data Warehouse*, están enfocados para resolver las necesidades de información y se utilizarán con algún software de inteligencia de negocios. Además, los datos deben integrarse para lograr una mejor información del conjunto de la organización (como parte de la transformación puede resultar necesario cambiar códigos o valores de distintas fuentes). También se van acumulando históricamente (para poder analizar tendencias o evoluciones) y consecuentemente, los datos no son volátiles (los datos ya almacenados no van a ser modificados). Los procesos ETL se realizan en forma programada, a intervalos regulares de tiempo y de tal manera de no entorpecer el procesamiento transaccional.

Se denomina *Data Mart* (mercado de datos) a un subconjunto de datos del *Data Warehouse*, más acotado o más específico sobre algún tema o área de la organización. Se habla de *Data Mart* cuando se desea indicar un alcance parcial, es decir, una base de datos que sólo cubre un subconjunto de las necesidades de este tipo de procesamiento. Desde el punto de vista de la metodología de trabajo, no hay diferencias entre *Data Mart* y *Data Warehouse*, salvo por el hecho de que, en general, siempre ha dado buenos resultados abordar problemas complejos dividiéndolos en varios sub problemas más pequeños. De todos modos, si se encara de esta manera, no debe perderse de vista la totalidad de las necesidades.

Estas necesidades siempre van a ser muy particulares de cada organización; aun cuando dos empresas se dediquen a lo mismo y participen del mismo mercado, otros factores como su posición en el mercado, su organización, modelo de negocios, cultura organizacional y personal directivo, puede hacer que las necesidades sean muy diferentes. Consecuentemente, utilizar modelos pre armados sin realizar un trabajo de análisis y diseño puede llevar el proyecto al fracaso.

Técnicamente, sería posible aplicar las herramientas de inteligencia de negocios sobre las bases de datos transaccionales. Pero, como hemos visto, el uso de este tipo de herramientas no es simplemente obtener reportes gerenciales agregados y/o integrados. En muchos casos, los resultados obtenidos hacen que un gerente quiera ahondar en la información u obtener otra que pueda estar relacionada; y ese proceso se realiza operando, en el momento, sobre la base de datos (por ejemplo, en análisis multidimensional).

Si los usuarios operativos y los gerentes trabajaran sobre la misma base de datos, es casi seguro que sus procesos terminarían siendo muy lentos para sus necesidades (por lo general, en inteligencia de negocios se trabaja con grandes volúmenes de datos). Por otro lado, las bases de datos transaccionales se diseñan pensando en todos los datos que se deben manejar en las distintas operaciones de la organización (incluyendo muchos datos que no se necesitarán para estas herramientas, como, por ejemplo, los datos impositivos), y en sus necesidades de rendimiento.

La consecuencia es que estas bases de datos no son muy eficientes para inteligencia de negocios y tendrán tiempos de respuesta más lentos al usar cualquiera de sus herramientas. Por último, los almacenes de datos guardan frecuentemente datos de muy diversas fuentes (no sólo del procesamiento transaccional) internas y externas, con registros históricos de más antigüedad, muchas veces con redundancia controlada (para que los procesos sean más rápidos), con tablas diseñadas pensando en las herramientas que se utilizarán y con características de mayor integración organizacional.

Para una mayor eficiencia en el uso de este tipo de software, generalmente un *Data Warehouse* se arma como si fuera un cubo multidimensional. Las herramientas OLAP permiten obtener información interactivamente utilizando distintas perspectivas. El usuario puede "desagregar" (*drill*) los datos para obtener más detalles o "resumir" (*roll*) la información obteniendo totales o "pivotear" para cambiar de perspectiva de análisis.

El diseño de un cubo OLAP (multidimensional) se basa en almacenar "medidas" de distintas "dimensiones". Las medidas son datos como la cantidad o el valor de venta y las dimensiones (lados del cubo) son atributos, como, por ejemplo, el artículo o producto, el cliente, fecha, etcétera. Una "tabla de hechos" es generalmente una tabla de detalle de operaciones, por ejemplo de ventas, que contendrá las medidas o valores y varias otras columnas que son las dimensiones (que tienen sus propias tablas).

Existen dos tipos de diseños OLAP: diseño de estrella y diseño de copo de nieve. En el diseño de estrella, las tablas de dimensiones (artículos, clientes, etcétera) se relacionan en forma directa con la tabla de hechos. En cambio, en el diseño de copo de nieve pueden existir niveles en las tablas de dimensiones y, por lo tanto, puede no haber una relación directa (por ejemplo, país, provincia, ciudad, cliente). Un modelo mixto puede tener dimensiones de ambos tipos.

Desde otro punto de vista, si utilizamos una base de datos relacional, estaremos en presencia del denominado ROLAP (del inglés *Relational OLAP*) y si utilizamos una base de datos multidimensional, recibe el nombre de MOLAP (del inglés *Multidimensional OLAP*). Para evitar entrar en detalles muy técnicos, imaginemos una base de datos multidimensional como una base de datos de una sola tabla; los registros de esa tabla tienen múltiples campos de dimensiones y de valores o medidas.

Las herramientas ROLAP tienen como ventaja una mayor escalabilidad para manejar grandes volúmenes de datos, menores tiempos de carga en la actualización de la base de datos y el hecho de usar una base de datos relacional (seguramente la misma que para el procesamiento transaccional), pero tienen como desventaja un menor rendimiento que las MOLAP.

Para el lector interesado en profundizar estos conceptos, le recomendamos la lectura del libro de Ernesto Chinkes, *Business Intelligence para mejores decisiones de negocio*.

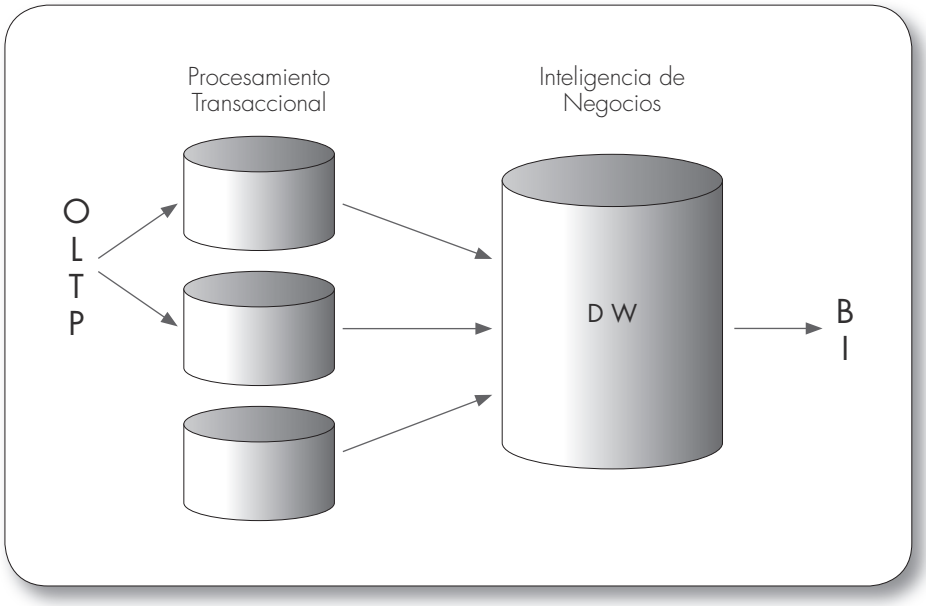


Figura 10.7
Inteligencia de negocios

CAPÍTULO 11

CICLO DE VIDA Y MODELOS DE DESARROLLO

CAPÍTULO 12

METODOLOGÍAS DE ANÁLISIS Y DISEÑO

ALCANCE

Esta Parte aborda los temas ciclo de vida y metodologías de desarrollo.

El Capítulo 11 presenta el concepto de ciclo de vida señalando la complementación entre los diferentes enfoques de este concepto, las tecnologías de gestión de proyectos y las metodologías de desarrollo para luego presentar los principales modelos de ciclo de vida, destacando las diferencias entre ellos y sus principales características.

El Capítulo 12 desarrolla el concepto de metodología para el análisis y diseño de sistemas, conceptualizando los objetivos, características y herramientas principales de la metodología estructurada, y la metodología orientada a objetos.